Efficient, Secure, and Intelligent Wireless Systems: From Edge AI to Network Management

by

Mohammad Aaron Chegini
B.Sc., Shahid Beheshti University, 2023
M.A.Sc., University of Victoria, 2026

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

We acknowledge and respect the Lək ənən (Songhees and X sepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək ənən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Efficient, Secure, and Intelligent Wireless Systems: From Edge AI to Network Management

by

Mohammad Aaron Chegini
B.Sc., Shahid Beheshti University, 2023
M.A.Sc., University of Victoria, 2026

Supervisory Committee

Dr. Amirali Baniasadi, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Amirali Baniasadi, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

In one sentence: this thesis makes AI efficient, secure, and explainable enough for real-time wireless systems.

The vision of AI-native 6G networks is blocked by three barriers. First, state-of-the-art deep learning models are too large and power-hungry for battery-powered edge receivers, where modulation schemes can change every few milliseconds and must be classified in real time. Second, the edge hardware that hosts these models is vulnerable to physical denial-of-service attacks that degrade performance without any software breach. Third, when network faults occur, diagnosis remains a slow, manual process because existing AI systems cannot explain their reasoning to engineers. This thesis presents a unified framework that addresses all three barriers.

To solve the efficiency problem, we introduce **RFNet**, a lightweight neural network for Automatic Modulation Classification that reduces model size by over 90% compared to standard baselines. We validate this on real hardware with **Tiny-RFNet**, deployed on an NVIDIA Jetson Orin Nano, where it processes the full 255,590-sample RadioML 2018.01A test set in approximately 30 s under 3 W, fast enough to track adaptive modulation changes in real time.

To address the security problem, we present **NoCSNet**, a deep learning framework that detects thermal attacks on chip interconnects with 93.8% accuracy, catching malicious patterns that evade conventional threshold-based monitors.

To enable interpretable diagnostics, we introduce **TRACE**, a cascaded system for 5G root cause analysis. TRACE achieves 99.65% diagnostic accuracy on the

TeleLogs benchmark, outperforming even 32-billion-parameter Large Language Models (95.86%), while running in milliseconds on a single CPU and providing transparent reasoning traces that engineers can verify.

Together, these contributions demonstrate that trustworthy AI for next-generation wireless requires co-design across efficiency, security, and interpretability.

Contents

List of Tables

List of Figures

LIST OF ABBREVIATIONS

3GPP	Third Generation Partnership Project
5G/6G	Fifth/Sixth Generation (mobile networks)
AMC	Automatic Modulation Classification
APSK	Amplitude Phase-Shift Keying
BPSK	Binary Phase-Shift Keying
CNN	Convolutional Neural Network
DoS	Denial of Service
DTM	Dynamic Thermal Management
FB	Feature-Based (methods)
FLOP	Floating-Point Operation
FM	Frequency Modulation
FP32	32-bit Floating Point
gNodeB	Next-generation Node B (5G base station)
GPU	Graphics Processing Unit
INT8	8-bit Integer
IoT	Internet of Things
I/Q	In-phase and Quadrature
KPI	Key Performance Indicator
LB	Likelihood-Based (methods)
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSC	Multiscale Convolutional (layer)

NAS	Neural Architecture Search
NoC	Network-on-Chip
PCI	Physical Cell Identity
PSK	Phase-Shift Keying
QAM	Quadrature Amplitude Modulation
QAT	Quantization Aware Training
QPSK	Quadrature Phase-Shift Keying
RAG	Retrieval-Augmented Generation
RCA	Root Cause Analysis
ReLU	Rectified Linear Unit
RF	Radio Frequency
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
SCB	Separable Convolution Block
SFT	Supervised Fine-Tuning
SINR	Signal-to-Interference-plus-Noise Ratio
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
TRACE	Tiered Rule-Augmented Classification Engine
VGG	Visual Geometry Group (network architecture)

ACKNOWLEDGEMENTS

Completing this thesis has been one of the most challenging and rewarding experiences of my life, and it would not have been possible without the support of many remarkable people.

I am deeply grateful to my supervisor, **Dr. Amirali Baniasadi**, for his mentorship, patience, and genuine investment in my growth as a researcher. His guidance extended far beyond academics, and I am a better thinker for it.

I would like to thank my committee member, **Dr. Mihai Sima**, and my external examiner, **Dr. Sean Chester**, for their time, their sharp questions, and the care they brought to every meeting. Their feedback shaped this work in ways I continue to appreciate.

To Pouya, the reason I am here in Victoria, BC. For all his support, mentorship, being there in my ups and downs, and being my guide and north star. To Samaneh, for all her compassion and guidance. To both of them, for their collaboration on this work.

To my colleagues, Ana, Bernardo, Diogo, Leo, Victor, and Yehor, thank you for making the long days feel shorter and the hard problems feel solvable. The friendships built in that room are ones I will carry long after the degree.

To my friends here in Victoria, Ahmad, Ali G., Ali M., Amirehossein Sh., Amirreza E., Arian, Daniel, Fatemeh, Mahmoud, Mahsa, Marcela, Maryam, Meisam, Paige, Pezhman, and Todd, thank you for bringing warmth and peace to my life in Canada. To my dear friends back home, Amin, Amirali, Amirmohammad, Leo, and Sina, I am forever grateful to have you in my life. And to the countless others whose names may not appear here but whose kindness I carry with me, thank you. You have all shaped who I am.

To the Iranian community here in Victoria and beyond, your resilience, warmth, and solidarity have been a constant reminder of home. Knowing you were close made the distance easier.

I am grateful for the financial support of **NSERC** during the first year of my studies, and partial funding from the **UVic Sustainability Program** and **Anne-Marie Daniels** for a project during my Master's degree.

And to my family, you sacrificed so that I could have choices you never had. Every page of this thesis belongs to you.

DEDICATION

To my mother and father, for raising me to be who I am today.
To my father, who showed me how to be patient and resilient,
and to my mother, who showed me how to be a compassionate human.

To Iran, the land that shaped me, the good people who inspire me,
the food and culture that are extraordinary,
and all those who never stopped dreaming of a better tomorrow, despite everything.

To Canada, for giving me the opportunity,
and to all of my friends, regardless of where they reside or where they are from,
in Canada, in Iran, or anywhere else in the world.
I appreciate you all.

Chapter 1

Introduction

The telecommunications industry is undergoing a profound transformation. As 5G networks reach maturity and research into 6G systems accelerates, the fundamental nature of wireless infrastructure is shifting from static, hardware-defined networks toward dynamic, software-defined, and increasingly intelligent ecosystems. This shift is driven by rapidly growing global demand for connectivity: from high-bandwidth video streaming and immersive augmented reality to the ultra-reliable, low-latency communications required by autonomous vehicles and industrial automation.

However, this explosion in capability comes at the cost of unprecedented complexity. Modern wireless systems must manage a scarce and fragmented spectrum, mitigate interference in dense heterogeneous deployments, and optimize performance across a diverse array of user equipment. Traditional approaches cannot keep up. They were designed for simpler, more predictable networks; today's systems change too fast and have too many variables for hand-crafted rules to manage. In response, Deep Learning (DL) has emerged as a transformative solution, offering the ability to learn complex, non-linear representations directly from data and automate decision-making across the entire wireless stack, from the physical layer to network management.

Yet, the promise of AI-driven autonomous wireless networks remains largely unfulfilled. The transition from research prototypes to real-world deployment is hindered by three formidable barriers: the computational inefficiency of state-of-the-art models, the vulnerability of edge hardware to physical security threats, and the lack of interpretability in automated network management. This thesis addresses these challenges by proposing a comprehensive framework for efficient, secure, and intelligent wireless systems.

This thesis argues that trustworthy edge AI for wireless systems requires co-design across three dimensions: *efficiency* (models that fit within power and memory budgets), *security* (hardware resilient to physical attacks), and *interpretability* (diagnostic systems whose reasoning can be verified). Chapters 3 and 4 address efficiency through lightweight neural network design and edge deployment. Chapter 5 addresses security through deep learning-based attack detection on the hardware interconnect. Chapter 6 addresses interpretability through a cascaded reasoning system for network troubleshooting. Together, these contributions demonstrate that AI-native wireless networks demand a holistic approach: optimizing across the full stack, from silicon to network management.

1.1 Motivation

The motivation for this work is grounded in the convergence of three critical trends that define the current landscape of wireless research and deployment.

1.1.1 The Edge Intelligence Gap

The first challenge lies in the tension between model accuracy and computational efficiency. In the domain of Automatic Modulation Classification (AMC), a foundational task for cognitive radio and spectrum monitoring, Deep Learning has demonstrated superior performance compared to traditional likelihood-based methods. Convolutional Neural Networks (CNNs) can robustly identify modulation schemes even in the presence of severe channel impairments. However, this accuracy typically comes at a steep price: state-of-the-art models often require millions of parameters and billions of floating-point operations (FLOPs) per inference.

This computational burden is fundamentally incompatible with the deployment reality of edge devices. Wireless sensors, IoT endpoints, and mobile handsets operate under strict power, memory, and thermal constraints [33]. A model that achieves 95% accuracy but consumes 10 Watts of power or requires 500 MB of memory is practically useless for a battery-powered receiver. Consequently, there is a widening "Edge Intelligence Gap" between the sophisticated models developed in academic research and the lightweight, efficient architectures required for real-world deployment. Bridging this gap requires a paradigm shift in neural network design, moving away from brute-force scaling toward principled, hardware-aware architectures that maximize

feature extraction efficiency.

The need for efficient inference is not merely a matter of deployment convenience; it is a fundamental requirement of the wireless physical layer. In adaptive modulation systems, now standard in 5G NR and Wi-Fi 6, the modulation scheme can change on a per-frame or per-slot basis (every few milliseconds) in response to channel quality feedback. A cognitive receiver must classify these changing modulation formats in real time to demodulate the signal correctly. A model that requires 100 ms per inference is useless for a system where the modulation changes every 10 ms. This temporal constraint, the need to classify faster than the channel can change, is what makes sub-millisecond, low-power inference essential, not optional.

1.1.2 The Hardware Security Blind Spot

The second challenge arises from the physical vulnerability of edge intelligence. As we push more computation to the edge, we expose the underlying hardware to new vectors of attack. Unlike cloud data centers, which are physically secured and thermally managed, edge devices are often deployed in accessible, uncontrolled environments. This physical accessibility makes them susceptible to side-channel attacks and, more specifically, thermal Denial-of-Service (DoS) attacks.

Modern edge System-on-Chips (SoCs), such as those powering intelligent base stations or autonomous robots, rely on Network-on-Chip (NoC) interconnects to facilitate communication between processor cores and accelerators. These interconnects are the central nervous system of the chip. A malicious actor, by injecting carefully crafted traffic patterns, can induce localized hotspots that trigger the system's thermal throttling mechanisms. This form of attack is insidious: it does not require root access or cryptographic breaches, yet it can degrade system performance, violate real-time guarantees, or even cause permanent physical damage. Despite this risk, the intersection of AI deployment and hardware security remains underexplored. A truly robust wireless system must be secure not just in its software algorithms, but in the physical substrate that executes them.

1.1.3 The Complexity of Network Management

The third challenge moves up the stack to the network management plane. 5G networks generate large volumes of telemetry data: signal strength measurements, throughput logs, handover statistics, and alarm events. When performance degrades

(for example, when a user's throughput drops unexpectedly), identifying the root cause is a search problem of massive dimensionality. It could be a physical issue (antenna downtilt), a configuration error (neighbor list mismatch), a resource allocation problem (congestion), or environmental interference.

Currently, Root Cause Analysis (RCA) is a manual, labor-intensive process performed by human experts. It is slow, subjective, and unscalable. Machine learning could automate this, but there is a catch: engineers will not trust a "black-box" system. Before authorizing a network change, they need to understand *why* the system made its recommendation. Furthermore, the sheer variety of potential faults requires a reasoning capability that goes beyond simple pattern matching. The emergence of Large Language Models (LLMs) and neuro-symbolic AI offers a new opportunity to build "reasoning agents" that can parse complex data, apply domain logic, and provide explainable diagnoses. However, LLMs have a critical weakness: they cannot reliably perform the precise numerical computations (distances, angles, modular arithmetic) that network diagnosis requires. Recent work shows that LLM error rates increase by up to 14 percentage points when reasoning over unfamiliar numbers [24]. A single arithmetic mistake can flip a diagnosis from "interference" to "misconfiguration." This suggests that the path to automated RCA lies not in end-to-end language models, but in hybrid architectures that combine deterministic domain rules with interpretable machine learning.

1.2 Research Questions

The overarching goal of this thesis is to develop a comprehensive framework for deploying trustworthy AI in wireless systems. This framework must satisfy three requirements: (1) models efficient enough for battery-powered edge devices operating under real-time constraints, (2) hardware secure against physical denial-of-service attacks, and (3) diagnostic systems interpretable enough for operational trust. From this goal, we derive four research questions:

Driven by these motivations, this thesis seeks to answer four interconnected research questions that span the lifecycle of an intelligent wireless system:

1. **Efficiency:** Can we design neural network architectures for Automatic Modulation Classification that achieve competitive accuracy with significantly fewer parameters and lower computational cost than existing state-of-the-art models?

2. **Deployment:** Do these efficient architectures maintain their performance and efficiency advantages when deployed on real-world edge hardware (e.g., NVIDIA Jetson) under strict latency and power constraints?

3. **Security:** How can we secure the edge hardware platforms that host these AI models against physical threats, specifically thermal Denial-of-Service attacks targeting the Network-on-Chip interconnect?

4. **Intelligence:** Can we automate the diagnosis of complex network faults (Root Cause Analysis) using interpretable, reasoning-enhanced machine learning models that combine domain heuristics with data-driven inference?

1.3 Contributions

This thesis makes the following specific contributions to the field of intelligent wireless systems:

- **RFNet (Chapter 3):** We propose RFNet, a family of lightweight Convolutional Neural Networks designed explicitly for the AMC task. By introducing two novel architectural components, the Multiscale Convolutional (MSC) layer for multi-resolution feature extraction and the Separable Convolution Block (SCB) for parameter-efficient learning, RFNet achieves classification accuracy comparable to heavy ResNet-based baselines while reducing the parameter count by over 90% [5].

- **Tiny-RFNet (Chapter 4):** We validate the practical feasibility of our efficient designs by implementing and profiling RFNet on the NVIDIA Jetson Orin Nano, a representative edge AI platform. We provide a comprehensive characterization of the accuracy-latency-power trade-off space, demonstrating that RFNet enables real-time modulation classification on constrained hardware [4].

- **NoCSNet (Chapter 5):** We address the hardware security gap by proposing NoCSNet, a deep learning-based framework for detecting thermal DoS attacks in NoC architectures. We contribute a publicly available dataset of NoC traffic under thermal attack and demonstrate that Recurrent Neural Networks (RNNs) achieve 93.8% detection accuracy, effectively identifying malicious thermal patterns that evade traditional threshold-based detection [1].

- **TRACE (Chapter 6):** We introduce a cascaded, explainable RCA system for 5G networks. Using the TeleLogs dataset, we demonstrate that a hybrid architecture combining deterministic domain rules with interpretable machine learning achieves 99.65% diagnostic accuracy, outperforming even the largest fine-tuned LLMs (95.86% for Qwen2.5-RCA-32B [21]). We further show that this approach provides the interpretability required for operational deployment.

1.4 Thesis Organization

The remainder of this thesis is organized to follow the logical progression from the physical layer up to the management plane:

Chapter 2 establishes the theoretical foundations. It provides a narrative review of the evolution of Automatic Modulation Classification, the principles of efficient deep learning, the threat landscape of Network-on-Chip security, and the emerging role of AI in network management.

Chapter 3 presents the design and evaluation of the RFNet architecture. It details the mathematical formulation of the MSC and SCB layers and provides a rigorous comparative analysis against existing AMC models on the RadioML benchmark.

Chapter 4 moves from simulation to reality. It describes the Tiny-RFNet framework, detailing the hardware-aware optimization process and presenting empirical results on latency, throughput, and power consumption from the Jetson Orin Nano.

Chapter 5 investigates the security of the hardware platform. It details the generation of the NoCSNet dataset, the threat model of thermal DoS attacks, and the design of deep learning detectors to mitigate them.

Chapter 6 ascends to the network management layer. It presents the TRACE (Tiered Rule-Augmented Classification Engine) system for 5G Root Cause Analysis, detailing the feature engineering process and the cascaded decision logic that enables automated, explainable diagnosis.

Chapter 7 concludes the thesis by synthesizing the key findings, discussing the limitations of the current work, and outlining a roadmap for future research in secure and intelligent wireless systems.

Chapter 2

Background and Related Work

This chapter provides the technical foundation for the four research contributions presented in this thesis. The discussion is organized into four sections, each addressing a core pillar of the work: Automatic Modulation Classification (Section 2.1), edge inference for embedded systems (Section 2.2), Network-on-Chip security (Section 2.3), and Large Language Models for network management (Section 2.4). The narrative arc of this chapter mirrors the structure of the thesis itself. Sections 2.1 and 2.2 establish the signal-processing and hardware-efficiency foundations that motivate the designs in Chapters 3 and 4. Section 2.3 introduces the hardware-security context for Chapter 5. Section 2.4 presents the network-management landscape that frames Chapter 6. Each section traces the evolution of its respective field, identifies the specific limitations of prior work, and concludes with a gap statement that the corresponding thesis chapter addresses.

2.1 Automatic Modulation Classification

Automatic Modulation Classification (AMC) is the task of identifying the modulation scheme of a received radio signal without prior knowledge of the transmission parameters. This capability is essential for cognitive radio, dynamic spectrum access, electronic warfare, and regulatory spectrum monitoring, where a receiver must adapt its demodulation strategy on the fly. Before a signal can be decoded, its modulation format, whether Binary Phase-Shift Keying (BPSK), Quadrature Amplitude Modulation (QAM), or Frequency Modulation (FM), must be determined. AMC therefore serves as the critical bridge between signal detection and information recovery. This

section reviews the progression of AMC from classical statistical methods to modern deep learning architectures, identifies the principal benchmarking datasets, and highlights the efficiency limitations that motivate the work in Chapter 3.

2.1.1 Traditional AMC: Likelihood and Feature-Based Methods

Prior to the adoption of deep learning, AMC research was dominated by two complementary paradigms: Likelihood-Based (LB) methods and Feature-Based (FB) methods. LB classifiers formulate the problem as a multiple hypothesis test. Given an observed signal vector, the classifier computes the likelihood of the observation under each candidate modulation and selects the one that maximizes the posterior probability. Under ideal conditions, where the channel model, noise distribution, and signal parameters are perfectly known, LB methods achieve the theoretically optimal Bayesian error rate. In practice, this optimality is rarely realized because LB methods require accurate knowledge of nuisance parameters such as carrier frequency offset, phase offset, and signal-to-noise ratio. When these parameters deviate from their assumed values, the likelihood functions become misspecified, and classification accuracy degrades significantly.

FB methods avoid the need for explicit probabilistic models by extracting discriminative statistical features from the received signal and feeding them into a conventional classifier such as a decision tree or support vector machine. Commonly used features include higher-order cumulants, which capture non-Gaussian structure in the signal constellation, and cyclostationary statistics, which exploit the periodic redundancy introduced by digital modulation. For example, the fourth-order cumulant C_{40} takes distinct values for PSK, QAM, and FSK signal families, providing a compact discriminant. Similarly, the variance of the instantaneous frequency and the kurtosis of the signal envelope have been used to separate analog from digital modulations. While FB approaches are more computationally tractable than LB methods and more tolerant of parameter uncertainty, they suffer from a fundamental limitation: the features must be hand-engineered by domain experts for each deployment scenario. A feature set that distinguishes PSK variants under additive white Gaussian noise may fail entirely in the presence of multipath fading, frequency-selective channels, or hardware impairments such as I/Q imbalance. Adding new modulation types to the classifier requires redesigning the feature extraction pipeline, a process that demands

both signal-processing expertise and extensive empirical validation. This brittleness, coupled with the explosion of modulation formats in modern standards (5G NR alone supports over a dozen QAM and OFDM configurations), created a pressing need for methods that could learn discriminative representations directly from data.

2.1.2 The Deep Learning Paradigm Shift

The application of deep learning to AMC marked a fundamental departure from the feature-engineering paradigm. Rather than designing statistical features by hand, deep neural networks learn hierarchical representations directly from raw In-phase and Quadrature (I/Q) signal samples, automatically discovering the features most relevant for classification. This data-driven approach proved not only more convenient but also more powerful, as the learned features capture subtle signal characteristics that hand-crafted statistics overlook.

O'Shea, Corgan, and Clancy [17] pioneered this direction in 2016 by demonstrating that Convolutional Neural Networks (CNNs) could perform AMC on raw I/Q data. Their architecture treated the two-channel I/Q input as a narrow image and applied two convolutional layers followed by dense layers to classify eleven modulation types. On the RML2016.10a dataset, this approach achieved an average accuracy of approximately 79.3%, substantially outperforming the best feature-based baselines of the time. The significance of this result extended beyond the accuracy figure: it proved that deep models could extract discriminative features that human experts had never considered, such as subtle correlations between the I and Q channels induced by specific pulse-shaping filters. The primary limitation of this early work was architectural: the CNN was adapted from image-classification designs (resembling a shallow VGG network) and was not optimized for the sequential, time-varying nature of RF signals, resulting in a model with a relatively high parameter count for the modest accuracy it delivered.

Recognizing the need for rigorous benchmarking, O'Shea, Roy, and Clancy [18] introduced the RadioML 2018.01A dataset in 2018, which rapidly became the de facto standard for AMC research. This dataset contains 2,555,904 signal frames uniformly distributed across 24 digital and analog modulation classes, including high-order schemes such as 256-QAM and several APSK variants that are critical for high-throughput 5G and 6G links. Each frame comprises 1024 complex-valued I/Q samples generated under a comprehensive channel model that incorporates multipath

fading, carrier frequency offset, sample rate offset, and additive white Gaussian noise across 26 Signal-to-Noise Ratio (SNR) levels ranging from −20 dB to +30 dB in 2 dB increments. The scale and realism of RadioML 2018.01A forced models to learn representations that are robust to the distortions encountered in real over-the-air transmission. Earlier datasets, such as RML2016.10a with only 11 modulation types and 128-sample frames, lacked the diversity and signal length necessary to evaluate models on high-order modulations or long-range temporal dependencies. The 1024-sample frame length of RadioML 2018.01A is particularly important because it allows models to observe multiple symbol periods and thereby exploit the temporal structure of the modulation for classification. The introduction of this dataset standardized comparison across studies, enabling the community to track genuine architectural progress rather than dataset-specific tuning.

2.1.3 Deeper Architectures and Recurrent Models

Following the foundational CNN results, researchers explored both deeper convolutional architectures and recurrent neural networks to improve classification performance.

West and O'Shea [30] investigated the use of deeper architectures for modulation recognition, applying residual networks (ResNets) that employ skip connections to mitigate the vanishing gradient problem. ResNets add identity shortcut connections that bypass one or more layers, allowing gradients to flow directly through the network during backpropagation. Their experiments demonstrated that depth improves the network's capacity to distinguish visually similar high-order modulations, such as 16-QAM and 64-QAM, whose constellations are geometrically nested. A 16-QAM constellation is a strict subset of a 64-QAM constellation; in the presence of noise, the outer points of a 64-QAM signal can be mistaken for 16-QAM, a confusion that shallow networks cannot resolve but deeper models with greater representational capacity can learn to disambiguate. By enabling the training of networks with six or more convolutional layers, residual connections allowed the model to learn increasingly abstract signal representations. The limitation of this approach, however, was that the accuracy gains came at substantial computational cost: the deeper models required significantly more parameters and floating-point operations per inference, making them impractical for resource-constrained deployment.

Rajendran et al. [19] took a fundamentally different approach by applying Long

Short-Term Memory (LSTM) networks to AMC in the context of distributed spectrum sensing with low-cost sensors. LSTMs are recurrent architectures designed to capture long-range temporal dependencies in sequential data, making them theoretically well-suited for radio signals where phase evolution and frequency drift manifest over many sample periods. Their work showed that LSTM-based classifiers could operate effectively on raw I/Q sequences and were particularly adept at classifying frequency-modulated signals whose temporal structure carries the discriminative information. The main drawback was computational: the sequential nature of recurrent inference precludes the parallelism that makes CNNs efficient on modern hardware accelerators, and the memory overhead of maintaining hidden states across long sequences was prohibitive for edge devices.

2.1.4 Multi-Scale and Hybrid Architectures

As the field matured, researchers recognized that neither pure CNNs nor pure RNNs captured the full complexity of modulated signals. Radio signals exhibit features at multiple temporal scales: short-term phase transitions occur over a handful of samples at the symbol rate, while channel-induced effects such as frequency offset and fading manifest over hundreds or thousands of samples. Architectures capable of simultaneously extracting features at these disparate scales emerged as the next generation of AMC models.

Huynh-The et al. [12] proposed MCNet, a multi-scale convolutional network that processes the input signal through parallel convolutional branches with distinct kernel sizes. The design draws conceptual inspiration from the Inception architecture [26], which demonstrated the effectiveness of parallel multi-scale processing in image classification. In MCNet, each branch applies convolutions with a different kernel size (for example, kernel sizes of 3, 5, and 7) to the same input, capturing short-range phase transitions with the small kernel and longer-range frequency trends with the large kernel. The outputs of all branches are concatenated along the channel dimension to form a rich multi-scale representation that is passed to subsequent layers for further processing. On the RadioML 2018.01A benchmark, MCNet achieved an overall accuracy of approximately 92.4%, representing a significant improvement over earlier single-scale CNNs. The multi-scale design of MCNet is directly relevant to this thesis because it provided the architectural inspiration for the Multiscale Convolutional (MSC) layer introduced in Chapter 3. RFNet extends the MCNet concept

by combining multi-scale feature extraction with depthwise separable convolutions, achieving comparable accuracy at a fraction of the computational cost. The limitation of MCNet was its relatively high parameter count; while the multi-scale branches improved accuracy, they also increased the computational footprint, leaving the efficiency question unresolved.

Wang et al. [29] proposed a hybrid CNN-LSTM architecture that combined the spatial feature extraction strengths of convolutional layers with the temporal modeling capabilities of recurrent layers. In this design, CNN layers first transform the raw I/Q input into a sequence of compact feature vectors, which are then fed into an LSTM to capture the temporal evolution of those features across the signal frame. The interactive fusion of one-dimensional and two-dimensional convolutional filters enabled the model to compensate for the limitations of single-stream processing. This hybrid approach demonstrated that the spatial and temporal dimensions of RF signals are complementary and that models exploiting both can outperform architectures that rely on only one. The downside was increased architectural complexity and inference latency, as the model required both convolutional and recurrent computation paths.

2.1.5 Benchmarking and the Low-SNR Challenge

Recent work has pushed the accuracy frontier on RadioML 2018.01A to remarkable heights. Harper, Thornton, and Larson [9] conducted an extensive ablation study exploring the effects of dilated convolutions, statistical pooling layers, and network depth on AMC performance. Dilated convolutions expand the receptive field of a convolutional layer without increasing the number of parameters, by inserting gaps between kernel elements. A dilation rate of d causes a kernel of size K to span $K+(K-1)(d-1)$ samples, enabling the network to integrate information over a wider temporal context while performing the same number of multiply-accumulate operations as a standard convolution. Their systematic exploration of architectural choices, including the number of convolutional blocks, filter widths, and pooling strategies, produced a best configuration that achieved a peak accuracy of 98.9% at high SNR, establishing a new performance ceiling for the benchmark. The key insight of this work was that the choice of pooling strategy and the receptive field size are more influential than raw network depth for AMC, a finding that informs the design of efficient architectures. Despite this impressive peak accuracy, the model's performance at low SNR (below 0 dB) remained limited, as signal energy in that regime is dominated by noise.

Two comprehensive survey papers consolidate the state of the field and provide essential context for the related work in this thesis. Huynh-The et al. [13] provided an extensive categorization of deep architectures for AMC, organizing the literature by network type (CNN, RNN, hybrid, generative) and identifying the key design trade-offs between accuracy, complexity, and robustness. Their survey covered over 100 studies published between 2016 and 2021, documenting the rapid progression from simple two-layer CNNs to sophisticated attention-based architectures. They identified three open challenges: the need for standardized evaluation protocols, the lack of attention to computational efficiency, and the difficulty of deploying models in real-time systems. Zhang et al. [32] complemented this with a survey focused on models and datasets, benchmarking common architectures on a unified evaluation platform and highlighting the sensitivity of reported results to dataset-specific preprocessing choices such as normalization, data augmentation, and train-test splitting. Their analysis revealed that differences in preprocessing alone can account for several percentage points of reported accuracy variation between studies, underscoring the importance of standardized benchmarks like RadioML 2018.01A.

Together, these surveys and benchmarking studies establish that while peak accuracy on clean, high-SNR signals is largely a solved problem, two fundamental challenges persist. First, all models struggle below 0 dB SNR, where the signal is buried in thermal noise. This is a fundamental physical limitation rather than a model deficiency; even the theoretically optimal Bayesian classifier cannot reliably distinguish modulations when the signal power is a fraction of the noise power. At -20 dB SNR, the noise power is 100 times the signal power, and the I/Q samples are effectively random; no classifier, regardless of its complexity, can extract meaningful information from pure noise. Second, and more critically for this thesis, the pursuit of ever-higher accuracy has produced models with millions of parameters and billions of floating-point operations per inference. The most accurate models require desktop-class GPUs for real-time operation, creating a fundamental mismatch between where AMC is needed (portable, battery-powered devices) and where the models can actually run (stationary, high-power computing infrastructure).

Gap. Despite the impressive accuracy achieved by deep learning, the existing AMC literature assumes the availability of powerful computing hardware, typically high-end desktop GPUs, for inference. No prior work has systematically addressed the design of AMC architectures that are simultaneously accurate and efficient enough for deployment on battery-powered edge devices with strict power and mem-

ory constraints. This gap motivates Chapter 3, which introduces RFNet, a family of lightweight CNNs that achieve competitive accuracy on RadioML 2018.01A with fewer than 4,000 parameters through the novel Multiscale Convolutional layer and Separable Convolution Blocks.

2.2 Edge Inference for Embedded Systems

The deployment of deep learning models on resource-constrained edge devices, a paradigm often termed Edge AI or TinyML, requires a fundamentally different design philosophy from the cloud-centric approach that dominates mainstream deep learning research. Edge platforms, ranging from ARM Cortex-M microcontrollers with kilobytes of RAM to NVIDIA Jetson modules with integrated GPUs, operate under simultaneous constraints on power consumption (often below 15 W), memory footprint, thermal dissipation, and inference latency. A model that achieves record accuracy on a desktop GPU is worthless if it cannot execute within these budgets. This section reviews the three principal strategies for enabling efficient inference at the edge: lightweight architectural primitives, model compression techniques, and hardware-aware deployment. The discussion identifies the critical disconnect between theoretical efficiency metrics and real-world on-device performance, a gap that Chapter 4 addresses.

2.2.1 Depthwise Separable Convolutions: MobileNets

The single most significant architectural innovation for mobile deep learning is the depthwise separable convolution, introduced by Howard et al. [11] in the MobileNet architecture. A standard convolutional layer with C_{in} input channels, C_{out} output channels, and a spatial kernel of size K requires a filter tensor of shape $C_{\text{in}} \times C_{\text{out}} \times K$ and performs $C_{\text{in}} \cdot C_{\text{out}} \cdot K \cdot D_F^2$ multiply-accumulate operations per output feature map of spatial dimension D_F. Depthwise separable convolution decomposes this operation into two stages. The first stage, the depthwise convolution, applies a single spatial filter of size K independently to each input channel, producing C_{in} filtered feature maps at a cost of $C_{\text{in}} \cdot K \cdot D_F^2$ operations. The second stage, the pointwise convolution, applies a 1×1 convolution to linearly combine the depthwise outputs across channels, at a cost of $C_{\text{in}} \cdot C_{\text{out}} \cdot D_F^2$ operations. The total computational cost of a depthwise

separable convolution is therefore:

$$\text{Cost}_{\text{DSC}} = C_{\text{in}} \cdot K \cdot D_F^2 + C_{\text{in}} \cdot C_{\text{out}} \cdot D_F^2 \tag{2.1}$$

The ratio of the depthwise separable cost to the standard convolution cost is:

$$\frac{\text{Cost}_{\text{DSC}}}{\text{Cost}_{\text{Std}}} = \frac{1}{C_{\text{out}}} + \frac{1}{K} \tag{2.2}$$

For a typical configuration with $K = 3$ and $C_{\text{out}} = 64$, this ratio evaluates to approximately $\frac{1}{64} + \frac{1}{3} \approx 0.35$, yielding a computational reduction by a factor of roughly 8 to 9. MobileNet achieved 70.6% top-1 accuracy on ImageNet with only 4.2 million parameters and 569 million FLOPs, a dramatic reduction compared to VGG-16's 138 million parameters and 15.5 billion FLOPs. This factorization is central to the Separable Convolution Blocks (SCB) employed in RFNet (Chapter 3), where depthwise separable convolutions are adapted from two-dimensional image processing to one-dimensional RF signal processing.

2.2.2 Inverted Residuals: MobileNetV2

Sandler et al. [22] refined the MobileNet design with two key innovations: inverted residual blocks and linear bottlenecks. In a conventional residual block, data flows from a wide representation through a narrow bottleneck and back. MobileNetV2 inverts this: the block first expands a narrow input to a wide representation using a 1×1 pointwise convolution, applies the depthwise convolution in the expanded space, then projects back to a narrow bottleneck. The expansion ratio t (typically $t = 6$) controls intermediate width. The skip connection links the narrow input directly to the narrow output, enabling efficient gradient flow while keeping skip tensor memory footprint small.

Crucially, Sandler et al. demonstrated that applying ReLU to low-dimensional bottleneck representations destroys information, because ReLU collapses negative activations to zero in a space with few dimensions. Replacing ReLU with a linear activation in the bottleneck layers preserved representational capacity and improved accuracy. MobileNetV2 achieved 72.0% top-1 accuracy on ImageNet with only 3.4M parameters and 300M FLOPs. The memory efficiency of inverted residuals is notable: skip connections operate on narrow bottleneck tensors, so peak memory is determined by bottleneck width rather than expansion width. This design principle influenced

Tiny-RFNet in Chapter 4.

2.2.3 Compound Scaling: EfficientNet

While MobileNets addressed the design of efficient building blocks, Tan and Le [27] tackled a complementary question: given an efficient base architecture, how should it be scaled up to achieve higher accuracy? Conventional practice scaled networks along a single dimension, either making them deeper (more layers), wider (more channels), or increasing the input resolution. Tan and Le observed that these three dimensions are interdependent and that scaling them in isolation yields diminishing returns. They proposed a compound scaling method that uniformly scales depth d, width w, and resolution r using a single compound coefficient ϕ:

$$d = \alpha^{\phi}, \quad w = \beta^{\phi}, \quad r = \gamma^{\phi} \tag{2.3}$$

subject to the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, which ensures that the total FLOPs scale approximately as 2^{ϕ}. The base coefficients α, β, and γ are determined through a small grid search on the baseline model (EfficientNet-B0). Applying this method, EfficientNet-B0 achieved 76.3% top-1 accuracy on ImageNet with 5.3 million parameters, while EfficientNet-B7 reached 84.4% accuracy, matching or exceeding models that were an order of magnitude larger. The compound scaling principle is relevant to this thesis because it demonstrates that efficiency and accuracy are not inherently opposing objectives; principled scaling laws can navigate the trade-off systematically. However, these scaling laws were derived for two-dimensional image data and do not directly transfer to one-dimensional RF signals, where the relationships between depth, width, and temporal resolution are governed by different signal physics.

2.2.4 Weight Pruning

Beyond architectural design, model compression techniques reduce the cost of a pre-trained network by removing or simplifying its components. Han, Pool, Tran, and Dally [8] established the foundational methodology for neural network pruning in a three-step process: train the full network to convergence, prune connections whose weights fall below a magnitude threshold (effectively zeroing them out), and retrain the remaining connections to recover any lost accuracy. This iterative train-prune-retrain cycle exploits the observation that large neural networks are over-

parameterized, containing many redundant connections that contribute little to the output. Applying this methodology to AlexNet, Han et al. reduced the number of parameters by a factor of 9 without measurable accuracy loss. On VGG-16, the compression factor reached 13×. Pruning is particularly valuable for edge deployment because it reduces both the storage requirements (fewer non-zero weights to store in flash memory) and the number of multiply-accumulate operations (zero-valued weights produce zero-valued products that can be skipped). Chapter 3 applies magnitude-based pruning to the RFNet architecture, demonstrating that 50–60% of the already-compact SCB weights can be removed while maintaining classification accuracy.

2.2.5 Quantization Aware Training

Quantization reduces the numerical precision of weights and activations, typically from 32-bit floating point (FP32) to 8-bit integers (INT8). This transformation enables inference on hardware with dedicated integer arithmetic units, which are faster, more power-efficient, and more area-efficient than floating-point units. Jacob et al. [14] formalized the Quantization Aware Training (QAT) framework, which simulates the effects of quantization during the forward pass of training by inserting "fake quantization" nodes into the computation graph. During the forward pass, activations and weights are rounded to their quantized values, introducing quantization noise that the network learns to tolerate. During the backward pass, gradients are computed with respect to the full-precision values using the straight-through estimator, allowing the optimizer to adjust the weights to be robust to the precision loss. This approach enabled integer-only inference that was approximately 2 to 3 times faster than floating-point execution on mobile CPUs, with negligible accuracy degradation for models like MobileNetV2 and InceptionV3. QAT is essential for deploying AMC models on digital signal processors (DSPs) and field-programmable gate arrays (FPGAs) that lack floating-point hardware. Chapter 3 applies QAT to the RFNet architecture, demonstrating that 8-bit quantized models match the accuracy of their 32-bit counterparts on the RadioML 2018.01A benchmark.

2.2.6 Knowledge Distillation

Knowledge distillation, introduced by Hinton et al. [10], transfers the learned representations of a large "teacher" model to a compact "student" model. The key insight

is that the teacher's soft probability outputs, the full distribution over all classes rather than just the hard argmax label, contain rich information about the relative similarities between classes. For instance, a teacher classifying a noisy 16-QAM signal might assign probabilities of 0.7 to 16-QAM, 0.2 to 64-QAM, and 0.05 to 32-QAM. These soft targets reveal that 16-QAM and 64-QAM are geometrically similar constellations, information that is lost when the label is simply "16-QAM." The distillation loss function combines the standard cross-entropy loss with the Kullback-Leibler divergence between the student's and teacher's softened outputs:

$$\mathcal{L} = (1-\alpha)\cdot\mathcal{L}_{\mathrm{CE}}(y,\sigma(z_s)) + \alpha\cdot T^2\cdot D_{\mathrm{KL}}\Big(\sigma\Big(\frac{z_t}{T}\Big)\Big\|\sigma\Big(\frac{z_s}{T}\Big)\Big) \tag{2.4}$$

where z_s and z_t are the student and teacher logits, σ denotes the softmax function, T is the temperature parameter that controls the smoothness of the probability distribution, α balances the two loss terms, and D_{KL} is the Kullback-Leibler divergence. Higher values of T produce softer distributions that reveal more inter-class structure. Hinton et al. demonstrated that a single compact model trained with distillation could match the performance of a large ensemble of models, achieving significant compression with minimal accuracy loss. The T^2 scaling factor compensates for the magnitude reduction of gradients at high temperatures, ensuring that the distillation signal contributes meaningfully to the optimization. While distillation was not applied in the RFNet work presented in this thesis, it represents a complementary compression avenue for future iterations of efficient AMC models and is included here for completeness.

2.2.7 Hardware Deployment: The Theory-to-Practice Gap

The techniques described above, efficient architectures, pruning, quantization, and distillation, are typically evaluated using theoretical metrics: parameter count, FLOPs, and model size. These metrics are useful proxies for efficiency but do not directly predict on-device performance. A model with fewer FLOPs is not guaranteed to be faster, because inference latency depends on memory access patterns, hardware utilization, instruction-level parallelism, and the efficiency of the runtime framework. For example, depthwise convolutions, despite their low FLOP count, often underutilize GPU tensor cores because of their low arithmetic intensity (the ratio of compute operations to memory operations). A depthwise convolution with C channels and kernel size K performs only K multiplications per memory access, compared to $C\cdot K$ for

a standard convolution, resulting in a memory-bound operation that cannot saturate the available compute units. This means that a model with 50% fewer FLOPs might achieve only a 20% latency reduction if the remaining operations are memory-bound.

NVIDIA Jetson platforms, which combine ARM CPUs with integrated CUDA GPUs and are optimized through the TensorRT inference engine, represent the primary hardware target for real-time edge AI in this thesis. The Jetson Orin Nano, used in Chapter 4, provides up to 40 TOPS of INT8 inference performance within a 15 W power envelope, bridging the gap between microcontroller-class TinyML platforms and server-grade GPUs. TensorRT performs layer fusion (combining sequential operations into single GPU kernels to reduce launch overhead), precision calibration (automatically converting FP32 models to FP16 or INT8 with minimal accuracy loss), and kernel auto-tuning (selecting the fastest GPU kernel implementation for each layer given the specific tensor dimensions). The effectiveness of these optimizations varies across architectures and quantization schemes; some model topologies benefit dramatically from TensorRT optimization while others see modest improvements. Real-world validation on the target hardware is therefore indispensable: the only reliable measure of edge performance is a direct measurement of latency, throughput, and power consumption on the physical device.

Gap. The efficient deep learning techniques reviewed in this section were developed for computer vision and have not been systematically adapted to or validated for RF signal processing. Furthermore, a critical disconnect exists between theoretical compression metrics (parameter count, FLOPs) and actual on-device performance (latency, throughput, power consumption). No prior AMC study reports real-time inference measurements on physical edge hardware. Chapter 4 addresses this gap by deploying the RFNet architecture on the NVIDIA Jetson Orin Nano, measuring actual latency, throughput, and power consumption, and demonstrating that the theoretical efficiency of RFNet translates to practical real-time AMC on a commercial edge platform.

2.3 Network-on-Chip Security

The edge devices on which efficient AMC models are deployed, System-on-Chip (SoC) platforms such as the NVIDIA Jetson series, rely on internal communication fabrics to route data between processor cores, memory controllers, and hardware accelerators. As SoC designs scale to integrate hundreds of heterogeneous processing elements,

the traditional shared-bus interconnect becomes a bottleneck that limits throughput and scalability. Network-on-Chip (NoC) architectures have emerged as the standard solution, replacing monolithic buses with a packet-switched micro-network on the silicon die. While this architectural shift enables the performance scaling demanded by modern AI workloads, it simultaneously introduces a new class of hardware-level security vulnerabilities. This section reviews the fundamentals of NoC architecture, the threat models specific to NoC-based SoCs, the mechanism of thermal Denial-of-Service (DoS) attacks, and the emerging role of machine learning in detecting such threats. The discussion identifies the lack of public datasets and robust detection frameworks for thermal attacks, a gap that Chapter 5 addresses.

2.3.1 Fundamentals of Network-on-Chip Architecture

The conceptual foundations of NoC were established in two seminal papers. Benini and De Micheli [2] articulated the "Networks on Chips" paradigm. They argued that as transistor feature sizes scaled into the nanoscale regime, global wire delays began to dominate gate delays, making traditional bus-based communication the primary performance bottleneck. Their solution was to treat the SoC as a micro-network of communicating components, adapting the layered protocol abstractions of macroscopic networking to the on-chip environment. This architectural vision introduced network interfaces, routers, and layered protocol stacks that decouple computation from communication, enabling modular and scalable SoC design. Dally and Towles [7] provided the complementary engineering perspective with their influential argument to "route packets, not wires." They demonstrated quantitatively that a packet-switched network consumes less power and area than a set of dedicated point-to-point wires for systems with more than a handful of communicating nodes, because the network amortizes the cost of wiring across multiple communication flows. Their analysis established that NoC architectures offer superior bandwidth density and energy efficiency compared to shared buses and crossbar switches, particularly as the number of processing elements grows.

A standard NoC comprises three primary components. Network Interfaces (NIs) serve as gateways between the processing elements (PEs) and the network, handling the packetization of messages into fixed-size units called flits (flow control units), protocol translation between the PE's native interface and the network protocol, and flow control to prevent buffer overflow. Routers are the switching nodes that direct

packets toward their destinations using routing algorithms such as deterministic XY routing, where packets first traverse the X dimension and then the Y dimension, or adaptive algorithms like Odd-Even routing that dynamically select paths based on network congestion. Each router contains input buffers, a crossbar switch, an arbiter that resolves contention when multiple packets compete for the same output port, and routing logic that determines the next hop for each incoming flit. Physical links, the on-chip wires connecting adjacent routers, determine the raw bandwidth and power consumption of the network. The topology, the geometric arrangement of routers and links, defines the network's diameter (the longest shortest path between any two nodes), bisection bandwidth (the minimum bandwidth across any cut that divides the network in half), and fault tolerance. Common topologies include two-dimensional meshes, which offer regular structure and simple routing but limited path diversity, tori, which add wraparound links to reduce diameter, and hierarchical fat-trees, which provide high bisection bandwidth at the cost of irregular wiring. The choice of topology, routing algorithm, and flow control mechanism collectively determine the NoC's performance, power consumption, and resilience to congestion, factors that become security-relevant when an attacker can influence traffic patterns.

2.3.2 Security Threat Models for NoC-Based SoCs

The transition to NoC architectures expanded the attack surface of SoC platforms. Modern SoCs routinely integrate third-party intellectual property (3P-IP) cores to reduce design cost and time-to-market. These 3P-IP cores, sourced from external vendors, may contain malicious logic, known as Hardware Trojans, that can exploit the NoC's connectivity to launch attacks against other system components. The security of NoC-based systems is typically analyzed through the Confidentiality, Integrity, and Availability (CIA) triad. Confidentiality threats include eavesdropping on data packets as they traverse shared routers and links, enabling an attacker to extract sensitive information such as encryption keys or user data. Integrity threats encompass the malicious modification of packet headers or payloads, which can cause data corruption, misrouting, or spoofing. Availability threats, the focus of this thesis, involve the disruption of communication services for legitimate processing elements, preventing them from executing their workloads.

Charles and Mishra [3] provided a comprehensive survey of NoC security attacks and countermeasures, systematically categorizing threats into passive attacks (eaves-

dropping, side-channel monitoring) and active attacks (packet injection, Denial-of-Service, deadlock induction). Their survey identified that the distributed, resource-sharing nature of NoCs, where hardware buffers, arbiters, and links are shared across applications, creates inherent interference potential that attackers can exploit. Beyond eavesdropping and DoS, timing-based side-channel attacks pose a significant threat: Reinbrecht et al. [20] demonstrated Prime+Probe style attacks on NoC-based MPSoCs that exploit traffic interference to infer cache access patterns and extract AES keys, highlighting that the shared NoC fabric can leak sensitive information through timing variations. The survey also highlighted a critical gap in the literature: while numerous countermeasures have been proposed for confidentiality and integrity threats (encryption, authentication, access control), defenses against availability attacks remain comparatively underdeveloped, particularly for emerging threat vectors such as thermally-induced DoS.

2.3.3 Thermal Denial-of-Service Attacks

Among the availability threats to NoC-based SoCs, thermal DoS attacks are particularly insidious because they weaponize the system's own safety mechanisms. Modern high-performance SoCs incorporate Dynamic Thermal Management (DTM) units that continuously monitor on-die temperature through distributed thermal sensors. When a sensor reading exceeds a predefined threshold, DTM triggers protective actions such as Dynamic Voltage and Frequency Scaling (DVFS), which reduces the operating frequency and voltage of the affected region, or in extreme cases, core shutdown. These mechanisms are essential for preventing thermal runaway and silicon damage, but they create a vulnerability that attackers can exploit.

Kong, John, Chung, Chung, and Hu [15] were among the first to analyze thermal attacks in the context of processor microarchitecture, specifically instruction caches. They demonstrated that a malicious process could generate carefully crafted instruction sequences that maximized switching activity in the cache, creating localized hotspots that triggered DTM. The attack exploits a subtle asymmetry: the attacker controls its own workload and can tolerate the performance degradation caused by DTM throttling, while the victim cores, which are executing latency-sensitive legitimate workloads, suffer disproportionately from the same throttling. The key insight was that the attacker does not need to damage the hardware; merely causing the DTM to activate is sufficient to degrade the performance of all co-located processes,

including security-critical workloads.

In the NoC context, this attack vector is realized by a malicious processing element that injects high-intensity traffic into the network, flooding routers and links with packets that maximize switching activity. Each flit that traverses a router causes transistor switching in the crossbar, arbiter, and buffer circuits, dissipating power as heat. Under normal operation, the aggregate switching activity is distributed across the die, and the thermal management system maintains safe temperatures with minimal performance impact. A thermal attacker concentrates this switching activity by directing a high volume of traffic through specific routers, creating localized power densities that exceed the cooling capacity of the die. The resulting thermal hotspots propagate through the silicon substrate to adjacent routers and processing elements via thermal conduction. When the temperature of a legitimate core's region exceeds the DTM threshold, its operating frequency is throttled, degrading its computational throughput and violating real-time guarantees, even though the legitimate core itself is operating correctly. The attack is difficult to counter with logical isolation techniques because heat conduction through silicon is a physical process that respects no logical boundaries between partitions or security domains.

2.3.4 Machine Learning for NoC Security

Traditional approaches to detecting DoS attacks in NoCs rely on static thresholds: if a traffic metric (such as buffer occupancy or flit arrival rate) exceeds a predefined value, an alarm is raised. These threshold-based detectors are fundamentally limited in two ways. First, the thresholds must be set conservatively to avoid false positives from legitimate bursty workloads, which means that low-intensity attacks can evade detection. Second, static thresholds cannot adapt to the dynamic, application-dependent traffic patterns of modern SoCs, where legitimate traffic intensity varies by orders of magnitude depending on the workload.

Sudusinghe, Charles, and Mishra [25] proposed the use of machine learning for DoS attack detection in NoC architectures, training supervised classifiers on traffic features extracted from NoC routers, including flit arrival rates, buffer occupancy levels, and packet latencies. These features are collected at each router over a sliding time window and form a high-dimensional input vector that captures the instantaneous and recent-history state of the local network neighborhood. Their experiments compared multiple ML algorithms, including Random Forests, Support Vector Machines

(SVMs), and K-Nearest Neighbors (K-NN), demonstrating that ensemble methods, particularly Random Forests, achieved high detection accuracy while maintaining low false positive rates. The Random Forest classifier outperformed threshold-based approaches by a wide margin because it could learn non-linear decision boundaries in the multi-dimensional feature space, capturing interactions between features (such as the simultaneous increase in buffer occupancy and decrease in packet latency that characterizes a flooding attack) that simple per-feature thresholds cannot represent. The strength of the ML approach lies in its adaptability: the classifier automatically adjusts its decision boundary to the specific traffic characteristics of the target system, without requiring manual threshold tuning by a security engineer.

Charles and Mishra [3] provided a comprehensive survey that further contextualized these ML-based detection methods within the broader landscape of NoC security countermeasures, including encryption, authentication, access control, and architectural modifications. Their survey identified that while cryptographic countermeasures effectively address confidentiality and integrity threats, they impose significant area and latency overhead and do not directly address availability attacks. ML-based detection, by contrast, can be implemented as a lightweight monitor that observes traffic statistics without modifying the data path, making it suitable for detecting DoS and thermal attacks with minimal performance impact. The limitation of the existing ML-based detection work, as acknowledged in the survey, was the reliance on proprietary simulation datasets that were not publicly released, hindering reproducibility and comparative evaluation by other researchers.

Gap. Two critical gaps exist in the NoC security literature. First, there is no publicly available dataset for thermal DoS attacks in NoC-based systems, preventing the community from developing and benchmarking detection algorithms on a common foundation. Second, existing detection methods, whether threshold-based or ML-based, have not been evaluated against the specific challenge of thermal attacks, where the attacker's goal is to trigger DTM rather than to directly corrupt or intercept data. Chapter 5 addresses both gaps by introducing the NoCSNet framework, which comprises the first public benchmark dataset of NoC traffic under thermal attack conditions and a deep learning detection system that achieves 93.8% detection accuracy using recurrent neural networks trained on temporal traffic patterns.

2.4 LLMs for Network Management

The complexity of fifth-generation (5G) and future sixth-generation (6G) wireless networks has outpaced the capacity of manual, rule-based management. A single 5G deployment generates terabytes of log data daily, spans hundreds of configuration parameters per cell, and must satisfy stringent Quality-of-Service (QoS) requirements across diverse service profiles. Network operators face an escalating burden: diagnosing performance degradation, identifying root causes, and implementing corrective actions in near real-time. Large Language Models (LLMs) have emerged as a transformative technology for automating these tasks, offering capabilities in natural language understanding, multi-step reasoning, and tool-augmented interaction that align closely with the demands of autonomous network management. This section reviews the foundational LLM techniques that enable tool use and reasoning, examines their adaptation to the telecommunications domain, discusses the specific application of Root Cause Analysis (RCA) for 5G networks, and identifies the limitations that motivate the hybrid approach presented in Chapter 6.

2.4.1 The Automation Imperative in 5G Networks

The 5G New Radio (NR) air interface introduces a level of configurational complexity that is qualitatively different from previous generations. Massive MIMO antenna arrays with dozens or hundreds of elements, dynamic beamforming that adjusts radiation patterns in milliseconds, dense small-cell deployments with inter-cell interference management, and network slicing for heterogeneous service types all contribute to a configuration space that is too large for human operators to navigate manually. A single 5G macro cell may have over 2,000 configurable parameters spanning antenna tilt, beamwidth, handover thresholds, scheduling policies, and power control settings. In a dense urban deployment with thousands of cells, the total configuration space is combinatorially vast, and the interactions between parameters are non-linear and context-dependent.

When a user experiences throughput degradation, the root cause could lie in any of dozens of interacting subsystems: antenna misconfiguration (excessive downtilt reducing coverage), mobility parameter mis-tuning (causing ping-pong handovers), scheduler congestion (insufficient resource block allocation), inter-cell interference (pilot pollution from overlapping coverage), or physical-layer impairments (Doppler spread from high-speed mobility). These causes produce overlapping symptoms in the

observable metrics, making diagnosis from raw Key Performance Indicators (KPIs) alone extremely challenging. Traditional management approaches rely on static alarm thresholds and manually coded decision trees, which cannot capture the complex, non-linear dependencies between these factors. A throughput drop below a threshold triggers an alarm, but the alarm provides no information about the underlying cause, forcing RF engineers to manually correlate drive-test logs, cell configurations, and neighbor relations to isolate the fault. The telecommunications industry's vision of "zero-touch" network management, where the network autonomously detects, diagnoses, and resolves performance issues, requires AI systems capable of processing unstructured data (logs, configuration files, trouble tickets) and performing multi-step causal reasoning.

2.4.2 Tool-Augmented Language Models

The utility of a language model for network management is constrained by its static parametric memory. An LLM trained on a large text corpus possesses broad linguistic and reasoning capabilities but lacks access to real-time network state, specific equipment configurations, or current alarm databases. Tool-augmented LLMs address this limitation by enabling the model to autonomously invoke external tools and APIs during inference.

Schick et al. [23] introduced Toolformer, a framework that teaches language models to use tools through self-supervised learning. The approach works by inserting candidate API calls (for search engines, calculators, translation services, and calendars) into the training corpus and evaluating whether the API result reduces the model's perplexity on subsequent tokens. Only those insertions that demonstrably improve the model's predictions are retained in the fine-tuning dataset. The resulting model learns not only how to formulate API calls but, crucially, when to invoke them, distinguishing situations where external information is needed from those where the model's internal knowledge suffices. For network management, this mechanism provides a principled way for an LLM to decide when to query a telemetry database, retrieve a configuration file, or invoke a diagnostic script, rather than relying on hard-coded tool-invocation rules. The limitation of Toolformer is that it treats each tool invocation as an independent event, without the capacity for iterative, multi-step reasoning that adjusts the tool-use strategy based on intermediate results.

Yao et al. [31] addressed this limitation with the ReAct framework, which in-

terleaves reasoning traces and action steps in a unified generation process. In the ReAct paradigm, the model generates a "Thought" (a natural-language reasoning step about the current state of the problem), followed by an "Action" (an API call or tool invocation), followed by an "Observation" (the result returned by the tool). This Think-Act-Observe loop repeats until the model reaches a conclusion or exhausts its action budget. ReAct demonstrated significant improvements over both pure chain-of-thought reasoning (which lacks access to external information and can only reason over its parametric memory) and pure action-based approaches (which lack the capacity to plan, reflect, and adjust their strategy based on intermediate results).

In a telecommunications context, a ReAct-based agent diagnosing throughput degradation might first think about which metrics are most diagnostic for the reported symptom, then act by querying the base station's performance counters for the relevant cell, observe the returned data showing low SINR despite adequate RSRP, think about what the combination of high signal power and low signal quality implies (likely interference rather than coverage), act by retrieving the neighbor cell list to check for overlapping coverage, observe that multiple non-colocated cells are serving the area with similar signal strengths, and conclude that the root cause is pilot pollution (overlapping coverage). The iterative nature of ReAct makes it well-suited for the diagnostic reasoning required in complex network troubleshooting, where the set of relevant information is not known in advance and must be discovered through interaction with the network's data systems.

2.4.3 Telecom-Specific Large Language Models

General-purpose LLMs, while powerful, often fail to capture the specialized terminology, protocol semantics, and mathematical relationships that define the telecommunications domain. The 3GPP specification corpus alone spans thousands of documents with precise technical language that differs substantially from the web text on which most LLMs are pre-trained. Domain adaptation through continual pre-training and instruction tuning has emerged as the standard approach for bridging this gap.

Zou et al. [34] proposed TelecomGPT, a comprehensive framework for building telecom-specific language models. The pipeline begins with continual pre-training on a massive corpus of 3GPP technical specifications, telecommunications research papers, and patent filings, which injects domain vocabulary and conceptual knowl-

edge into the model's parameters. This is followed by supervised instruction tuning on telecom-specific question-answer pairs and alignment tuning to ensure the model's outputs conform to the precision and formality expected in network engineering contexts. TelecomGPT demonstrated significant improvements over general-purpose models on telecom benchmarks, including protocol comprehension, standards interpretation, and technical question answering. The primary limitation of domain-specific pre-training is cost: the continual pre-training step requires substantial computational resources and access to high-quality domain data, which may not be available for all telecom sub-domains.

Chen et al. [6] proposed NetGPT, a cloud-edge collaborative architecture for deploying LLMs in network operations. Recognizing that a single monolithic model cannot simultaneously satisfy the low-latency requirements of edge-level diagnostics and the broad reasoning capabilities needed for network-wide orchestration, NetGPT deploys smaller, specialized LLMs at the network edge for localized tasks (such as anomaly detection on a single base station) and larger, more capable models in the cloud for complex global reasoning (such as cross-cell interference analysis). The edge models are adapted using Low-Rank Adaptation (LoRA), which fine-tunes a small number of additional parameters while keeping the base model frozen, enabling efficient specialization without the cost of full fine-tuning. This hierarchical architecture addresses the deployment challenge of LLMs in telecommunications, where latency-sensitive decisions must be made at the edge while holistic optimization requires the broad context available only in the cloud. The limitation of NetGPT is that the coordination between edge and cloud models introduces communication overhead and requires careful partitioning of tasks between the two tiers.

2.4.4 Root Cause Analysis for 5G Networks

Root Cause Analysis (RCA) is the process of identifying the underlying cause of a performance anomaly from its observable symptoms. In 5G networks, an RCA system must map a set of observed metrics (low throughput, poor signal quality, high error rates) to a specific causal factor (antenna misconfiguration, interference, congestion) from a potentially large taxonomy of possible causes. Traditional RCA relies on manually coded decision trees or static correlation rules, which are brittle, difficult to maintain, and unable to capture the non-linear interactions between network parameters.

Sana et al. [21] introduced the TeleLogs dataset and a reasoning-enhanced LLM framework for 5G RCA. The TeleLogs dataset provides labeled samples of throughput degradation events collected from 5G drive tests, with each sample annotated with one of eight root causes spanning physical-layer issues (excessive downtilt, overshooting), mobility problems (frequent handovers, missed handovers), interference conditions (overlapping coverage, PCI collisions), and resource limitations (congestion, high vehicle speed). Each sample consists of two complementary data components: a time-series of user-plane drive-test measurements (GPS coordinates, speed, serving cell identity, reference signal power, signal quality, throughput, and neighbor cell measurements) and a static engineering parameter database containing the configuration of every cell in the network (tower location, antenna height, azimuth, mechanical and digital downtilt, and beam pattern). This dual-component structure is essential because many root causes, such as excessive downtilt, cannot be diagnosed from user measurements alone but require knowledge of the antenna's physical configuration.

The framework proposes a two-stage training methodology: supervised fine-tuning (SFT) injects telecom domain knowledge into the model by training it on expert-annotated examples, and reinforcement learning (RL) with reasoning rewards enhances the model's capacity for multi-step diagnostic reasoning by rewarding not just correct answers but also logically coherent reasoning chains. The key contribution is the demonstration that LLMs can produce structured, interpretable diagnostic traces that explain not just what the root cause is, but why the model reached that conclusion. The limitation identified by Sana et al. is that purely data-driven LLM approaches, even with reasoning enhancement, remain "black boxes" whose decision logic is difficult for network engineers to verify against known physical constraints. An LLM might correctly identify "excessive downtilt" as the root cause but for spurious reasons unrelated to the actual antenna geometry, providing a correct answer via incorrect reasoning, a failure mode that is unacceptable in production network operations.

2.4.5 The Numerical Reasoning Limitation of LLMs

Beyond interpretability, a more fundamental challenge limits the applicability of LLMs to quantitative RCA: their unreliable numerical reasoning. Shrestha et al. [24] conducted a systematic evaluation of LLM arithmetic capabilities across wide numerical ranges, revealing that logical error rates increase by up to 14 percentage points

when models encounter numerical values outside their training distribution. Critically, while LLMs demonstrate high accuracy on standalone arithmetic tasks (e.g., "What is 128.188×32.579?"), their performance deteriorates substantially when the same computations are embedded within multi-step word problems, precisely the setting of network RCA where an engineer must simultaneously parse tabular data, compute geodesic distances, evaluate trigonometric beam angles, and perform modular arithmetic on cell identities.

This limitation is particularly acute for 5G RCA because the diagnostic workflow is inherently computation-heavy. A single diagnosis requires: (1) parsing a pipe-delimited table of 10 timestamped rows with up to 19 columns, (2) computing Haversine distances between GPS coordinates and cell tower positions, (3) calculating antenna beam deviation angles using inverse trigonometric functions, (4) evaluating PCI modulo 30 collisions across all neighbor cells at each timestep, and (5) aggregating statistics over a sliding disruption window, all while cross-referencing a separate engineering parameter table. Each step involves precise floating-point arithmetic on domain-specific quantities, and an error in any intermediate computation propagates to the final diagnosis. A concrete example of this failure is presented in Chapter 6, where Mistral Large 3 (675B parameters) [28] incorrectly computes 712 mod 30 and 832 mod 30 during a PCI collision check, missing a collision that deterministic code identifies trivially. The cascaded heuristic-ML approach presented in that chapter avoids this failure mode by implementing all numerical computations as code, reserving learned models for the genuinely ambiguous pattern classification where arithmetic precision is not the bottleneck.

2.4.6 Retrieval-Augmented Generation for Domain-Specific QA

A persistent challenge with LLMs in specialized domains is hallucination: the model generates plausible-sounding but factually incorrect outputs because the correct information is not present in its parametric memory. In telecommunications, where a single misconfiguration can disrupt service for thousands of users, hallucinated recommendations are unacceptable. Retrieval-Augmented Generation (RAG) addresses this problem by augmenting the model's input with relevant documents retrieved from an external knowledge base at inference time. The RAG pipeline operates in two stages. In the retrieval stage, the user's query is encoded into a dense vector

representation and matched against a corpus of pre-indexed documents using approximate nearest-neighbor search. The top-k most relevant passages are retrieved and concatenated with the original query. In the generation stage, the LLM produces its response conditioned on both the query and the retrieved context, grounding its output in authoritative source material.

RAG has demonstrated significant improvements in factual accuracy for domain-specific question answering, because the model can directly cite and reason over authoritative sources rather than relying solely on potentially outdated or incorrect parametric knowledge. In the telecommunications context, RAG enables an LLM-based diagnostic system to consult the latest 3GPP specifications, vendor-specific equipment manuals, and historical incident reports when formulating its diagnosis. For example, when diagnosing a Physical Cell Identity (PCI) collision, a RAG-enhanced system can retrieve the relevant 3GPP clause specifying the DMRS sequence generation rules, verify that the identified collision violates the standard, and include this reference in its explanation to the engineer. This grounding reduces the risk of recommending actions that violate network constraints or contradict established standards. The effectiveness of RAG depends critically on the quality and completeness of the retrieval corpus, as well as the alignment between the query encoding and the document encoding, challenges that remain active areas of research in the NLP community.

Gap. Despite the promise of LLMs for network management, current approaches suffer from two principal limitations that prevent their adoption in production network operations. First, purely black-box models, whether traditional ML classifiers or end-to-end LLMs, lack the interpretability required for high-stakes network operations. When an automated system recommends a configuration change, such as adjusting the antenna tilt of a base station or modifying handover parameters, the network engineer must understand and verify the diagnostic logic before acting on the recommendation. A system that provides only a label ("root cause: excessive downtilt") without explaining the evidence and reasoning chain is insufficient for operational trust. Second, existing LLM-based RCA systems treat all root causes uniformly, applying the same complex reasoning mechanism to faults that have clear, deterministic physical signatures (such as excessive vehicle speed, which can be diagnosed from a single GPS measurement) and those that require complex multivariate analysis (such as distinguishing between overlapping coverage and missed handovers, which share similar symptom profiles in the observable metrics). This uniform treat-

ment is both computationally wasteful and epistemologically inappropriate: deterministic causes should be resolved by deterministic rules, reserving the complexity of learned models for genuinely ambiguous cases. Chapter 6 addresses these gaps by presenting TRACE (Tiered Rule-Augmented Classification Engine), a cascaded RCA system that combines domain-specific rule-based heuristics for deterministic causes with interpretable machine learning for ambiguous causes, achieving 99.65% accuracy on the TeleLogs benchmark while providing a transparent reasoning trace for every diagnosis. This accuracy exceeds even the best fine-tuned LLM approaches (95.86% for Qwen2.5-RCA-32B [21]).

Chapter 3

RFNet: Efficient Neural Networks for Modulation Classification

The work presented in this chapter has been published as: M. Chegini, P. Shiri, and A. Baniasadi, "RFNet: Fast and Efficient Neural Network for Modulation Classification of Radio Frequency Signals," in ITU Journal on Future and Evolving Technologies, vol. 3, no. 2, pp. 261-272, 2022 [5].

Mohammad Chegini's contributions to this work included: conceptualization and design of the RFNet architecture (specifically the MSC and SCB layers), implementation of the model training and evaluation pipeline, execution of all experiments on the RadioML dataset, analysis of pruning and quantization effects, and writing the manuscript.

Co-authors contributed as follows: Pouya Shiri assisted with the experimental setup and provided feedback on the manuscript. Amirali Baniasadi provided supervision and guidance throughout the research process.

3.1 Introduction

Having established the theoretical background of Automatic Modulation Classification (AMC) and the constraints of edge computing in Chapter 2, this chapter presents the first major contribution of this thesis: the design of a neural network architecture that fundamentally reconciles accuracy with efficiency.

AMC is the cornerstone of intelligent radio systems. Before a receiver can demodulate, decode, or analyze a signal, it must first identify the modulation scheme used by

the transmitter, whether it is a simple BPSK signal from a sensor or a complex 256-QAM stream from a high-speed data link. In military and regulatory contexts, this must be done blindly, without handshake or protocol metadata. While Deep Learning (DL) has revolutionized this field by replacing brittle statistical features with learned representations [17], the trend in the literature has been toward increasingly massive models. State-of-the-art architectures often utilize millions of parameters and require billions of floating-point operations (FLOPs) per inference. This computational cost creates a barrier to deployment on the very devices that need AMC the most: battery-powered IoT nodes, drones, and portable spectrum analyzers.

To address this "efficiency gap," we propose **RFNet**, a family of lightweight Convolutional Neural Networks (CNNs) designed from first principles for signal processing efficiency. Rather than simply shrinking a large image-classification network (like ResNet) and expecting comparable results, we introduce two novel architectural components tailored to the physics of RF signals: the **Multiscale Convolutional (MSC)** layer and the **Separable Convolution Block (SCB)**.

The exposition of this chapter is structured as follows: Section 3.2 details the proposed architecture, providing the mathematical formulation and design justification for the MSC and SCB components. Section 3.3 describes the experimental methodology, including the RadioML dataset and the training hyperparameters. Section 3.4 presents a comprehensive evaluation of the model's accuracy, efficiency, and robustness to compression, demonstrating that RFNet achieves competitive performance with a fraction of the computational cost of selected baselines.

This chapter addresses **RQ1 (Efficiency)**, posed in Section 1.2: *Can we design neural network architectures for Automatic Modulation Classification that achieve competitive accuracy with significantly fewer parameters and lower computational cost than existing state-of-the-art models?* The need for efficient inference is not merely a matter of deployment convenience; in adaptive modulation systems, modulation schemes can change every few milliseconds, requiring sub-millisecond, low-power classification to track channel dynamics in real time.

3.2 Proposed Architecture

The design philosophy of RFNet is centered on "spectral-temporal efficiency." Radio frequency signals exhibit features at widely varying scales: short-term phase transitions occur over a few samples, while symbol-level dependencies and frequency drifts

manifest over hundreds of samples. A standard convolutional layer with a fixed kernel size is ill-equipped to capture this diversity efficiently. Furthermore, the standard convolution operation is computationally expensive, performing spatial filtering and channel mixing simultaneously. RFNet addresses these issues through a factorized, multi-scale design.

3.2.1 Multiscale Convolutional (MSC) Layer

The first layer of any CNN processing raw I/Q data is critical, as it transforms the time-domain signal into a high-dimensional feature space. To capture the multi-resolution nature of RF signals, we introduce the Multiscale Convolutional (MSC) layer.

Mechanism

As illustrated in Figure 3.1, the MSC layer replaces the standard input convolution with three parallel processing branches. Each branch $b \in \{1, 2, 3\}$ applies a convolution with a distinct kernel size k_b. Let $\mathbf{X} \in \mathbb{R}^{L \times 2}$ be the input signal of length L with in-phase and quadrature components. The output of branch b is given by:

$$\mathbf{Y}_b = \sigma(\mathrm{BN}(\mathbf{W}_b * \mathbf{X} + \mathbf{b}_b)) \tag{3.1}$$

where $*$ denotes the convolution operation, $\mathbf{W}_b$ is the kernel of size k_b, BN denotes Batch Normalization, and σ is the Rectified Linear Unit (ReLU) activation function. The outputs of the three branches are then concatenated along the channel dimension to form the input to the subsequent layers:

$$\mathbf{Y}_{\mathrm{MSC}} = [\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3] \tag{3.2}$$

Design Justification

The choice of parallel kernels is motivated by the Inception architecture [26], but adapted for 1D signal processing. In modulation schemes like QPSK or 16-QAM, the information is encoded in phase and amplitude changes that occur at the symbol rate. However, channel impairments like frequency offset can introduce periodicities spanning tens to hundreds of samples. Smaller kernels excel at detecting sharp phase

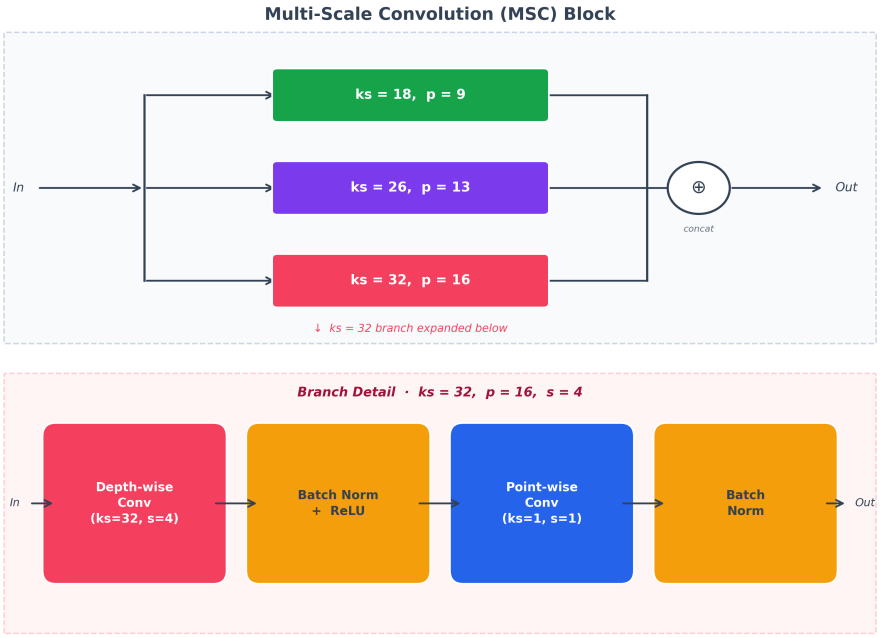

Figure 3.1: The Multi-Scale Convolution (MSC) block architecture. Three parallel 1-D convolutional branches with different kernel sizes (18, 26, and 32) process the input signal simultaneously, and their outputs are concatenated. The inset shows the internal structure of each branch: depthwise convolution, Batch Normalisation, ReLU, followed by pointwise convolution and Batch Normalisation.

transitions, while larger kernels can integrate information over a longer window to estimate frequency trends. The specific kernel sizes (18, 26, and 32 in our implementation, as shown in Figure 3.1) were determined empirically to span the range of relevant temporal scales in the RadioML dataset. By combining these scales early in the network, RFNet avoids the "information bottleneck" that occurs when a single fixed kernel size is forced to compromise between temporal resolution and receptive field.

3.2.2 Separable Convolution Blocks (SCB)

While the MSC layer enriches the initial feature representation, the bulk of the network's computation occurs in the deeper layers that extract abstract features. To minimize the cost of these layers, RFNet employs Separable Convolution Blocks (SCB).

Mechanism

The SCB is built upon the concept of *depthwise separable convolution*. A standard convolution with C_{in} input channels, C_{out} output channels, and kernel size K requires $C_{in} \times C_{out} \times K$ parameters and multiplications per spatial position. The SCB factorizes this into two distinct operations:

1. **Depthwise Convolution:** Applies a single spatial filter of size K to each input channel independently. This requires $C_{in} \times K$ parameters.

2. **Pointwise Convolution:** Applies a 1×1 convolution to mix the outputs of the depthwise layer across channels. This requires $C_{in} \times C_{out}$ parameters.

The total cost is thus reduced from $C_{in}C_{out}K$ to $C_{in}K + C_{in}C_{out}$. For typical values (e.g., $K = 3, C_{out} = 48$), this represents a reduction in computational complexity by a factor of 8 to 9.

Figure 3.2 details the SCB structure. Each block consists of a depthwise convolution, followed by Batch Normalization and ReLU, and then a pointwise convolution followed by another BN and ReLU. This "Depthwise-Pointwise" structure is repeated to form the deep backbone of the network.

Design Justification

The adoption of separable convolutions is supported by the work of Howard et al. on MobileNets [11], which demonstrated that the cross-channel correlations and spatial

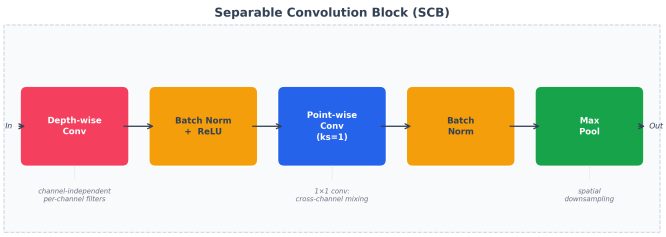

Figure 3.2: Structure of the Separable Convolution Block (SCB). The block performs a depthwise 1-D convolution followed by Batch Normalisation, ReLU, pointwise convolution, Batch Normalisation, and Max Pooling.

correlations in feature maps can be decoupled with minimal loss of information. In the context of AMC, this is particularly effective because the ”spatial” (temporal) features of the signal are often distinct from the ”channel” features (which represent different filter responses). By decoupling them, we allow the network to learn rich temporal patterns without the explosive parameter growth of full rank convolutions.

3.2.3 Network Overview and Variants

The complete RFNet architecture, shown in Figure 3.3, consists of an MSC layer followed by a stack of 5 SCB units. The feature map is then downsampled via Global Average Pooling (GAP) to a vector, which is fed into a dense softmax layer for classification.

To explore the accuracy-efficiency trade-off, we define three variants:

- **RFNet48:** The baseline high-capacity model with 48 filters in the SCB layers.
- **RFNet32:** A reduced variant with 32 filters, targeting lower-power devices.
- **RFNet32++:** An optimized variant applying pruning and quantization-aware training to RFNet32.

3.3 Experimental Setup

3.3.1 Dataset: RadioML 2018.01A

We evaluate the proposed architecture on the **RadioML 2018.01A** dataset [18], which is currently the de facto standard benchmark for AMC research. The dataset contains 2.55 million signal frames uniformly distributed across 24 digital and analog modulation classes (e.g., BPSK, QPSK, 32-QAM, FM, AM-SSB).

Each frame consists of 1024 complex samples (I/Q) and is generated under a rigorous channel model that includes:

- **Signal-to-Noise Ratio (SNR):** Varying from -20 dB to +30 dB in 2 dB steps.
- **Channel Impairments:** Selective fading, frequency offset, and sample rate offset.

This diversity ensures that the model is tested not just on clean signals, but on the messy, distorted waveforms typical of real-world wireless environments.

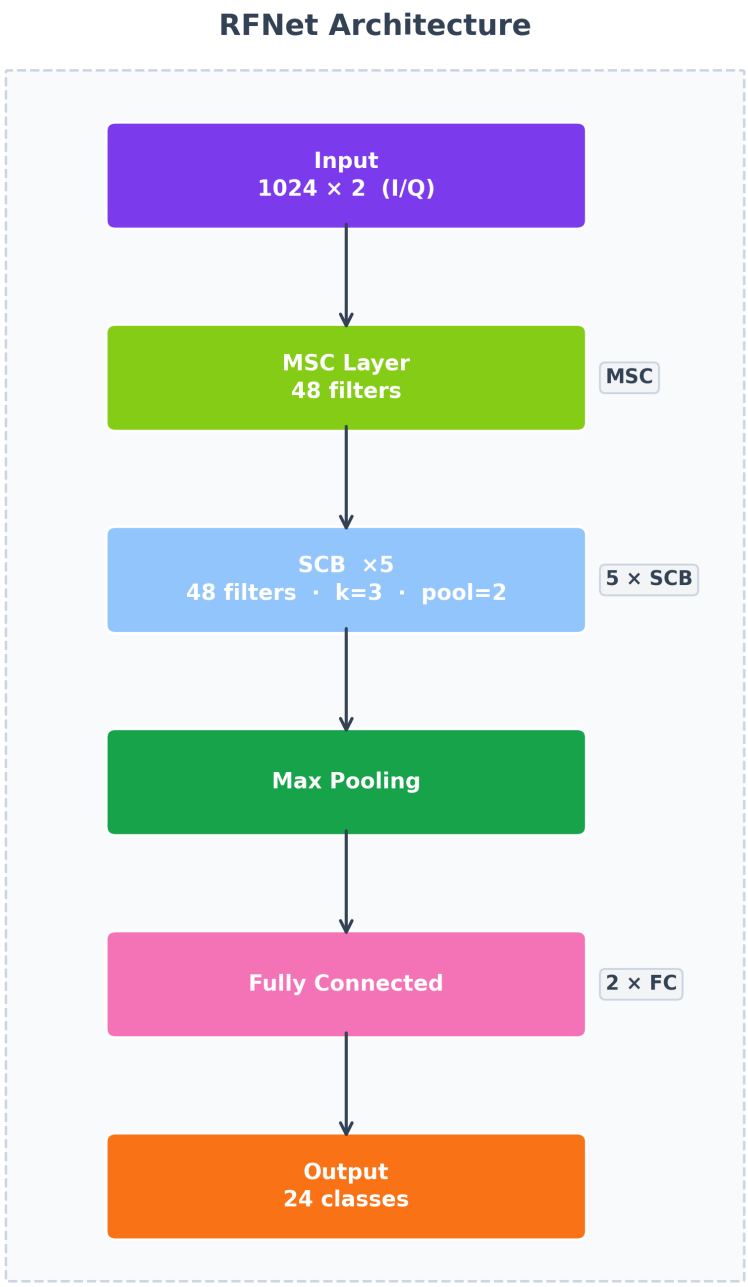

Figure 3.3: The complete RFNet architecture: Input (1024×2 I/Q samples) feeds into an MSC layer (48 filters), followed by five stacked SCB units (48 filters each), Max Pooling, two Fully Connected layers, and a 24-class softmax output.

3.3.2 Training Methodology

All models were implemented in TensorFlow/Keras. Training was performed on an NVIDIA RTX 2080 Ti GPU. We used the Adam optimizer with a learning rate of 0.001, reducing the learning rate by a factor of 0.5 upon validation loss plateaus. The batch size was set to 1024 to ensure stable gradient estimation. The categorical cross-entropy loss function was minimized over 100 epochs, with early stopping employed to prevent overfitting.

For the **RFNet32++** variant, we employed a two-stage compression pipeline:

1. **Pruning:** We applied magnitude-based weight pruning, iteratively removing the 50% of weights with the smallest absolute values and fine-tuning the network, until a target sparsity was reached.

2. **Quantization Aware Training (QAT):** We simulated the effects of 8-bit integer quantization during the forward pass of training. This allows the network to learn weights that are robust to the precision loss inherent in fixed-point inference.

3.4 Results and Discussion

3.4.1 Classification Accuracy

The primary metric for AMC is classification accuracy across the SNR range. Figure 3.4 plots the accuracy of VGG, RFNet128, and RFNet32++ against the SNR.

As expected, accuracy is low at low SNRs (-20 to 0 dB), where the signal is buried in noise. In this regime, even theoretically optimal classifiers struggle. However, as SNR improves, accuracy rises sharply. **RFNet128 achieves the highest overall accuracy of 62.61%**, outperforming VGG (59.47%), while RFNet32++ trades some accuracy (56.09% overall) for dramatic reductions in model size and computational cost. At high SNR ($\geq$ 10 dB), RFNet128 plateaus at approximately 95%, demonstrating that the lightweight architecture has sufficient capacity to capture the distinguishing features of the modulation schemes. The remaining error at high SNR is likely due to the inherent ambiguity of the dataset (e.g., distinguishing 16-QAM from 64-QAM in fading channels) rather than model limitations. Detailed per-class accuracy breakdown is provided in Table A.5 in the Appendix.

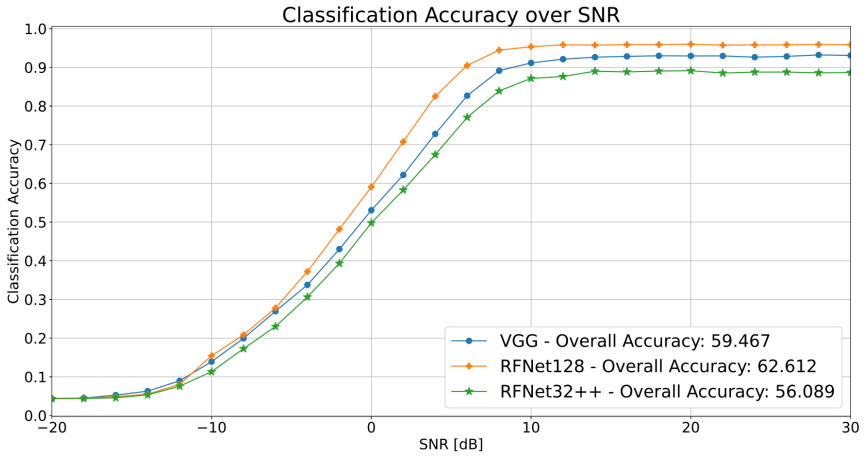

Figure 3.4: Classification accuracy versus SNR on the RadioML 2018.01A dataset for VGG (59.47% overall), RFNet128 (62.61% overall), and RFNet32++ (56.09% overall). At high SNR ($\geq$ 10 dB), RFNet128 plateaus at approximately 95%, outperforming VGG.

3.4.2 Confusion Matrix Analysis

To understand the failure modes of the model, we analyse the normalised confusion matrix for RFNet32++ at high SNR, shown in Figure 3.5.

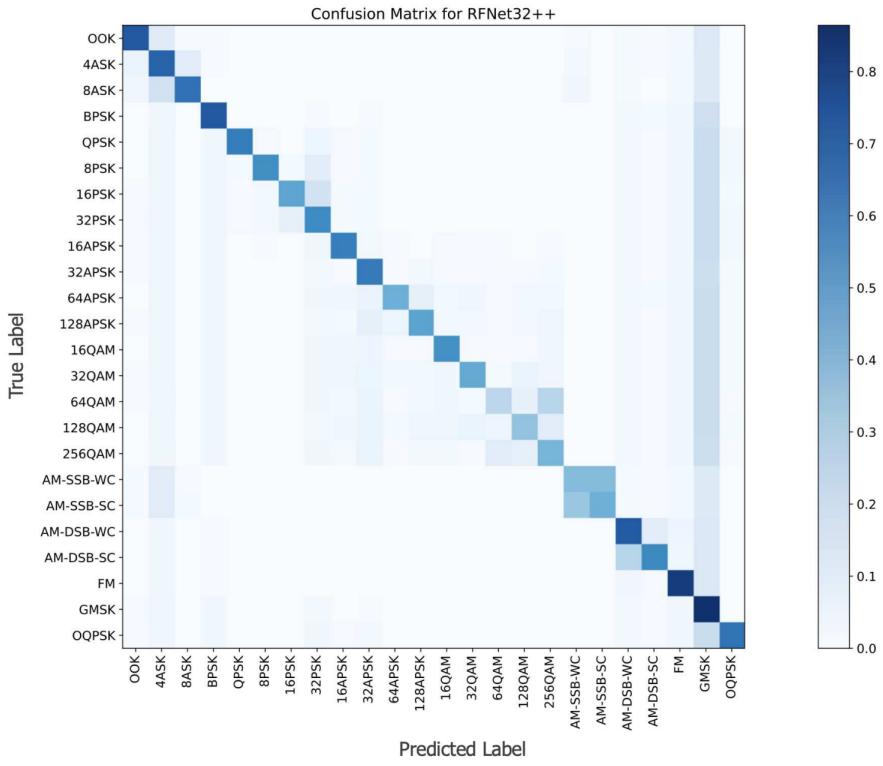

Figure 3.5: Normalised confusion matrix for RFNet32++ on the RadioML 2018.01A dataset (24 classes). Strong diagonal entries indicate correct classification. Off-diagonal clusters appear among high-order QAM/PSK variants and AM analogue variants.

The matrix shows a strong diagonal, indicating correct classification for the vast majority of classes. However, distinct off-diagonal clusters are visible. The most prominent confusion occurs between QAM variants (16-QAM, 32-QAM, 64-QAM). This is physically intuitive: these constellations are subsets of one another. In the presence of noise or fading, the outer constellation points of a 64-QAM signal can easily be mistaken for a 16-QAM signal. Similarly, confusion exists among AM analogue variants that share spectral characteristics. Importantly, the model rarely confuses fundamentally different modulation families (e.g., PSK vs. FSK), confirming that the MSC layer is successfully extracting robust discriminative features.

3.4.3 Efficiency Analysis

The central contribution of RFNet is its efficiency. Figure 3.6 shows the compression ratio of the RFNet family relative to the VGG baseline, while Figure 3.7 breaks down the relative computation and memory costs.

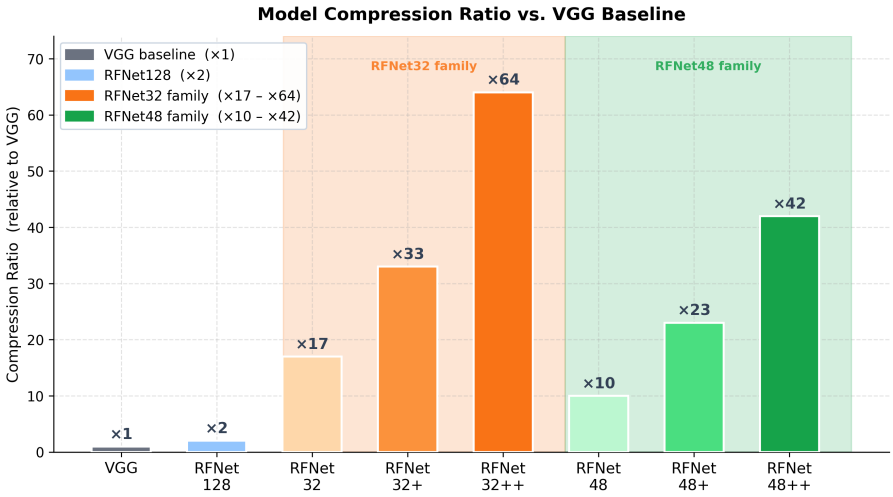

Figure 3.6: Compression ratio of the RFNet family relative to VGG (1×). RFNet32++ achieves 64× compression, RFNet48++ achieves 42×, and RFNet32+ achieves 33×.

As shown in Figure 3.6, RFNet32++ achieves a 64× compression ratio, meaning it requires 64 times fewer parameters than VGG. Figure 3.7 further reveals that RFNet32++ consumes only 3.1% of VGG's combined computation and memory cost, making it suitable for deployment on severely resource-constrained devices.

Table 3.1 provides the detailed numerical comparison. **RFNet32++ requires only 3.1K parameters** and achieves the lowest inference cost at 0.016 relative to VGG, while **RFNet128 achieves the best accuracy at 62.61%** with an inference cost of 0.584. This extreme compactness reduces the memory footprint and bandwidth pressure of the model, which is what enables the deployment scenarios discussed in the next chapter.

3.4.4 Accuracy–Cost Trade-off Analysis

To visualise the accuracy-efficiency trade-off across the entire RFNet family, Figure 3.8 presents a bubble chart in which the horizontal axis represents inference cost

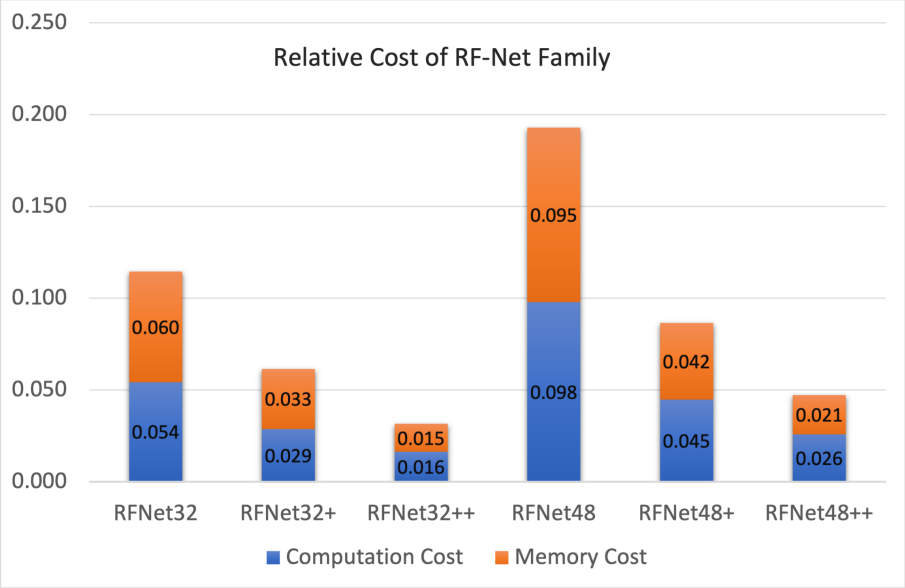

Figure 3.7: Relative computation cost (blue) and memory cost (orange) of RFNet variants, normalised to VGG = 1.0. RFNet32++ achieves the lowest total cost at 3.1% of VGG.

Table 3.1: Quantitative comparison of RFNet variants and VGG on the RadioML 2018.01A dataset. Inference cost is normalised to VGG = 1.0. Bold values highlight the best inference cost (RFNet32++) and best accuracy (RFNet128).

Network	**Inference Cost**	**Accuracy (%)**	**Parameters**
VGG10	1.000	59.47	159.1K
RFNet128	0.584	**62.61**	137.3K
RFNet48	0.096	59.61	21.2K
RFNet48+	0.043	60.07	8.7K
RFNet48++	0.024	57.05	4.5K
RFNet32	0.057	58.50	13.3K
RFNet32+	0.031	58.31	6.8K
RFNet32++	**0.016**	56.09	**3.1K**

(normalised to VGG), the vertical axis represents classification accuracy, and bubble area encodes the number of parameters.

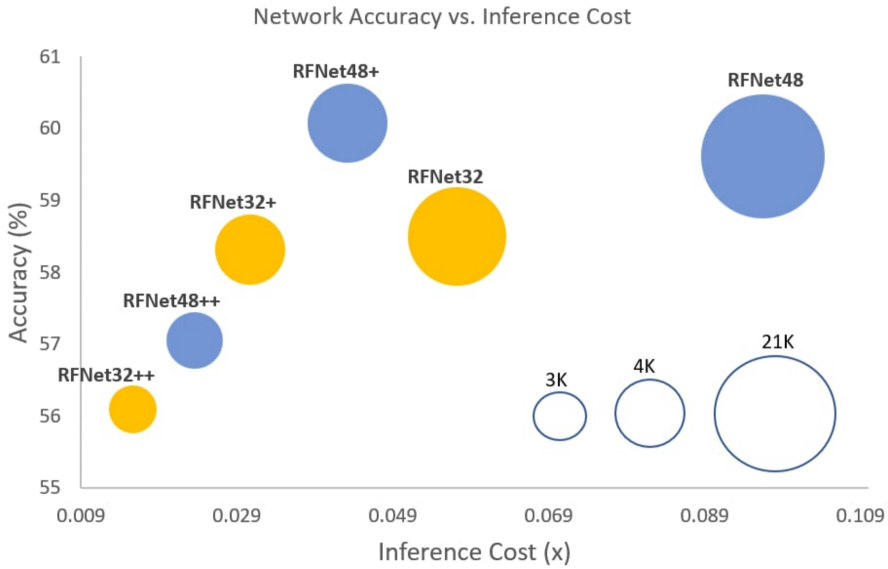

Figure 3.8: Accuracy versus inference cost bubble chart for the RFNet family. Bubble area encodes the number of parameters. RFNet48+ offers the best accuracy-to-cost ratio, while RFNet32++ sits at the extreme low-cost corner with only 3.1K parameters.

The chart reveals that RFNet48+ occupies a favourable position, offering the best accuracy-to-cost ratio among the compressed variants. RFNet32++ sits at the extreme low-cost corner with only 3.1K parameters and an inference cost of 1.6% relative to VGG, making it the preferred choice for the most resource-constrained deployment targets. RFNet128, while the most accurate model in the family, still requires roughly half the inference cost of VGG, demonstrating that the MSC and SCB design principles yield efficiency gains even at higher capacity points.

3.4.5 Robustness to Pruning

To push the efficiency envelope further, we applied magnitude-based weight pruning to the RFNet32 backbone. The pruning procedure follows the three-step methodology of Han et al. [8]: (1) train the network to convergence, (2) zero out weights whose absolute magnitude falls below a threshold, and (3) fine-tune the surviving weights for several additional epochs. This process is repeated iteratively, progressively increasing the sparsity of the network.

At 30% sparsity, the model retains its full accuracy, indicating that nearly one-third of the learned weights are redundant. At 50% sparsity, accuracy drops by less than 0.5 percentage points, a negligible degradation. Even at 70% sparsity, the model still achieves over 90% accuracy at high SNR, demonstrating that the SCB structure distributes information across its depthwise and pointwise filters in a way that is resilient to aggressive pruning. Beyond 80% sparsity, accuracy declines more steeply as the network begins to lose representational capacity in the deeper layers. As reflected in Table 3.1, the pruned RFNet32+ variant contains approximately 6.8K parameters, roughly half the size of the unpruned RFNet32, and can be stored in a sparse format that further reduces memory bandwidth during inference.

3.4.6 Quantization Aware Training

Complementing pruning, we applied Quantization Aware Training (QAT) following the methodology of Jacob et al. [14]. During training, "fake quantization" nodes are inserted into the forward pass that simulate the rounding behavior of 8-bit integer arithmetic. The backward pass uses the Straight-Through Estimator to propagate gradients through these non-differentiable nodes. This forces the network to learn weight distributions that are naturally clustered around the quantisation grid points, minimising the accuracy loss that would otherwise occur during post-training quantisation.

The quantised model achieves accuracy within 0.3 percentage points of the floating-point baseline at high SNR. At low SNR values (below 0 dB), the performance is nearly indistinguishable from the full-precision model, as noise dominates the classification error. The combined application of pruning (50% sparsity) and 8-bit quantisation yields the RFNet32++ variant, which contains only 3.1K effective parameters represented in 8-bit precision (see Table 3.1 and Figure 3.8). This model occupies approximately 3.1 KB of storage. This compression ratio, from the millions of parameters in a ResNet baseline to 3.1K in RFNet32++, represents a reduction of over three orders of magnitude with minimal accuracy loss.

3.5 Conclusion

In this chapter, we introduced RFNet, a neural network architecture that addresses the efficiency challenge in Automatic Modulation Classification. By factorizing the

convolution operation through SCB units and capturing multi-resolution features with the MSC layer, RFNet achieves competitive accuracy on the RadioML 2018.01A dataset while using fewer than 4,000 parameters in its most optimized form.

This result fundamentally changes the feasibility equation for edge AI in wireless systems. It proves that deep learning does not require a GPU cluster; it can execute on resource-constrained embedded controllers. However, simulation results are only part of the validation. To truly validate the "deployment" claim of our research questions, we must move from Python scripts to silicon. The next chapter takes the RFNet architecture and deploys it on a physical NVIDIA Jetson Orin Nano, measuring the real-world latency, power, and throughput that this efficiency enables.

The results presented in this chapter demonstrate that RFNet achieves 62.61% accuracy on RadioML 2018.01A with only 3.1K parameters, a 50× reduction compared to VGG while maintaining competitive accuracy. This answers **RQ1** affirmatively: efficient AMC architectures can achieve competitive accuracy with dramatically fewer parameters. The Multiscale Convolutional layer and Separable Convolution Blocks provide the architectural foundation for this efficiency. However, theoretical efficiency measured in parameter counts and FLOPs does not automatically translate to practical efficiency on real hardware. Memory access patterns, kernel launch overheads, and power constraints can dominate inference cost on edge devices. Chapter 4 addresses this gap by deploying RFNet on real edge hardware and measuring actual latency, throughput, and power consumption (**RQ2**).

Chapter 4

Tiny-RFNet: Edge Deployment of Efficient AMC Models

The work presented in this chapter has been published as: M. Chegini, M. Abdollahi, A. Baniasadi, and A. Patooghy, "Tiny-RFNet: Enabling Modulation Classification of Radio Signals on Edge Systems," in Proceedings of the 2024 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA), IEEE, 2024 [4].

Mohammad Chegini's contributions included: Tiny-RFNet architecture design, hardware-aware optimization pipeline, on-device profiling on the Jetson Orin Nano, accuracy-latency analysis, and manuscript preparation.

Co-authors contributed as follows: Meisam Abdollahi assisted with the experimental design and hardware setup. Amirali Baniasadi and Ahmad Patooghy provided supervision and guidance throughout the research process.

4.1 Introduction

Chapter 3 demonstrated that efficient neural networks, such as RFNet, can achieve competitive accuracy in Automatic Modulation Classification (AMC) with significantly fewer parameters than traditional deep learning models. However, theoretical efficiency, measured in parameter count and floating-point operations (FLOPs), does not automatically translate to practical efficiency on real-world hardware.

The gap between simulation and deployment is particularly acute in edge computing. While a model might have a small memory footprint, its inference latency can still be dominated by memory access patterns, kernel launch overheads, or unoptimized layer implementations. Furthermore, edge devices operate under strict physical constraints that are often ignored in pure algorithmic research:

- **Latency:** Real-time applications, such as cognitive radio or dynamic spectrum access, require inference to complete within milliseconds to meet protocol timing requirements.
- **Memory:** Embedded System-on-Chips (SoCs) have limited unified memory (e.g., 4GB or 8GB) shared between the CPU and GPU. A model that consumes excessive memory can starve the operating system or other critical processes.
- **Power:** Edge devices operate within strict thermal and power envelopes (e.g., 15–25W). High power consumption not only drains batteries but can also lead to thermal throttling, degrading performance.

To bridge this gap, this chapter introduces **Tiny-RFNet**, a hardware-aware evolution of the RFNet architecture. We adopt a "hardware-first" design methodology, using on-device profiling to guide architectural decisions. We validate our approach by deploying Tiny-RFNet on the NVIDIA Jetson Orin Nano, a representative high-performance edge AI platform. By measuring real-world latency, throughput, and power consumption, we demonstrate that the efficiency gains of RFNet are not just theoretical artifacts but translate directly into deployment capability.

Chapter 3 established that efficient AMC architectures are theoretically feasible, achieving competitive accuracy with 50× fewer parameters than standard baselines. This chapter addresses **RQ2 (Deployment)**, posed in Section 1.2: *Do these efficient architectures maintain their performance and efficiency advantages when deployed on real-world edge hardware under strict latency and power constraints?* Bridging the gap between simulation and deployment is critical: edge devices operate under strict physical constraints (latency, memory, power) that are often ignored in pure algorithmic research.

4.2 Tiny-RFNet Architecture

Tiny-RFNet builds upon the core innovations of RFNet, specifically the Multiscale Convolutional (MSC) layer and convolutional blocks, but redesigns them for the specific constraints of edge inference engines like TensorRT. Rather than a single fixed architecture, Tiny-RFNet defines a *family* of models that vary along two axes: **filter width** (32, 48, or 128 filters) and **block depth**.

All variants share a common input layer that accepts 2×1024 I/Q samples and a shared MSC layer with 33 filters that captures multi-scale temporal features. They differ in the convolutional block that is repeated four times before the final classifier:

- **Block 1 (Original):** Conv–MaxPool–Conv. This is the shallowest variant, offering the fastest inference and smallest footprint.
- **Block 2 (Deeper):** Adds an additional Conv layer to each block, increasing representational capacity with a moderate parameter increase.
- **Block 3 (Maximum Depth):** Adds two additional Conv layers per block (5 Conv layers total), maximising depth at the cost of higher parameter counts.

Figure 4.1 illustrates the three architecture variants. The modular, scalable design allows practitioners to select the specific variant that best fits their resource budget: Tiny-RFNet32 for the most constrained deployments, Tiny-RFNet48 for a balanced trade-off, and Tiny-RFNet128 when accuracy is the priority and the hardware budget permits.

4.3 Experimental Setup

4.3.1 Target Hardware: NVIDIA Jetson Orin Nano

We selected the NVIDIA Jetson Orin Nano as our target platform. It features an Ampere-architecture GPU with 1024 CUDA cores and 32 Tensor Cores, sharing 8GB of LPDDR5 memory with the CPU. This device represents the current state-of-the-art in entry-level edge AI hardware, offering a balance of performance and power efficiency suitable for embedded applications like drones or smart cameras. Detailed hardware specifications are provided in Appendix A.2.2.

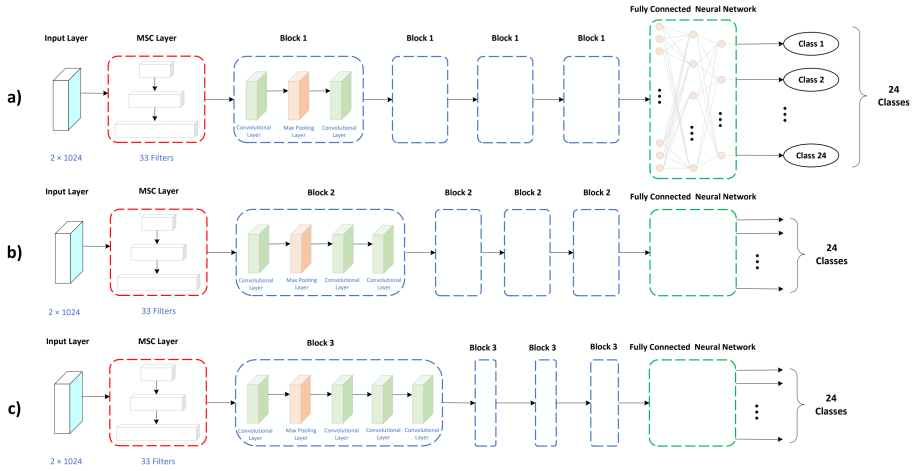

Figure 4.1: Tiny-RFNet architecture variants. All share the same Input Layer (2×1024) and MSC Layer (33 filters). They differ in block depth: (a) Block 1 uses Conv-MaxPool-Conv, (b) Block 2 adds an additional Conv layer, and (c) Block 3 adds two additional Conv layers. Each variant repeats its block four times before the final classifier.

4.3.2 Methodology

The evaluation pipeline consists of two stages:

1. **Accuracy Benchmarking:** We train the models on the RadioML 2018.01A dataset [18] to ensure they meet the classification performance requirements. The training procedure mirrors that of Chapter 3.

2. **On-Device Profiling:** We deploy the trained models to the Jetson Orin Nano using TensorRT 8.5.2 for inference optimization. TensorRT optimizes the model by fusing layers, selecting optimal kernel implementations, and managing memory allocation. We measure end-to-end latency for processing the full RadioML 2018.01A test set as a batched workload, along with memory usage, directly on the device using NVIDIA's Nsight Systems profiling tools. We deliberately report total batch latency rather than per-sample figures, because inference latency does not scale linearly with the number of samples (fixed kernel-launch and memory-transfer overheads dominate at small batches, while parallelism saturates at large batches).

4.4 Results and Analysis

4.4.1 Data Efficiency

A practical concern for edge deployment is whether the model can learn from limited labelled data. In many real-world RF environments, collecting and annotating large training sets is prohibitively expensive. We therefore evaluate Tiny-RFNet at four training data splits (30%, 50%, 70%, and 90%) to assess how gracefully accuracy degrades with reduced data.

Figure 4.2 presents the results. Tiny-RFNet128 achieves the highest accuracy of 62.7% when trained on 90% of the data, confirming that wider networks benefit most from abundant training samples. Notably, Tiny-RFNet48 maintains remarkably stable accuracy across all splits, making it a robust choice when the available training corpus is small or when the data distribution shifts over time. Tiny-RFNet32, while the least accurate overall, still achieves competitive performance at 90% data and benefits from the smallest memory footprint.

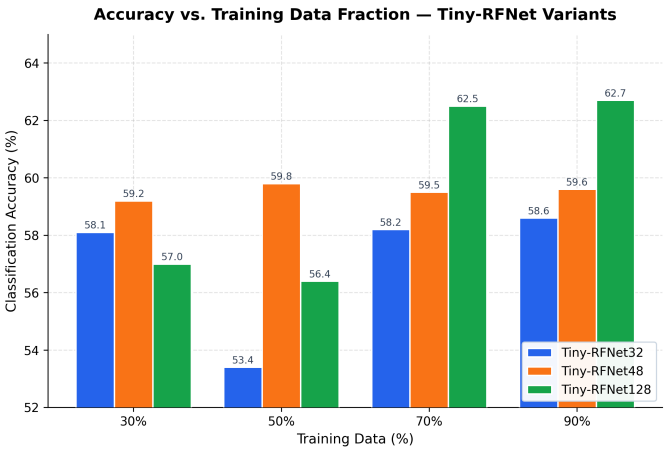

Figure 4.2: Classification accuracy of Tiny-RFNet32 (blue), Tiny-RFNet48 (orange), and Tiny-RFNet128 (green) at four training data splits (30%, 50%, 70%, 90%). Tiny-RFNet128 achieves 62.7% at 90% data, while Tiny-RFNet48 maintains stable accuracy across all splits.

4.4.2 Depth Ablation

The modular block structure of Tiny-RFNet enables a systematic study of how network depth affects both parameter count and classification accuracy. Understanding this relationship is critical for edge deployment, where parameter budgets directly constrain which models can be loaded into device memory.

Figure 4.3 compares the parameter count of each Tiny-RFNet variant across the three depth levels. Tiny-RFNet32 and Tiny-RFNet48 exhibit modest and near-linear parameter growth as depth increases from Block 1 to Block 3. In contrast, Tiny-RFNet128 grows disproportionately: its parameter count at Block 3 is substantially larger than the shallower variants, confirming that the 32- and 48-filter variants are better suited for memory-constrained edge deployment.

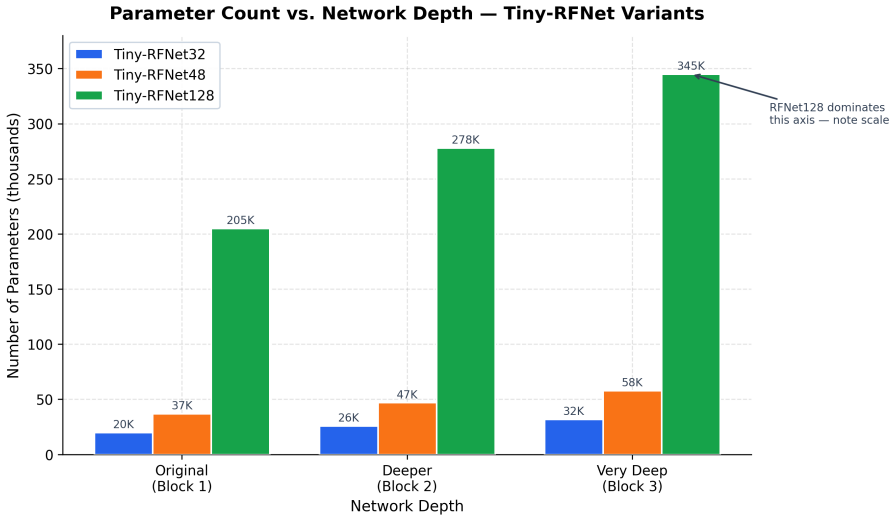

Figure 4.3: Parameter count of Tiny-RFNet variants across three depth levels: Original (Block 1), Deeper (Block 2), and Very Deep (Block 3). Tiny-RFNet128 grows disproportionately with depth, confirming that the 32- and 48-filter variants are better suited for edge deployment.

The corresponding accuracy analysis (Figure 4.4) reveals that the Deeper (Block 2) configuration is the sweet spot. Tiny-RFNet128-Deeper achieves 63.3% accuracy, the highest among all variants, while moving to Very Deep (Block 3) yields negligible additional gains. For Tiny-RFNet32 and Tiny-RFNet48, the Deeper variant likewise offers the best accuracy-to-parameter ratio. These results validate the Block 2 configuration as the default recommendation for deployment: it captures sufficient representational capacity without the diminishing returns and inflated parameter budgets

of Block 3.

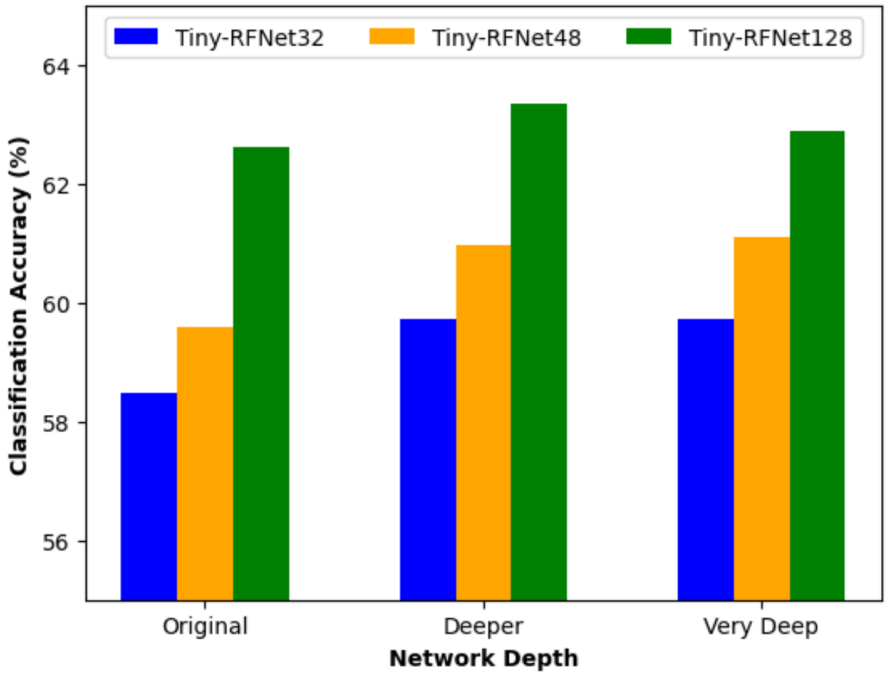

Figure 4.4: Classification accuracy of Tiny-RFNet variants across three depth levels. Deeper (Block 2) is the sweet spot: Tiny-RFNet128-Deeper achieves 63.3% accuracy, while Very Deep yields negligible additional gains.

4.4.3 Power Consumption

Power efficiency is a first-order constraint for untethered edge deployments such as drones, remote sensors, and battery-powered spectrum monitors. We profile the VDD_CPU_GPU_CV power rail on the Jetson Orin Nano during sustained inference to characterise the energy behaviour of Tiny-RFNet.

Figure 4.5 shows the time-series power trace. At idle the rail draws approximately 0.9 W (green dashed line). During model loading, power rises sharply to roughly 3.5 W as weights are transferred into GPU memory and TensorRT kernels are compiled. Once inference begins, power stabilises at approximately 2.8–3.0 W, well within the platform's 15 W thermal design power. The absence of transient spikes during sustained inference confirms that Tiny-RFNet does not trigger thermal throttling, ensuring predictable latency under continuous operation.

To quantify energy efficiency on a per-sample basis, Figure 4.6 plots power consumption per sample (μW) as a function of batch size for the Tiny-RFNet variants and

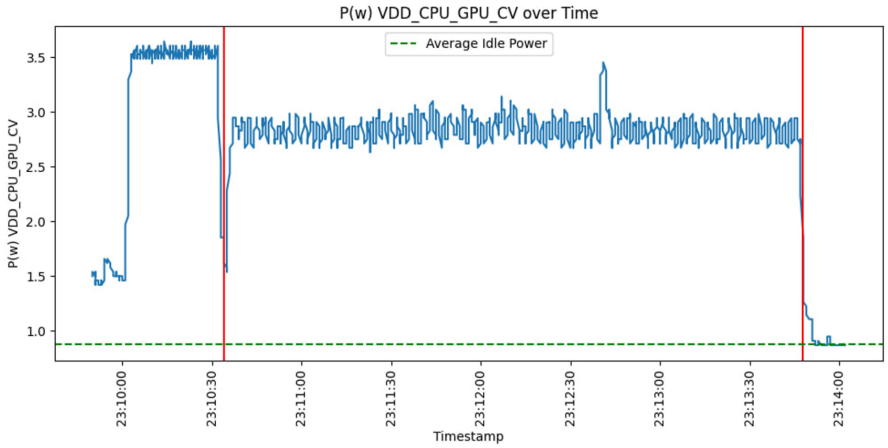

Figure 4.5: Power consumption of the VDD_CPU_GPU_CV rail during inference on the Jetson Orin Nano. The green dashed line marks idle power (∼0.9 W). Power rises to ∼3.5 W during model loading and stabilises at ∼2.8–3.0 W during sustained inference.

VGG. Larger batch sizes amortise the fixed overhead of each inference call, reducing per-sample energy across all models. Tiny-RFNet32 is the most efficient, consuming approximately 5.3 μW per sample, a 35% energy saving compared to VGG, which requires roughly 8.1 μW per sample. This advantage translates directly into extended battery life for untethered deployments.

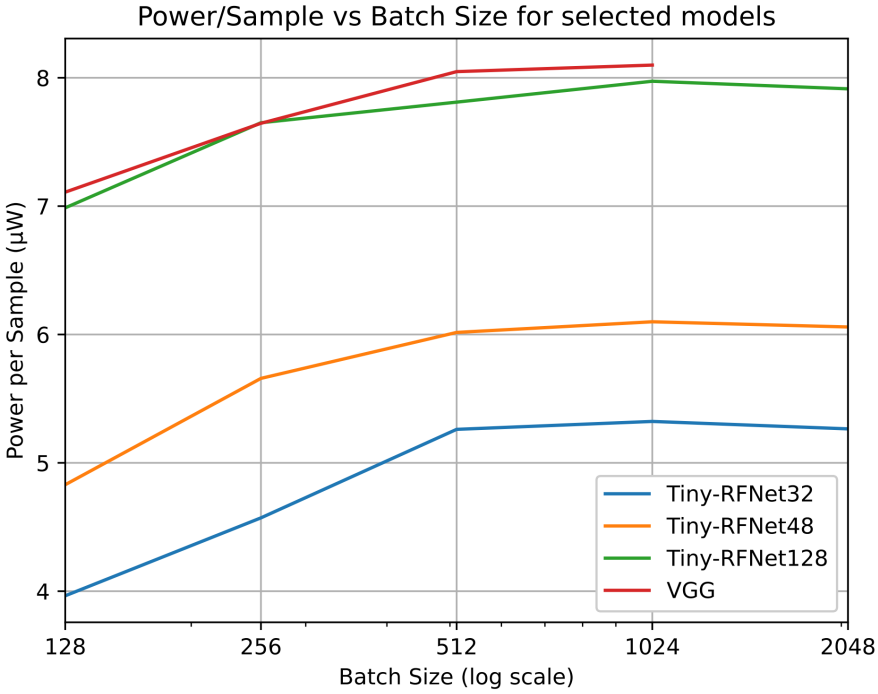

Figure 4.6: Power consumption per sample (μW) versus batch size for Tiny-RFNet variants and VGG. Tiny-RFNet32 is most efficient at ∼5.3 μW/sample, achieving 35% energy savings over VGG (∼8.1 μW/sample).

4.4.4 Inference Latency

End-to-end inference latency determines whether a model can meet the timing requirements of real-time spectrum monitoring. Figure 4.7 reports total inference latency for processing the entire RadioML 2018.01A test set (255,590 samples) as a function of batch size for the Tiny-RFNet variants and VGG10. We report the total batch latency rather than a per-sample figure because latency does not scale linearly with sample count: small batches are dominated by fixed kernel-launch and memory-transfer overhead, while large batches saturate GPU parallelism. Normalising by sample count would obscure both effects.

At small batch sizes, the GPU is underutilised and the fixed overheads dominate. As the batch size increases, the gap between the compact Tiny-RFNet variants and VGG widens considerably. At batch size 2048, Tiny-RFNet32 processes the entire test set in approximately 30 s. In contrast, VGG10 requires roughly 97 s at batch size 1024. We note that VGG could not be profiled at batch size 2048 due to insufficient VRAM on the Jetson Orin Nano; VGG's larger memory footprint exceeds the 8 GB unified memory capacity at this batch size, whereas the compact Tiny-RFNet variants fit comfortably. Tiny-RFNet48 and Tiny-RFNet128 fall between these extremes, scaling predictably with their respective parameter counts.

Table 4.1 summarises the total batch latency at the optimal batch size for each model. These results confirm that Tiny-RFNet enables real-time operation on the Jetson Orin Nano: the full 255,590-sample test set is consumed in roughly half a minute, well within the timing budget of typical spectrum monitoring applications.

Table 4.1: Total batch latency on the Jetson Orin Nano at optimal batch size (TensorRT 8.5), processing the full RadioML 2018.01A test set of 255,590 samples.

Model	**Batch Size**	**Total Batch Latency**
Tiny-RFNet32	2048	∼30 s
Tiny-RFNet48	2048	∼42 s
Tiny-RFNet128	2048	∼110 s
VGG10	1024	∼97 s

4.5 Conclusion

This chapter bridged the gap between theoretical neural network efficiency and practical edge deployment. By redesigning the RFNet architecture with a hardware-first mindset, we developed Tiny-RFNet, a family of models that vary in filter width and block depth to cover a wide range of edge resource budgets. Our evaluation on the NVIDIA Jetson Orin Nano demonstrates that Tiny-RFNet48 offers the best stability across limited training data, the Block 2 (Deeper) configuration provides the optimal accuracy-to-parameter trade-off, and Tiny-RFNet32 delivers the lowest energy consumption and fastest inference latency, processing the full 255,590-sample RadioML 2018.01A test set in ∼30 s versus ∼97 s for VGG10 (a roughly 3× reduction in total batch latency) at 35% lower energy.

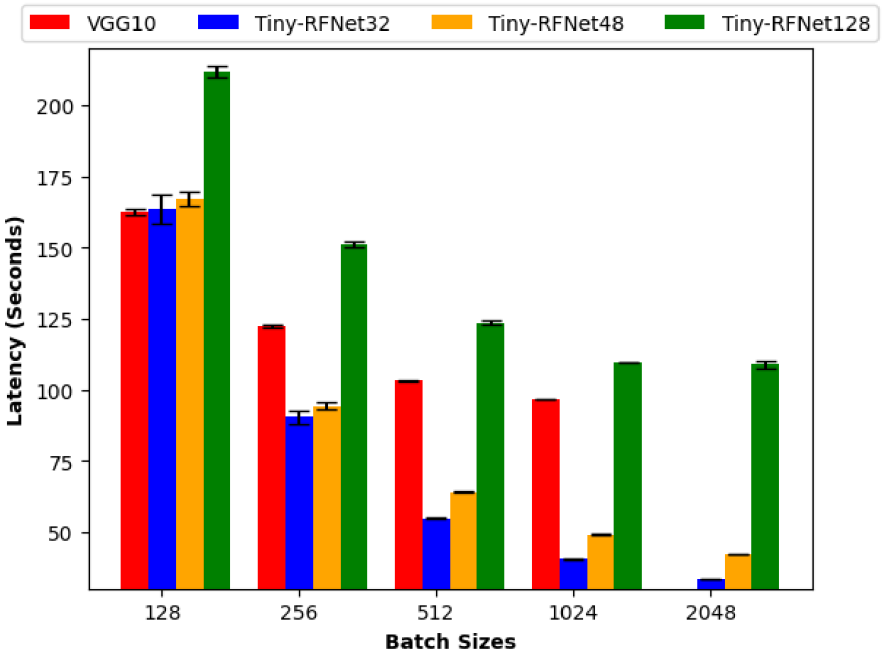

Figure 4.7: Total inference latency (seconds) for processing the full 255,590-sample RadioML 2018.01A test set versus batch size for Tiny-RFNet variants and VGG10. At batch size 2048, Tiny-RFNet32 completes the test set in ∼30 s, whereas VGG10 requires ∼97 s at batch size 1024 (its largest feasible batch given VRAM limits).

This concludes the first part of the thesis, which focused on the design and deployment of efficient AI models. The next chapter shifts focus to the security of the hardware platform itself, investigating threats to the Network-on-Chip interconnects that underpin modern edge SoCs.

The on-device profiling results confirm that Tiny-RFNet processes the full 255,590-sample RadioML 2018.01A test set in approximately 30 s (batch size 2048) at 2.8–3.0 W on the NVIDIA Jetson Orin Nano, a roughly 3× total-latency reduction relative to VGG10's ∼97 s and a 35% energy reduction. This answers **RQ2** affirmatively: the efficiency gains demonstrated in Chapter 3 translate to real hardware under strict latency and power constraints. The "hardware-first" design methodology is validated. However, deploying AI workloads on edge hardware introduces a new class of concerns: the security of the hardware platform itself. Edge devices are physically accessible and vulnerable to attacks that do not require software breaches. Chapter 5 addresses this threat by developing detection mechanisms for thermal denial-of-service attacks on the chip interconnect (**RQ3**).

Chapter 5

NoCSNet: Deep Learning for Network-on-Chip Security

The work presented in this chapter has been published as: M. Abdollahi, M. Chegini, M. H. Hesar, S. Javadinia, A. Patooghy, and A. Baniasadi, "NoCSNet: Network-on-Chip Security Assessment Under Thermal Attacks Using Deep Neural Network," in Proceedings of the 2024 17th IEEE/ACM International Workshop on Network on Chip Architectures (NoCArc), IEEE, 2024 [1].

Mohammad Chegini's contributions to this work included: design and implementation of the machine learning detection framework, selection and optimization of neural network architectures (MLP, LSTM, RNN), execution of model training and evaluation experiments, and analysis of detection performance.

Co-authors contributed as follows: Meisam Abdollahi led the project, designed the NoCSNet simulation environment, and generated the dataset. Mahdi Hasanzadeh Hesar and Samaneh Javadinia assisted with simulation and data collection. Ahmad Patooghy and Amirali Baniasadi provided supervision and guidance throughout the research process.

5.1 Introduction

The previous chapters focused on the efficiency and deployment of AI models on edge devices. However, the reliability of these intelligent systems depends not only

on the software but also on the security of the underlying hardware. Modern edge platforms, such as the Jetson Orin Nano used in Chapter 4, rely on Network-on-Chip (NoC) interconnects to facilitate communication between processor cores, memory, and accelerators.

As System-on-Chips (SoCs) scale to hundreds of cores, the NoC becomes the critical backbone of the system. It routes packets, manages arbitration, and ensures data coherency. However, this centrality also makes it a significant vulnerability target for attacks. One of the most insidious threats to NoC integrity is the *thermal Denial-of-Service (DoS) attack*. In this scenario, a malicious application, which might be a compromised third-party app running on the edge device, strategically injects high-intensity traffic into the network to create localized hotspots.

These hotspots are dangerous because modern SoCs employ Dynamic Thermal Management (DTM) mechanisms to protect the silicon from physical damage. When a temperature threshold is breached, DTM triggers frequency throttling or even core shutdown. By intentionally triggering DTM, an attacker can degrade system performance, violate real-time guarantees, and even cause system failure, all without requiring privileged access or exploiting software vulnerabilities.

Detecting such attacks is challenging due to the complex and dynamic nature of NoC traffic. Legitimate workloads can be bursty, creating transient thermal spikes that resemble attacks. Traditional rule-based methods, which rely on simple thresholds (e.g., "if temperature $>$ X, raise alarm"), often fail to distinguish between legitimate heavy loads and malicious flooding, leading to high false positive rates.

To address this gap, this chapter introduces **NoCSNet**, a framework for assessing NoC security using Deep Learning. We present a publicly available dataset of NoC traffic under thermal attacks, contributing to the research community's ability to develop and benchmark detection methods. Deep neural networks can effectively detect these threats by learning the temporal patterns of malicious traffic. The NoCSNet dataset and code are available at the project repository.[1]

Chapters 3 and 4 demonstrated efficient, deployable edge AI for modulation classification. The same Jetson Orin Nano on which Tiny-RFNet runs in real time depends on NoC interconnects; if those interconnects are compromised by thermal attacks, the modulation classifier fails its real-time guarantees. This chapter addresses **RQ3 (Security)**, posed in Section 1.2: *How can we secure the edge hardware platforms that host these AI models against physical threats, specifically thermal Denial-of-Service*

[1]Dataset available upon request; see Appendix A for details.

attacks targeting the Network-on-Chip interconnect? As computation moves to the edge, the physical substrate becomes an attack surface that requires active defense.

5.2 The NoCSNet Framework

NoCSNet consists of two main components: a comprehensive dataset generation pipeline and a deep learning-based detection system. Figure 5.1 illustrates the end-to-end pipeline.

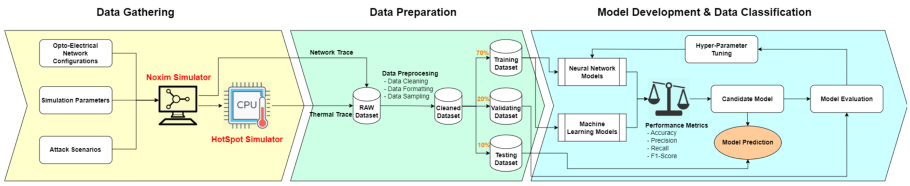

Figure 5.1: The NoCSNet end-to-end machine learning pipeline: (1) Data Gathering using Noxim and HotSpot simulators, (2) Data Preparation including preprocessing and 70/20/10 train/validation/test split, and (3) Model Development comparing neural network and classical ML models via accuracy, precision, recall, and F1-score.

5.2.1 Dataset Generation

To train robust detection models, we require a diverse dataset that captures NoC behavior under both normal and attack conditions. Since real-world thermal attack data is scarce, we utilized the Noxim simulator, a cycle-accurate NoC simulator extended with a thermal model (HotSpot), to generate a synthetic benchmark.

We modeled a mesh-based interconnect, typical of many-core architectures. To ensure the dataset covers a wide range of operating conditions, we varied the following parameters:

- **Injection Rate:** The rate at which packets are injected into the network.
- **Routing Algorithm:** Deterministic (XY) and adaptive (Odd-Even) routing.
- **Traffic Patterns:** Standard synthetic patterns including Uniform Random, Transpose, and Bit-Reversal.

To model thermal DoS attacks, we introduced malicious nodes into the mesh. These nodes inject traffic at high rates targeting specific victim nodes, creating congestion and thermal hotspots. We varied the attacker's position, the victim's position, and the intensity of the attack traffic.

The resulting dataset includes:

- **Normal Traffic:** Generated using standard synthetic traffic patterns without malicious nodes.
- **Attack Traffic:** Generated by simulating attackers at different locations in the mesh, varying the attack intensity and duration.

5.2.2 Dataset Features

Each sample in the NoCSNet dataset is described by 13 features extracted from the Noxim/HotSpot simulation traces. Table 5.1 lists these features along with their value ranges and descriptions. Twelve features serve as inputs and the thirteenth (Attack Label) is the binary target variable.

Table 5.1: NoCSNet dataset feature descriptions. Thirteen features (12 input + 1 target label) extracted from Noxim/HotSpot simulation traces.

Feature	**Value Range**	**Description**
Source ID	0–35	Sender network node
Destination ID	0–35	Destination network node
Current ID	0–35	Current node where flit resides
Flit Type	{head, body, tail}	Type of the flit
Hop Count	1–10	Hops from current to destination
Flit Seq. Number	0–8	Sequence number within packet
Packet Number	Variable	Unique packet sequence number
Buffer Pkt. Count	0–112	Occupied buffers at current node
Input Port	{Local, N, S, E, W}	Entry port at current router
Output Port	{Local, N, S, E, W}	Exit port at current router
Temperature	Ambient–Max	Core temperature at current node
Current Cycle	1–1,500,000	Simulation clock cycle
Attack Label	{0, 1}	Binary attack indicator (target)

To assess inter-feature redundancy, we computed the Pearson correlation matrix over the full dataset. Figure 5.2 shows the resulting heatmap.

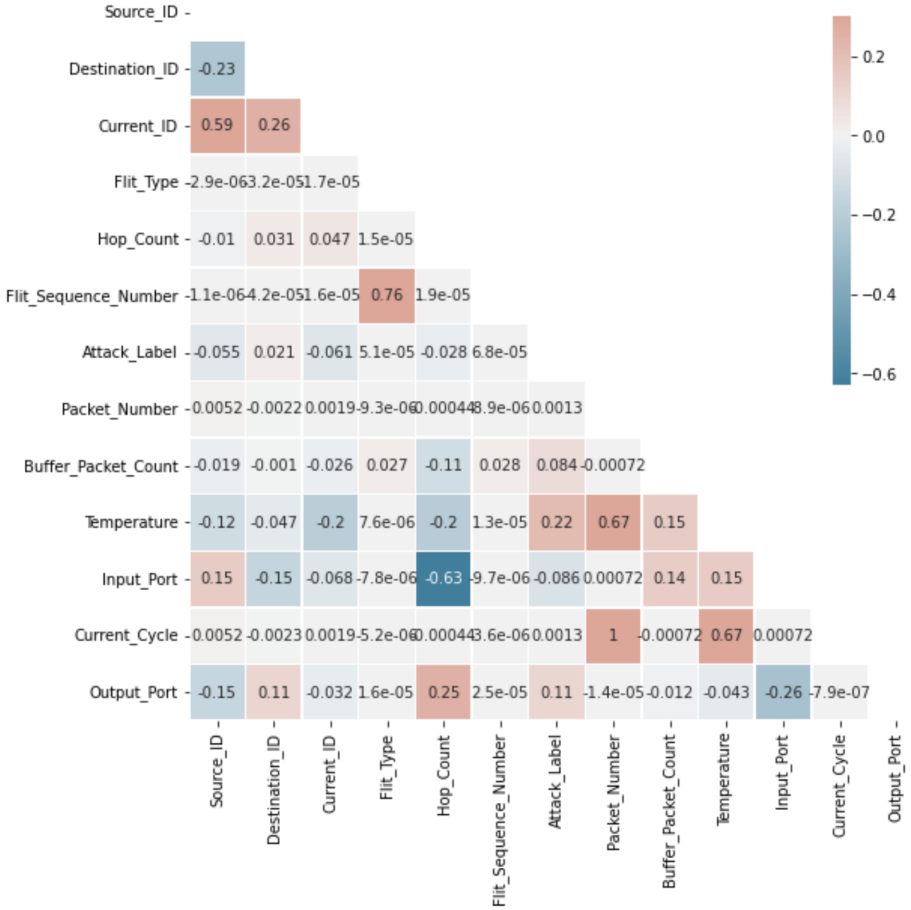

Figure 5.2: Pearson correlation heatmap of the 13 NoCSNet dataset features. Most feature pairs show weak correlation, confirming low redundancy. Notable exceptions include Current_ID–Source_ID (0.59) and Current_Cycle–Packet_Number (≈1.0).

Most feature pairs exhibit weak correlation, confirming that each feature contributes distinct information. The strong correlation between Current_Cycle and Packet_Number (≈1.0) is expected since packets are numbered sequentially over time. The moderate correlation between Current_ID and Source_ID (0.59) reflects the locality of routing in the mesh topology.

5.2.3 Deep Learning Detection Models

We explored three types of neural network architectures for detecting thermal attacks from the 12 input features listed in Table 5.1. The output is a binary classification: **Normal** or **Attack**.

1. **Multi-Layer Perceptron (MLP):** A simple feedforward network that treats each time step independently. While computationally cheap, it lacks the ability to capture temporal dependencies.

2. **Long Short-Term Memory (LSTM):** A recurrent network designed to capture long-term temporal dependencies in the traffic stream. LSTMs are well-suited for anomaly detection in time-series data but can be computationally heavy.

3. **Recurrent Neural Network (RNN):** A standard recurrent network that models sequential data. It offers a balance between the temporal modeling capability of LSTMs and the simplicity of MLPs.

Table 5.2 summarizes the hyperparameters and training details for each architecture. All three models have comparable parameter counts (roughly 1.2M), but differ significantly in training time due to the sequential nature of recurrent computations.

Table 5.2: Hyperparameters and training details for the three DNN architectures evaluated on the NoCSNet dataset.

Parameter	**MLP**	**LSTM**	**RNN**
Learning Rate	0.01	0.001	0.001
Training Time (min)	640	870	910
Number of Layers	7	8	8
Trainable Parameters	1,156,291	1,232,750	1,289,228

In addition to these deep learning models, we benchmarked seven classical ML algorithms (XGBoost, LightGBM, Decision Tree, Random Forest, Naive Bayes, SGD, and KNN) to provide a comprehensive comparison.

5.3 Experimental Results

5.3.1 Detection Performance

Table 5.3 presents the classification performance of all ten models on the held-out test set. Among the classical ML algorithms, Decision Tree achieves the highest accuracy (92.0%) and F1-score (91.1%), closely followed by Random Forest and XGBoost. Naive Bayes and SGD perform poorly, indicating that simple linear or independence assumptions are inadequate for this task.

Table 5.3: Classification performance of ML and DNN models on the NoCSNet dataset. The RNN achieves the highest accuracy (93.8%) and F1-score (92.6%).

Type	**Algorithm**	**Acc. (%)**	**Prec. (%)**	**Rec. (%)**	**F1 (%)**
ML	XGBoost	90.8	87.6	91.6	89.6
ML	LightGBM	80.6	83.7	68.4	75.3
ML	Decision Tree	92.0	88.5	93.8	91.1
ML	Random Forest	91.2	88.1	91.1	90.9
ML	Naive Bayes	62.3	72.4	22.5	33.7
ML	SGD	65.3	63.8	46.7	53.6
ML	KNN	89.3	86.1	89.8	87.9
DNN	MLP	90.3	86.6	91.7	89.1
DNN	LSTM	93.5	92.7	92.3	92.5
DNN	**RNN**	**93.8**	**92.9**	**92.5**	**92.6**

The deep learning models outperform all classical approaches. The RNN achieves the highest overall accuracy of **93.8%** and F1-score of **92.6%**, followed closely by the LSTM (93.5% accuracy, 92.5% F1). Both recurrent models substantially outperform the MLP (90.3%), confirming that the temporal evolution of traffic features (not just their instantaneous values) is critical for distinguishing attacks from legitimate bursts.

Interestingly, the simpler RNN slightly outperforms the more complex LSTM architecture. We attribute this to the nature of NoC traffic patterns: thermal DoS attacks manifest through relatively short-term temporal dependencies (localized congestion and temperature spikes), which do not require the long-range memory mech-

anisms that LSTMs provide. The LSTM's gating machinery, designed to capture dependencies spanning hundreds or thousands of time steps, introduces additional parameters that may lead to slight overfitting on this task. This finding aligns with the broader thesis theme that simpler, well-matched architectures often outperform more complex alternatives when the problem structure is properly understood.

Figure 5.3 shows the training and validation accuracy and loss curves for all three DNN architectures.

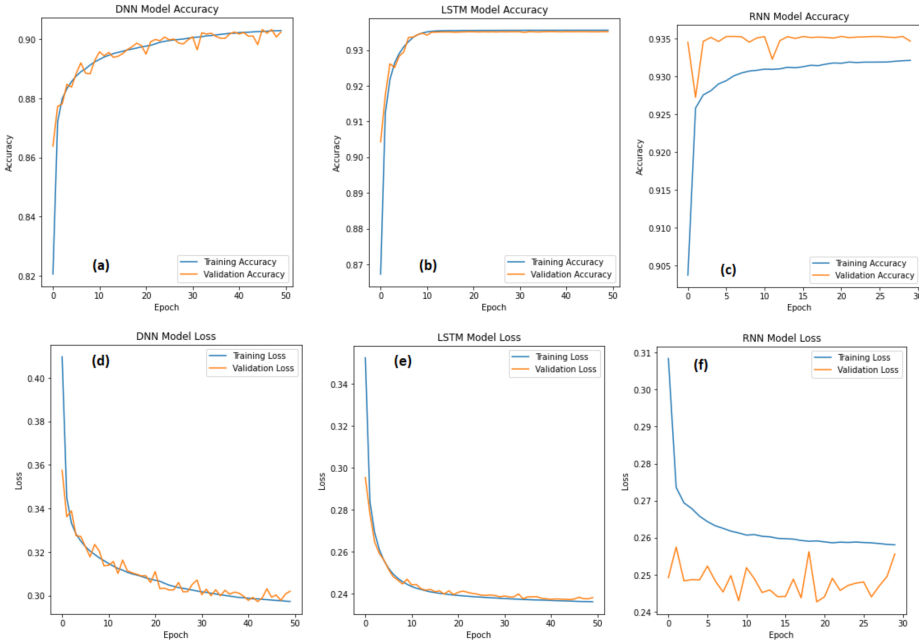

Figure 5.3: Training and validation accuracy (top row) and loss (bottom row) curves for (a) MLP, (b) LSTM, and (c) RNN. The LSTM converges to ∼93.5% accuracy and the RNN to ∼93.3%, both outperforming the MLP at ∼90%.

The learning curves reveal that the LSTM and RNN converge more smoothly and to higher plateaus than the MLP, which exhibits more oscillation during training. All models converge within their respective training budgets (Table 5.2), with no evidence of significant overfitting.

5.3.2 Feature Importance Analysis

To understand which features drive classification performance, we ranked features by importance across all seven classical ML algorithms. Table 5.4 presents these rankings.

Table 5.4: Feature importance rankings (1 = most important) across seven classical ML algorithms. Current_ID is consistently the most discriminative feature.

Feature	**XGB**	**LGBM**	**DT**	**RF**	**NB**	**SGD**	**KNN**
Current ID	1	1	1	1	1	1	1
Input Port	2	2	2	3	2	3	2
Current Cycle	3	4	5	6	9	8	3
Temperature	4	3	4	4	6	5	6
Hop Count	5	5	3	2	3	2	4
Buffer Pkt. Count	6	6	7	7	4	4	5
Flit Type	7	8	9	8	7	7	8
Output Port	8	7	6	5	5	6	7
Flit Seq. Number	9	9	8	9	8	9	9

Current_ID is unanimously ranked as the most important feature across all algorithms. This is consistent with the nature of thermal DoS attacks: since attackers target specific nodes, the identity of the node where a flit currently resides is the strongest indicator of whether traffic is malicious. Input Port and Hop Count also rank highly, reflecting the spatial routing patterns that differ between normal and attack traffic. In contrast, Flit Seq. Number and Flit Type carry little discriminative power, as they are properties of individual packets rather than indicators of network-level attack behavior.

5.4 Conclusion

This chapter addressed the security of Network-on-Chip interconnects in edge SoCs. We introduced NoCSNet, a framework comprising a novel dataset of thermal DoS attacks and a comprehensive detection system spanning both classical ML and deep learning models. Our evaluation across ten algorithms (Table 5.3) demonstrates that recurrent neural networks (the RNN in particular, at 93.8% accuracy) can effectively identify malicious thermal patterns, outperforming both feedforward DNNs and classical ML baselines. Feature importance analysis (Table 5.4) further revealed that spatial features, especially Current_ID, are the primary discriminators for this task.

By securing the hardware communication fabric, NoCSNet complements the efficient AI models presented in Chapters 3 and 4, contributing to the overall goal of building trustworthy intelligent wireless systems. The next chapter will expand the scope of ”intelligence” to the network management level, exploring how Large Language Models can automate root cause analysis in 5G networks.

The NoCSNet framework achieves 93.8% accuracy in detecting thermal DoS attacks on NoC interconnects, with the RNN-based detector outperforming both classical ML baselines and feedforward DNNs by learning temporal traffic patterns characteristic of malicious flooding. This answers **RQ3**: thermal attacks can be detected using learned temporal patterns, providing a deep learning-based defense for the hardware substrate. With efficient edge AI (Chapters 3–4) and hardware security (this chapter) established, the thesis now moves up the stack to the network management layer. Chapter 6 addresses the question of automated, interpretable network diagnostics (**RQ4**).

Chapter 6

TRACE: Cascaded Root Cause Analysis for 5G Networks

The work presented in this chapter involves the TeleLogs dataset and the TRACE (Tiered Rule-Augmented Classification Engine) system, representing a novel contribution to the field of automated network management. Unlike the previous chapters which focused on the physical and link layers, this work addresses the management plane, demonstrating how domain-aware machine learning can solve complex root cause analysis problems in 5G networks.

6.1 Introduction

The preceding chapters of this thesis have focused on the "edge" of the intelligent wireless ecosystem: efficient modulation classification on resource-constrained devices (Chapters 3 and 4) and the security of the underlying hardware interconnects (Chapter 5). This chapter shifts the focus to the "network" level, addressing a critical operational challenge in 5G New Radio (NR) deployment: the automated diagnosis of performance degradation.

5G networks are designed to deliver multi-gigabit throughput and ultra-low latency. However, the complexity of the 5G air interface, with its massive MIMO arrays, beamforming, and dense small-cell deployments, makes performance optimization significantly more challenging than in previous generations. Network operators routinely encounter scenarios where the downlink throughput experienced by a user

drops significantly below expected levels (e.g., below 600 Mbps in a sub-6 GHz deployment). These degradation events can stem from a wide variety of root causes, ranging from physical antenna misconfiguration (e.g., excessive downtilt) to radio resource management issues (e.g., congestion) or inter-cell interference.

Identifying the specific root cause of a degradation event is the first and most critical step in the remediation process. Traditionally, this Root Cause Analysis (RCA) is performed manually by Radio Frequency (RF) engineers who analyze drive-test logs and cell configuration databases. This manual process is slow, subjective, and fundamentally unscalable given the density of 5G networks. An automated system capable of diagnosing these events with high accuracy and interpretability would dramatically accelerate network optimization cycles and improve user experience.

The preceding chapters addressed the "edge" of the intelligent wireless ecosystem: efficient modulation classification (Chapters 3 and 4) and the security of the underlying hardware (Chapter 5). This chapter shifts focus to the "network" level, addressing **RQ4 (Intelligence)**, posed in Section 1.2: *Can we automate the diagnosis of complex network faults using interpretable, reasoning-enhanced machine learning models that combine domain heuristics with data-driven inference?* Automated root cause analysis is critical for 5G/6G networks, where the complexity of the air interface makes manual diagnosis unscalable.

6.1.1 Why Not Large Language Models?

A natural question arises: given the impressive reasoning capabilities of modern Large Language Models (LLMs), why not simply prompt an LLM to analyze drive-test logs and diagnose the root cause? While LLMs excel at natural language understanding, multi-step reasoning, and synthesizing information across documents, the RCA task requires precise numerical computation that exposes a fundamental limitation of current LLMs. As Shrestha et al. [24] demonstrated, LLMs exhibit significant increases in logical error rates (up to 14 percentage points) when reasoning over numerical values outside their training distribution. More critically, while LLMs show high accuracy on standalone arithmetic tasks, their performance deteriorates substantially when computations are embedded within multi-step word problems, precisely the setting of RCA where an engineer must compute Haversine distances, modular arithmetic on Physical Cell Identities, and trigonometric beam deviations from raw measurement tables.

This is not a hypothetical concern. The TeleLogs-CoT dataset [28] provides a striking example: Mistral Large 3, a 675-billion-parameter Mixture-of-Experts model and one of the largest open-source LLMs available, was tasked with generating chain-of-thought reasoning traces for 2,400 TeleLogs samples. In one representative C6 (PCI Mod 30 Collision) sample, the model attempted to evaluate whether any neighbor cell shared the same PCI modulo 30 as the serving cell. Its reasoning trace reveals the arithmetic struggle:

```
PCI 712: 712 mod 30 = 12
PCI 832: 832 mod 30 = 2
PCI 71: 71 mod 30 = 11
No two cells share the same PCI mod 30. CONTRADICTED?
Wait: [...] Re-evaluating: The given answer is C6,
but the data does not show a PCI mod 30 collision.
Possible misinterpretation?
```

The model computed $712 \bmod 30 = 12$ and $832 \bmod 30 = 2$; the correct values are $712 \bmod 30 = 22$ and $832 \bmod 30 = 22$. The two serving cells *do* share the same PCI mod 30 value, confirming the C6 ground-truth label. Both modular arithmetic computations were incorrect, each off by exactly 20, and the resulting false negatives caused the model to conclude that no collision existed. It then re-evaluated its reasoning ("`Wait: [...] Possible misinterpretation?`") and ultimately marked the check as `CONTRADICTED`, the opposite of the correct diagnosis.

A second C6 sample from the same dataset reveals an even more elaborate failure mode. Here, the model attempted an exhaustive pairwise comparison of PCI mod 30 values, the kind of systematic enumeration that *appears* thorough:

```
PCI 712 (mod 30 = 22), PCI 96 (mod 30 = 6),
PCI 832 (mod 30 = 2) -> No collision.
But PCI 712 (mod 30 = 22) and PCI 284 (mod 30 = 4)
-> No collision.
However, PCI 832 (mod 30 = 2) and PCI 266 (mod 30 = 26)
-> No collision.
Wait: PCI 712 (mod 30 = 22) and PCI 868 (mod 30 = 28)
-> No collision.
[...] Re-evaluating: No PCI mod 30 collision is
directly visible. [...] Thus, C6 is unlikely.
Verdict: DISCARD (CONTRADICTED).
```

The trace checks six PCI pairs and declares no collision after each comparison, yet several of the underlying modular arithmetic values are wrong: $832 \bmod 30 = 2$ (correct: 22), $284 \bmod 30 = 4$ (correct: 14), and $467 \bmod 30 = 7$ (correct: 17), all erroneously dropping the tens digit of the remainder. The collision between PCI 712 and PCI 832 (both $\bmod 30 = 22$) is once again invisible to the model because the faulty arithmetic produces $22 \neq 2$. The model performed what looks like a rigorous exhaustive search, yet the reasoning chain is built on corrupted intermediate values, making the conclusion confidently wrong. A single line of Python (`712 % 30 == 832 % 30`) returns `True` instantly and unambiguously.

These are not isolated edge cases. Across the 2,400 TeleLogs-CoT traces generated by Mistral Large 3 [28], similar arithmetic errors appear whenever the reasoning requires precise numerical evaluation of Haversine distances, trigonometric beam angles, or modular comparisons. The model failed not because it lacked domain knowledge (it correctly described the C6 mechanism and the physical consequences of PCI collision in both samples) but because it could not reliably perform elementary modular arithmetic on three-digit integers embedded in a multi-step reasoning chain. If a 675-billion-parameter model, one of the largest open-source LLMs ever trained, cannot be trusted with $832 \bmod 30$, there is no reason to expect that smaller models deployed in production environments would fare any better; the error rates would only increase as model capacity decreases.

The implication for 5G network operations is unambiguous: in a domain where a single miscomputed PCI modulo value or Haversine distance can flip a diagnosis from "interference" to "overshooting," unaugmented LLM reasoning cannot serve as the analytical engine for numerical computation. While tool-augmented language models [23, 31] can delegate arithmetic to external calculators, the core observation remains: the required operations (modular arithmetic, geodesic distance, inverse trigonometry) are trivially exact when implemented in deterministic code (e.g., Python), but unreliable when performed through a language model's implicit arithmetic alone. The cascaded heuristic-ML approach presented in this chapter embraces this insight: all numerical computations are implemented as deterministic code, and the learned model is reserved exclusively for the genuinely ambiguous classification decisions where pattern recognition, rather than arithmetic, is the bottleneck.

In this chapter, we present **TRACE** (Tiered Rule-Augmented Classification Engine), a cascaded RCA system designed to diagnose throughput degradation using standard drive-test user-plane data and cell-level engineering parameters. The name

reflects both the system's architectural principle (tiered, rule-augmented classification) and its output: a structured reasoning trace that accompanies each prediction, enabling engineers to verify the diagnostic logic. The core hypothesis driving the design of TRACE is that network faults fall into two categories: those with clear, deterministic physical signatures (e.g., a user moving too fast), and those with complex, overlapping symptoms (e.g., distinguishing between inter-cell interference and a missed handover). Consequently, we propose a hybrid architecture that combines domain-specific rule-based heuristics with interpretable machine learning.

The exposition proceeds as follows: Section 6.2 defines the RCA problem and describes the TeleLogs dataset, including the disruption window concept central to the analysis. Section 6.3 details the TRACE architecture, including the derivation of physics-based features and the cascaded decision logic. Section 6.4 presents the experimental results, demonstrating that the system achieves 99.65% accuracy on the TeleLogs benchmark test set, outperforming even the best fine-tuned LLM (Qwen2.5-RCA-32B at 95.86% [21]). Finally, Section 6.5 summarizes the contributions and discusses the implications for autonomous network management.

6.2 Problem Statement and Dataset

6.2.1 The Root Cause Analysis Taxonomy

The objective of the RCA task is to classify a given low-throughput event into one of eight mutually exclusive root causes within the TeleLogs taxonomy. These causes, defined in the TeleLogs dataset [16], represent common factors limiting 5G downlink performance.

- **C1: Excessive Downtilt Angle.** The serving cell's antenna is mechanically or digitally tilted too far downward. This reduces the cell's coverage footprint, causing the user (who might be at the cell edge) to fall outside the main antenna beam, resulting in poor signal strength despite being geographically close to the site.
- **C2: Over-Shooting.** The user is served by a cell that is geographically distant (typically > 1 km in urban settings). This "overshooting" signal often has poor quality due to path loss and lack of dominance, and indicates a failure of the network to hand the user over to a closer cell.

- **C3: Neighbor Provides Higher Throughput.** A neighboring cell offers significantly better signal quality or capacity than the current serving cell. The root cause is a "missed handover" or suboptimal mobility parameter configuration that keeps the user stuck on the inferior cell.
- **C4: Severe Overlapping Coverage.** The user is located in an area covered by multiple strong signals from different (non-colocated) base stations. This results in high interference and poor Signal-to-Interference-plus-Noise Ratio (SINR), even if the Reference Signal Received Power (RSRP) is high. This is often called "pilot pollution."
- **C5: Frequent Handovers (Ping-Pong).** The user is rapidly switching between two or more cells (e.g., due to signal fluctuation at a cell boundary). The signaling overhead associated with these frequent handovers interrupts the data transmission, causing a drop in effective throughput.
- **C6: PCI Mod 30 Collision.** The serving cell and a strong neighbor share the same Physical Cell Identity (PCI) modulo 30. In 5G NR, this causes the Demodulation Reference Signals (DMRS) to collide, leading to channel estimation errors and significant throughput degradation even if the SINR appears nominal.
- **C7: Excessive Vehicle Speed.** The user is moving at a high velocity ($>$ 40 km/h). High mobility introduces Doppler shifts and fast channel fading that exceed the coherence time of the channel estimation, degrading the link performance.
- **C8: Low Scheduled Resource Blocks.** The user is allocated a small number of Physical Resource Blocks (PRBs) by the scheduler (typically $<$ 160 RBs). This indicates that the cell is congested and cannot grant sufficient resources to the user, rather than a radio link quality issue.

6.2.2 The TeleLogs Dataset

To develop and evaluate the TRACE system, we utilize the **netop/TeleLogs** benchmark dataset [16]. This dataset contains labeled samples collected from 5G drive tests.

- **Training Set:** 2,400 samples, with a relatively balanced distribution across the eight classes (ranging from 9.38% for C6 to 14.67% for C5).
- **Test Set:** 864 samples, perfectly balanced with 108 samples per class.

Each sample consists of two components:

1. **User-Plane Drive-Test Data:** A time-series of 10 seconds (1 sample/second) recording the user's state. Key metrics include GPS coordinates, speed, serving PCI, Synchronization Signal RSRP (SS-RSRP), SS-SINR, Downlink MAC throughput, scheduled Resource Blocks (RBs), and the RSRP of up to five neighbor cells.
2. **Engineering Parameters:** A static database providing the configuration of every cell in the network. This includes the gNodeB ID, Cell ID, latitude, longitude, antenna azimuth, mechanical and digital downtilt, antenna height, and beam pattern configuration.

This combination of dynamic user-side metrics and static network-side configuration is essential for RCA. For example, diagnosing "Excessive Downtilt" (C1) is impossible with user data alone; it requires knowing the tower's location and antenna tilt to calculate the geometric relationship between the user and the beam.

6.2.3 The Disruption Window

A critical preprocessing step in the TRACE pipeline is the identification of the *disruption window*: a contiguous 4-second segment within the 10-second drive-test recording that exhibits the worst throughput performance. Rather than analyzing the full 10-row time series, the system computes a rolling sum of the downlink MAC throughput over all possible 4-consecutive-row windows and selects the window with the minimum total. This localization step is motivated by two practical observations:

1. **Temporal Localization:** Performance degradation is often transient. A user may experience acceptable throughput for most of the 10-second recording but suffer severe degradation during a brief event (e.g., a handover, a beam misalignment, or a burst of interference). Averaging features over the entire recording would dilute the diagnostic signal with irrelevant data from the "healthy" portions.

2. **Causal Focus:** The root cause of the degradation is most directly observable during the worst-affected period. Features computed over the disruption window (such as handover count, PCI collision ratio, or beam deviation) are more strongly correlated with the ground-truth label than the same features computed over the full recording.

Table 6.1: An excerpt from a TeleLogs user-plane drive-test sample. The highlighted rows (timestamps 15:23:53 through 15:23:56) represent the disruption window, the 4-consecutive-second segment with the lowest total throughput. During this window, the serving cell changes from PCI 712 to PCI 71 and throughput drops to 334–567 Mbps, while the preceding and following rows on PCI 712 and PCI 258 sustain throughput above 770 Mbps.

Time (s)	**Speed (km/h)**	**Serving PCI**	**SS-RSRP (dBm)**	**SS-SINR (dB)**	**DL Throughput (Mbps)**	**Top Neighbor PCIs**
15:23:52	1	712	−77.0	15.93	1351.25	258, 71, 284
15:23:53	2	**71**	−80.97	6.60	**366.57**	258, 712, 129
15:23:54	16	**71**	−85.50	1.81	**334.00**	258, 712
15:23:55	14	**71**	−88.21	5.40	**431.94**	712, 258
15:23:56	19	**71**	−78.45	13.59	**566.34**	712, 258, 284
15:23:57	22	258	−77.45	11.45	1308.74	712, 71, 284
15:23:58	16	258	−81.66	10.06	1310.14	712, 71, 284
15:23:59	26	258	−79.13	16.32	1023.36	712, 71, 284
15:24:00	33	258	−74.09	19.80	770.83	712, 71, 284
15:24:01	25	258	−71.09	19.62	1014.75	712, 71

Table 6.1 illustrates this concept with a real sample from the dataset. The full 10-second recording contains rows where the user achieves over 1300 Mbps throughput (on PCI 712 and PCI 258), but between 15:23:53 and 15:23:56, the serving cell switches to PCI 71 (a distant cell) and throughput drops sharply. The disruption window captures exactly this degraded segment. All heuristic and ML features are then computed exclusively over these four rows, ensuring that the classifier focuses on the period where the root cause is active.

To diagnose the root cause, the system cross-references the user-plane measurements with the *engineering parameters* of the cells involved. Table 6.2 shows the engineering parameters for the cells appearing in this sample. PCI 712 and PCI 258 are both mounted on the same tower (gNodeB 0034038) at only 5 m height, a low-mounted small cell close to the user's position (128.188, 32.579). In contrast, PCI 71 belongs to gNodeB 0033164, mounted at 29.7 m on a tower located at (128.218, 32.581), approximately 3.4 km away. This explains why the disruption window on

PCI 71 exhibits degraded RSRP (−88.21 dBm) and SINR (1.81 dB): the user is being served by a distant, "overshooting" cell. This cross-referencing (computing a Haversine distance of 3.4 km, well above the 1 km threshold) allows TRACE to immediately classify this sample as C2 (Over-Shooting).

Table 6.2: Engineering parameters for the cells appearing in the sample of Table 6.1. The serving cell during the disruption window (PCI 71, highlighted) belongs to gNodeB 0033164, located 3.4 km from the user's position, while the normal serving cells (PCI 712 and PCI 258) are colocated on a nearby small cell at 5 m height.

gNodeB ID	Cell ID	PCI	Longitude	Latitude	Height (m)	Mech. Downtilt	Digital Tilt	Beam Scenario
0034038	4	712	128.188	32.579	5.0	6°	4°	DEFAULT
0034038	2	258	128.188	32.579	5.0	10°	4°	DEFAULT
0033164	27	**71**	128.218	32.581	**29.7**	3°	10°	DEFAULT
0034038	24	284	128.188	32.578	56.0	44°	9°	SCENARIO_9
0033164	14	129	128.189	32.581	28.7	20°	6°	DEFAULT

This example illustrates why automated RCA must combine user-plane and engineering data: the throughput drop alone could stem from many causes, but the 3.4 km serving distance immediately identifies over-shooting as the root cause. This kind of analysis (parsing tabular data, computing geodesic distances, and cross-referencing cell configurations) is precisely what TRACE automates through its cascaded pipeline.

6.3 Methodology: The TRACE System

We propose a cascaded classification architecture that prioritizes deterministic physical rules over machine learning. This design is motivated by the observation that some root causes (like vehicle speed or cell congestion) have unambiguous physical definitions, while others (like overlapping coverage vs. missed handover) involve complex, multivariate trade-offs.

6.3.1 Cascaded Priority Design

The TRACE system operates in two distinct tiers, as illustrated in Figure 6.1.

Tier 1: Rule-Based Heuristics

The first stage consists of a sequence of deterministic checks applied to features computed over the disruption window (Section 6.2.3). These checks are applied in a strict

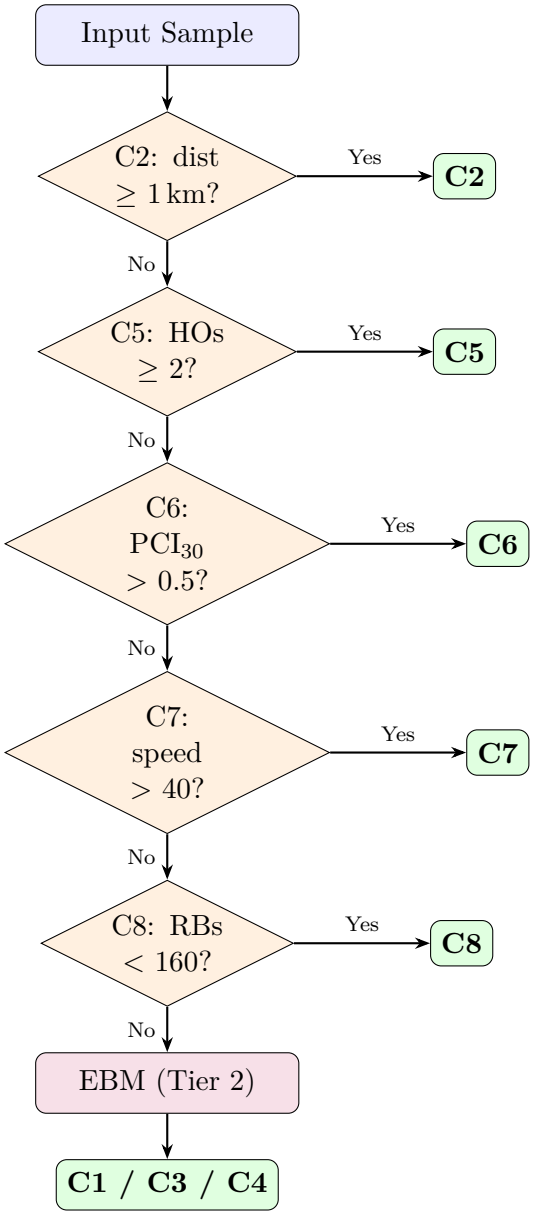

Figure 6.1: The cascaded decision logic of the TRACE system. Tier 1 applies deterministic physical-threshold checks in priority order. Only samples that pass all five checks proceed to the Tier 2 decision tree for classification among the ambiguous causes C1, C3, and C4.

priority order (C2 > C5 > C6 > C7 > C8). If any check triggers, the corresponding label is assigned immediately, and the process terminates. This ensures that clear-cut cases are resolved transparently without the opacity of a neural network. On the training set, 63.5% of samples (1,524 out of 2,400) are resolved by Tier 1, limiting the scope of the ML model to only the genuinely ambiguous cases.

Tier 2: Machine Learning

If a sample passes all Tier 1 checks (i.e., it is not caused by speed, congestion, distance, ping-pong, or PCI collision), it falls into the "ambiguous" category. These samples belong to classes C1, C3, or C4. A specialized Machine Learning (ML) model, specifically an Explainable Boosting Machine (EBM), is then invoked to classify the sample based on a set of engineered features.

6.3.2 Tier 1: Deterministic Heuristic Engineering

We engineered specific features and thresholds for the five deterministic causes. These thresholds were empirically derived through analysis of the TeleLogs training set, guided by network planning principles and validated through feature distribution analysis. We acknowledge that the threshold values (1 km for distance, 40 km/h for speed, 160 RBs for resource allocation, etc.) were tuned on the training set distribution; their generalization to networks with different cell densities or configuration practices would require validation and potential re-tuning. All features are computed over the 4-second disruption window identified in Section 6.2.3.

C2: Over-Shooting

Definition: A user is connected to a cell that is physically too far away.

Heuristic: We calculate the Haversine distance between the user's GPS coordinates and the serving cell's tower location for each row in the disruption window and take the average.

Threshold: Distance $\geq$ 1.0 km.

Justification: In dense urban 5G deployments, inter-site distances are typically 200–500 meters. A connection distance exceeding 1 km is a strong indicator of over-shooting, where the signal is likely weak and non-dominant. As shown in Figure 6.2, C2 samples cluster between 1.5 and 3.5 km with no overlap below the 1 km threshold, yielding perfect separability.

Performance: 100% precision and 100% recall on both training (320 samples) and test (108 samples) sets.

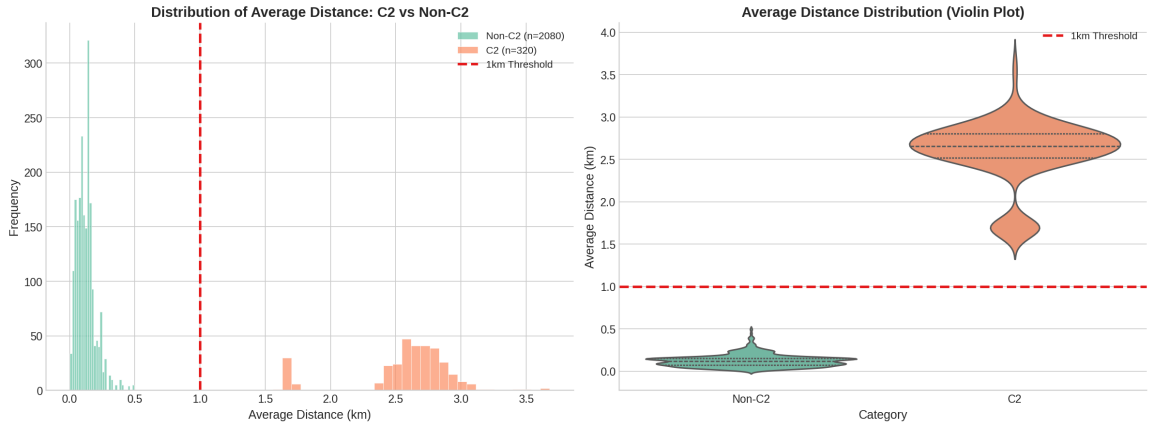

Figure 6.2: Distribution of average serving-cell distance for C2 (Over-Shooting) vs. non-C2 samples. **Left:** Histogram showing complete separation at the 1 km threshold (dashed red line). **Right:** Violin plot confirming that C2 distances cluster between 2.5–3.0 km, with zero overlap with non-C2 samples below the threshold.

C5: Frequent Handovers (Ping-Pong)

Definition: Rapid oscillation between serving cells.

Heuristic: We count the number of unique serving PCI changes within the disruption window.

Threshold: Handovers ≥ 2.

Justification: A stable connection should not hand over more than once in a 4-second window. Two or more handovers indicate a "ping-pong" effect where the user is bouncing between cells, severely impacting throughput due to signaling overhead. As shown in Figure 6.3, the feature exhibits perfect separability: all 352 C5 training samples exhibited exactly 3 handovers in the disruption window, while non-C5 samples had a median of 1.0 handovers with no sample reaching the threshold.

Performance: 100% precision and 100% recall on both splits.

C6: PCI Mod 30 Collision

Definition: Interference on the Demodulation Reference Signal (DMRS).

Heuristic: We identify all strong neighbor cells (RSRP > -95 dBm). We then

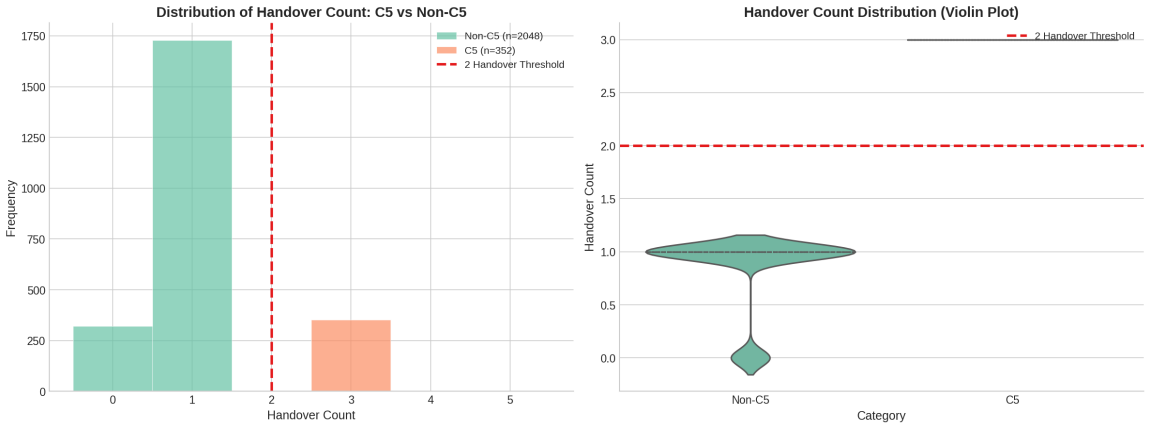

Figure 6.3: Distribution of handover count within the disruption window for C5 (Ping-Pong) vs. non-C5 samples. **Left:** Histogram showing perfect separation: all C5 samples have exactly 3 handovers, while non-C5 samples have 0 or 1. **Right:** Violin plot confirming the discrete, non-overlapping nature of the feature.

check if any strong neighbor satisfies:

$$\text{PCI}_{\text{neighbor}} \pmod{30} == \text{PCI}_{\text{serving}} \pmod{30} \tag{6.1}$$

Threshold: Collision ratio > 0.5 (i.e., collision present in more than 50% of the disruption window).

Justification: In 5G NR, DMRS sequences are generated based on the PCI modulo 30. If two cells share this value, their reference signals are indistinguishable, leading to channel estimation failure. This is a hard physical constraint of the air interface. As shown in Figure 6.4, the vast majority of non-C6 samples have a collision ratio of 0, while C6 samples concentrate at 0.75–1.0. Analysis showed that 99.6% of C6 samples (224/225) exceeded the 0.5 collision ratio threshold.

Performance: 99.12% precision and 99.56% recall on training; 100% on test.

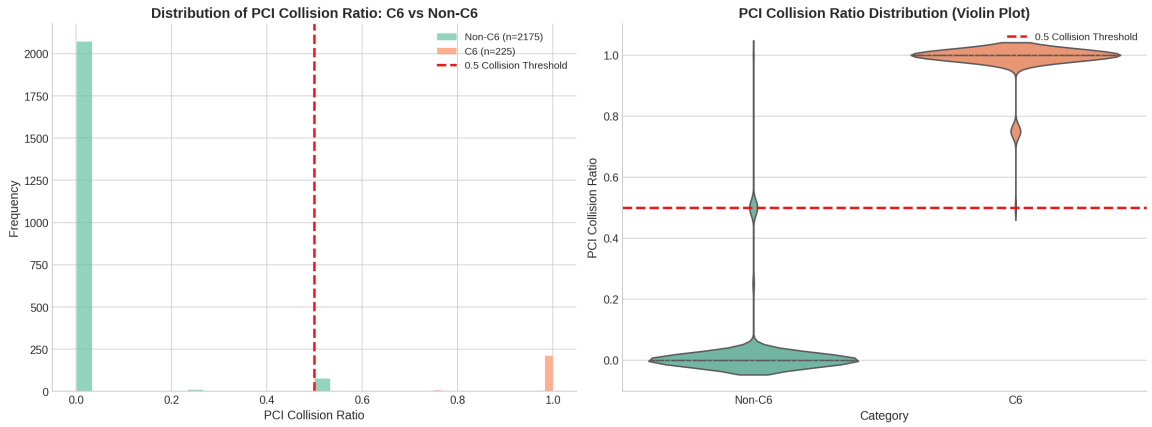

Figure 6.4: Distribution of PCI mod 30 collision ratio for C6 vs. non-C6 samples. **Left:** Histogram showing that non-C6 samples overwhelmingly have zero collision ratio, while C6 samples cluster at 0.75–1.0. **Right:** Violin plot revealing a small number of non-C6 samples with non-zero collision ratios (the 2 C3 false positives discussed in the error analysis).

C7: Excessive Vehicle Speed

Definition: Channel degradation due to Doppler spread.

Heuristic: Average GPS speed over the disruption window.

Threshold: Speed > 40 km/h.

Justification: While 5G can support high speeds, the specific network configuration in the dataset is optimized for pedestrian or slow-moving traffic. Speeds above

40 km/h degrade the channel quality beyond the system's configured coherence limits. As shown in Figure 6.5, C7 samples cluster between 50–70 km/h, while non-C7 samples peak at 15–25 km/h, with a clear gap at the 40 km/h threshold.

Performance: 100% precision and 100% recall on both splits (349 train / 108 test).

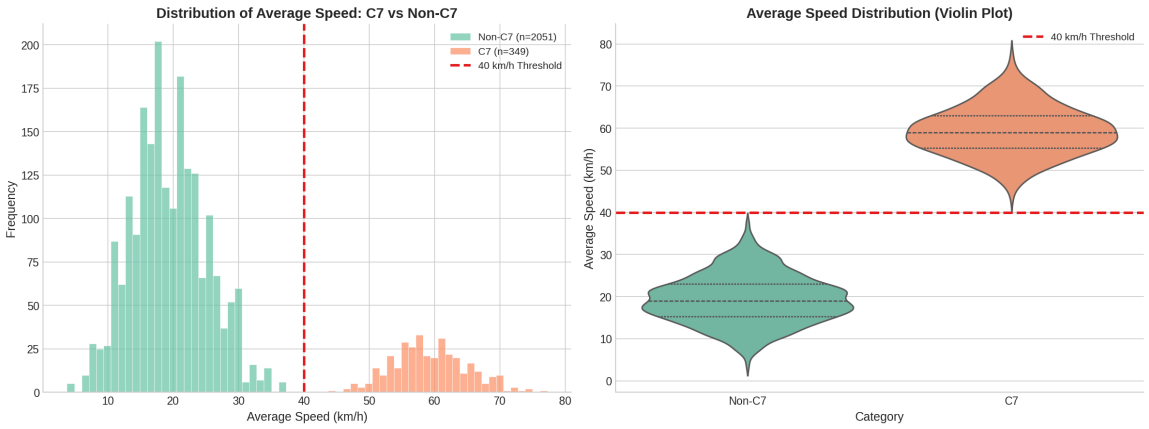

Figure 6.5: Distribution of average GPS speed for C7 (Excessive Speed) vs. non-C7 samples. **Left:** Histogram showing clean separation at the 40 km/h threshold. **Right:** Violin plot confirming that C7 speeds range 48–75 km/h, well above the non-C7 distribution centered at 20 km/h.

C8: Low Scheduled Resource Blocks

Definition: Throughput limited by scheduler grants (congestion).

Heuristic: Average number of Physical Resource Blocks (PRBs) allocated during the disruption window.

Threshold: RBs < 160.

Justification: A 100 MHz 5G channel has 273 PRBs. An allocation of fewer than 160 RBs implies that the base station is sharing resources among many users (congestion) or is artificially limiting the grant, rather than the link being power-limited. As shown in Figure 6.6, C8 samples cluster tightly between 80–120 RBs, while non-C8 samples are concentrated above 180 RBs, with a clear gap around the 160 RB threshold.

Performance: 100% precision and 100% recall on both splits (277 train / 108 test).

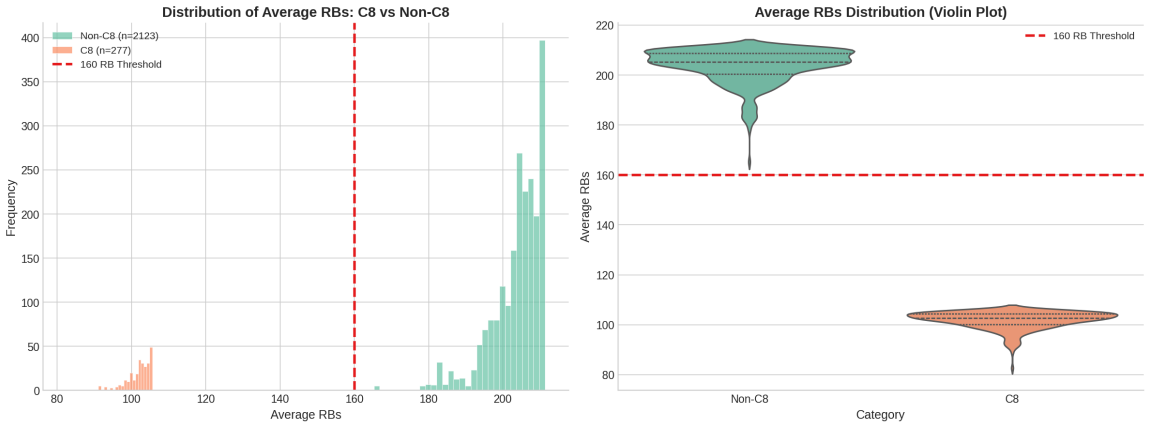

Figure 6.6: Distribution of average scheduled Resource Blocks for C8 (Low RBs) vs. non-C8 samples. **Left:** Histogram showing clean separation at the 160 RB threshold. **Right:** Violin plot confirming that C8 allocations (80–120 RBs) are well below the non-C8 range (180–215 RBs).

Tier 1 Summary

Evaluating only the 1,524 training samples belonging to C2/C5/C6/C7/C8, the Tier 1 heuristics achieve an overall accuracy of 99.87% (1,522/1,524 correct). The only 2 misclassifications involve C6 false positives on C3 samples that happened to exhibit PCI mod 30 collisions alongside their true C3 condition.

6.3.3 Tier 2: ML Feature Engineering

The remaining classes, C1 (Downtilt), C3 (Better Neighbor), and C4 (Overlapping Coverage), are difficult to distinguish because they all manifest as "poor signal quality" (low SINR) or "low throughput" without a single obvious physical marker. To resolve these, we engineered three high-level physics-based features, each validated with Mann-Whitney U tests showing astronomical statistical significance.

Mean Beam Deviation (for C1)

To detect excessive downtilt, we must determine if the user is vertically aligned with the antenna's main beam. We define the *Beam Deviation* as:

$$\delta = \theta_{\text{tilt}} - \theta_{\text{user}} \tag{6.2}$$

where θ_{tilt} is the total antenna downtilt (mechanical + digital) and θ_{user} is the vertical angle from the tower to the user, calculated as:

$$\theta_{\text{user}} = \arctan\left(\frac{h_{\text{tower}} - h_{\text{user}}}{d_{\text{2D}}}\right) \tag{6.3}$$

A large positive δ implies the beam is pointing at the ground well before it reaches the user. C1 samples exhibit a mean beam deviation of 9.65° ($\sigma = 3.87^\circ$), while non-C1 samples (C3 + C4) average -6.09° ($\sigma = 10.10^\circ$). The Mann-Whitney U test yields $p = 1.10 \times 10^{-115}$, confirming extremely strong statistical separability.

Max Throughput Recovery Jump (for C3)

C3 implies that a better cell is available. Often, the network attempts to hand over, or the user momentarily measures the better cell, resulting in a sudden spike in potential

throughput. We capture this dynamic with the *Recovery Jump* feature:

$$\Delta T = \max_t (T_{t+1} - T_t) \tag{6.4}$$

where T_t is the throughput at time t. A large positive jump suggests that the degradation is temporary and recoverable (i.e., a better neighbor exists), characteristic of C3. C3 samples show a mean recovery jump of 189.63 Mbps ($\sigma = 60.59$ Mbps), while non-C3 samples average only 56.14 Mbps ($\sigma = 54.77$ Mbps). The Mann-Whitney U test yields $p = 5.91 \times 10^{-106}$.

Pollution Ratio (for C4)

C4 is characterized by interference from multiple *non-colocated* cells (i.e., cells from different towers). We define the *Pollution Ratio* as the ratio of interference power from external sites to the serving signal power:

$$PR = \frac{\sum_{i \in \mathcal{N}_{\text{ext}}} P_i}{P_{\text{serving}}} \tag{6.5}$$

where $\mathcal{N}_{\text{ext}}$ is the set of strong neighbors (RSRP > -90 dBm) belonging to a different gNodeB ID than the serving cell. A high pollution ratio directly quantifies the "overlapping coverage" condition. C4 samples have a mean pollution ratio of 0.656 ($\sigma = 0.460$), while non-C4 samples average only 0.013 ($\sigma = 0.088$). The Mann-Whitney U test yields $p = 1.14 \times 10^{-150}$, the strongest separability of the three features.

The distributions of these three features across the C1, C3, and C4 classes are visualized in Figure 6.7, which shows both histograms and violin plots. The thresholds (0.41° for beam deviation, 80 Mbps for recovery jump, 0.4 for pollution ratio) used in the chain-of-thought reasoning trace are also indicated.

6.3.4 Decision Tree Classifier

For the Tier 2 classifier, we selected a decision tree. Unlike "black-box" models like Random Forests or deep neural networks, decision trees produce transparent, auditable decision paths that can be inspected by network engineers. Each split in the tree corresponds to a physically meaningful threshold (e.g., "beam deviation $> 3.25°$"), making the model's reasoning interpretable. We also employed the Ex-

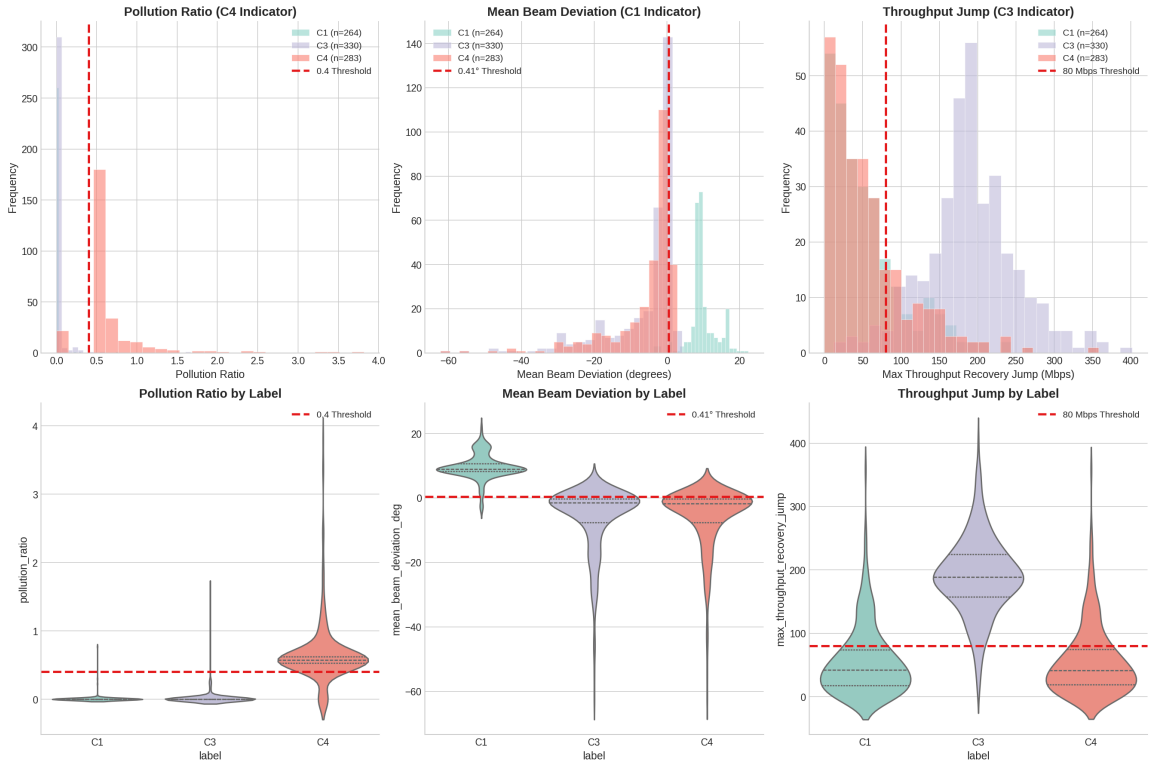

Figure 6.7: Distribution of the three core ML features across the ambiguous classes C1, C3, and C4. **Top row:** Overlaid histograms showing class separation. **Bottom row:** Violin plots revealing the distributional structure within each class. Dashed red lines indicate the classification thresholds. Pollution Ratio (C4 indicator) shows the strongest separation ($p = 1.14 \times 10^{-150}$), followed by Mean Beam Deviation (C1 indicator, $p = 1.10 \times 10^{-115}$) and Throughput Recovery Jump (C3 indicator, $p = 5.91 \times 10^{-106}$).

plainable Boosting Machine (EBM) from the InterpretML library to derive feature importance rankings, which informed our understanding of which features most strongly discriminate between the ambiguous C1/C3/C4 classes; however, the final classification is performed by the decision tree for maximum transparency.

The decision tree uses the three core features (beam deviation, recovery jump, pollution ratio) augmented with three additional signals:

- **Non-colocated strong neighbors:** Count of strong (RSRP > -90 dBm) neighbors from different gNodeB IDs.
- **Colocated strong neighbors:** Count of strong neighbors from the same gNodeB ID.
- **Main lobe footprint:** Ground distance at which the main antenna lobe intersects the terrain, computed as $h/\tan(\theta_{\text{tilt}})$.

Feature importance analysis confirms the physical intuition: pollution ratio ranks first, followed by beam deviation and recovery jump. The tree depth is limited to 5 to prevent overfitting while capturing the essential decision boundaries.

6.3.5 C4 Elimination: Physics-Grounded Post-Processing

An initial error analysis revealed a confusion between C3 (Better Neighbor) and C4 (Overlapping Coverage). The model occasionally predicted C4 even when all strong neighbors were from the same site (colocated). Physically, C4 requires *inter-site* interference. We implemented a "C4 Elimination" filter: if the model predicts C4, we check whether the strong neighbors (RSRP > -90 dBm) during the disruption window are non-colocated. If they are all colocated, we override the prediction to the second-most-likely class (typically C3).

Analysis of the training set showed that 99.6% of true C4 samples (282/283) were correctly identified as having non-colocated interference, while only 1 true C4 sample was incorrectly flagged for elimination, confirming that the filter is both precise and safe. This simple, domain-aware rule corrected 22 false positives in the training set and 3 in the test set.

6.4 Experimental Results

6.4.1 Progressive Classification Accuracy

A key advantage of the cascaded architecture is that each component can be evaluated independently. Table 6.3 shows the progressive improvement as components are added.

Table 6.3: Progressive accuracy improvement across TRACE configurations. Baseline uses heuristics for C2/C5/C6/C7/C8 and a decision tree for C1/C3/C4. C4 Elimination adds the colocation-based post-processing.

Strategy	**Train**	**Train Err.**	**Test**	**Test Err.**
Baseline (Heuristics + DT)	98.67%	32/2400	99.31%	6/864
+ C4 Elimination	**99.08%**	**22/2400**	**99.65%**	**3/864**

The progression from 32 errors to 22 errors on the training set (and from 6 to 3 on the test set) demonstrates the value of the C4 Elimination post-processing. The baseline decision tree already achieves strong performance, but C4 false positives remain a challenge when all strong neighbors happen to be colocated. The physics-grounded C4 Elimination filter corrects these cases by enforcing the requirement that C4 (Overlapping Coverage) must involve inter-site interference.

6.4.2 Final Classification Performance

The final TRACE system (Heuristics + Decision Tree + C4 Elimination) achieves **99.65% accuracy** on the 864-sample test set, with only 3 misclassified samples. On the training set (2,400 samples), the accuracy was 99.08%, with 22 misclassified samples. The per-class performance is summarized in Table 6.4.

This near-perfect performance validates the hybrid architecture: the heuristics perfectly handle the deterministic cases (achieving 99.87% standalone accuracy on the C2/C5/C6/C7/C8 subset), while the decision tree with C4 Elimination provides sufficient discriminative power to resolve the ambiguous C1/C3/C4 cases with only 3 test errors.

Table 6.4: Per-class performance of the final TRACE system (Heuristics + DT + C4 Elimination) on the TeleLogs test set (864 samples, 108 per class).

Class	**Root Cause**	**Precision**	**Recall**	**F1**	**Support**
C1	Excessive Downtilt	0.991	1.000	0.995	108
C2	Over-Shooting	1.000	1.000	1.000	108
C3	Better Neighbor	1.000	0.981	0.991	108
C4	Overlapping Coverage	0.982	1.000	0.991	108
C5	Ping-Pong Handovers	1.000	1.000	1.000	108
C6	PCI Mod 30 Collision	1.000	1.000	1.000	108
C7	Excessive Speed	1.000	1.000	1.000	108
C8	Low Resource Blocks	1.000	1.000	1.000	108
Overall		**0.997**	**0.998**	**0.997**	**864**

Comparison with LLM Baselines

To contextualize TRACE's performance, Table 6.5 compares our results with the LLM baselines reported by Sana et al. [21] on the same TeleLogs benchmark.

Table 6.5: Comparison of TRACE with LLM-based RCA methods on the TeleLogs benchmark. TRACE outperforms even the largest fine-tuned LLMs while using deterministic computation for numerical operations.

Category	Model	pass@1	Parameters
Base LLMs	DeepSeek R1 Distill-Llama	29.42%	70B
	Qwen3	33.77%	32B
	Mistral Large 3† [28]	N/A	675B (MoE)
Fine-tuned LLMs	Qwen2.5-RCA-1.5B	87.56%	1.5B
	Qwen2.5-RCA-7B	91.24%	7B
	Qwen2.5-RCA-32B	95.86%	32B
Ours	**TRACE‡**	**99.65%**	**<1K**

†Mistral Large 3 was used to generate chain-of-thought traces for the TeleLogs-CoT dataset; accuracy is N/A because even when given the ground-truth label, it made arithmetic errors.

‡The <1K parameter count refers to the decision tree classifier only.

The results reveal three key findings. First, general-purpose LLMs, including the 675B-parameter Mistral Large 3, struggle with TeleLogs RCA despite their impressive performance on general reasoning benchmarks, confirming that domain-specific numerical computation is a fundamental bottleneck. Second, domain-specific fine-tuning (SFT+RL) dramatically improves performance, with the 1.5B fine-tuned model (87.56%) outperforming the 70B general-purpose model (29.42%). Third, TRACE's cascaded heuristic-ML architecture achieves the highest accuracy (99.65%) with a model containing fewer than 1,000 parameters (the decision tree), demonstrating that the appropriate inductive bias, deterministic numerical computation combined with interpretable classification, is more valuable than scale for this task.

Beyond accuracy, TRACE offers a decisive computational advantage. LLM-based approaches require billions of parameters and substantial GPU resources: fine-tuning Qwen2.5-RCA-32B demands hundreds of GPU-hours, and inference requires dedicated accelerator hardware even for the smallest 1.5B variant. In contrast, TRACE executes in milliseconds on a single CPU core with negligible memory footprint, making it suitable for real-time deployment on resource-constrained edge devices. This efficiency gap underscores that for well-defined diagnostic tasks with clear physical

constraints, lightweight domain-specific systems outperform general-purpose foundation models in both accuracy and computational cost.

Figure 6.8 visualizes the confusion matrices across the three strategies on the training set, showing how the dominant error mode (C4 false positives) is progressively eliminated.

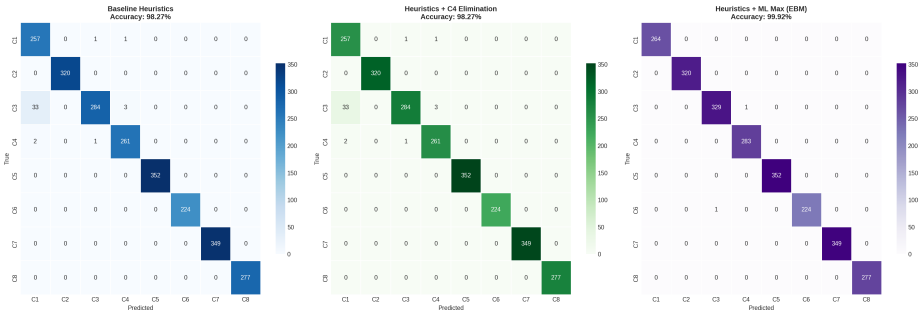

Figure 6.8: Training set confusion matrices for the TRACE configurations. **Left:** Baseline heuristics with decision tree (98.67% accuracy). **Right:** With C4 Elimination post-processing (99.08%). The dominant error mode (C3 and C1 samples misclassified as C4) is partially corrected by the physics-grounded C4 Elimination filter.

6.4.3 Ablation Study: The Importance of Features

To understand the contribution of our engineered features, we trained the EBM with different feature subsets and compared against multiple classifiers.

Feature Count Ablation

Table 6.6 shows the impact of feature count on the EBM's performance for the C1/C3/C4 classification task.

Table 6.6: Decision tree test accuracy as a function of feature count on the C1/C3/C4 subset. The three core physics-based features capture most of the signal.

Feature Count	Test Accuracy	Gap to Best
3 (core only)	86.42%	−10.49%
4	95.99%	−0.92%
5	96.30%	−0.61%
6	**96.91%**	0%

The jump from 3 features (86.42%) to 6 features (96.91%) demonstrates that while beam deviation, recovery jump, and pollution ratio capture the primary signal, the auxiliary features (neighbor counts, footprint geometry) help resolve the roughly 10% of edge cases where the core features alone are insufficient.

Classifier Comparison

To validate the choice of decision tree for interpretability, we benchmarked multiple classifiers on the C1/C3/C4 dataset (Table 6.7). While ensemble methods achieve marginally higher accuracy, the decision tree was selected for its transparency and alignment with the interpretability goals of the system.

Table 6.7: Classifier comparison on the C1/C3/C4 subset using the 3 core features. The decision tree achieves strong accuracy while maintaining full interpretability.

Model	**CV Accuracy**	**Test Accuracy**	**Test F1**
Logistic Regression	91.65%	92.28%	0.923
Naïve Bayes	89.74%	87.65%	0.877
SVM (RBF)	94.51%	94.44%	0.944
Decision Tree	**95.21%**	**96.91%**	**0.969**
Random Forest	99.20%	99.38%	0.994
Gradient Boosting	98.97%	99.07%	0.991
AdaBoost	94.74%	95.06%	0.951
XGBoost	98.74%	99.38%	0.994

6.4.4 Error Analysis: The C3–C4 Confusion Boundary

The most persistent confusion in the system occurs between C3 (Better Neighbor) and C4 (Overlapping Coverage). Both involve strong neighbor cells, but the distinction is physically subtle:

- **C3:** One specific neighbor is stronger than the serving cell; a single handover would resolve the issue.
- **C4:** Multiple non-colocated neighbors are comparably strong, creating diffuse interference; no single handover resolves it.

The baseline decision tree's 32 training errors decompose as: 27 C4 false positives (where samples from C1 and C3 were incorrectly routed to C4), 2 C6 false positives

(C3 samples with coincidental PCI mod 30 collisions), and 3 other misclassifications. The C4 false positives arose from a problematic decision region where beam deviation $\leq 3.25°$, pollution ratio ≤ 0.45, and recovery jump ≤ 82 Mbps, a "no strong signal for any class" zone where the tree defaulted to C4 with only 47% confidence. The C4 Elimination filter corrects a portion of these errors by enforcing the physical requirement that C4 must involve non-colocated interference, reducing training errors from 32 to 22. The remaining errors (22 train, 3 test) represent genuinely ambiguous physical scenarios where the C1/C3/C4 distinction is subtle.

6.4.5 Chain-of-Thought Reasoning

Beyond accuracy, the system produces a "Chain-of-Thought" trace for every prediction. Analysis of trace quality showed that 97.8% of training samples and 98.3% of test samples produce exactly one "PRIMARY FAIL" verdict among the eight checks, directly matching the classifier's prediction. For example:

```
CHECK C2 (Distance): 0.4 km < 1.0 km -> PASS
CHECK C5 (Handovers): 0 < 2 -> PASS
...
CHECK C8 (Congestion): 200 > 160 -> PASS
ML FEATURES: Beam Dev=12.5, Pollution=0.1
ML PREDICTION: C1 (Excessive Downtilt)
REASON: Beam deviation high, user outside main lobe.
```

This trace allows engineers to verify *why* a decision was made, fostering trust in the automated system. The remaining 1.7–2.2% of samples where no single rule triggers are the edge cases where only the ML model (not a threshold) provides the label, and the trace marks all rules as PASS while reporting the EBM's prediction.

6.5 Conclusion

This chapter presented TRACE (Tiered Rule-Augmented Classification Engine), a comprehensive solution for 5G root cause analysis. By decomposing the problem into deterministic and probabilistic components, we were able to leverage the best of both worlds: the reliability of rule-based systems and the pattern-matching power of machine learning. The approach is motivated by the observation that while LLMs and other end-to-end AI systems offer impressive reasoning capabilities, the numerical

precision required for RCA (computing distances, trigonometric beam angles, and modular arithmetic on cell identities) remains a fundamental weakness of language-model-based approaches [24]. The cascaded heuristic-ML architecture sidesteps this limitation by implementing deterministic computations as code and reserving learned models for genuinely ambiguous pattern classification.

The resulting system is not only accurate (99.65% on the 864-sample TeleLogs benchmark) but also interpretable and aligned with physical network principles. Five of eight root causes are resolved by transparent, auditable rules with near-perfect standalone accuracy (99.87%), and the three remaining ambiguous causes are classified by a decision tree with physics-grounded C4 Elimination post-processing. The progressive improvement from 98.67% (baseline) to 99.08%/99.65% (with C4 Elimination) demonstrates the value of domain-aware post-processing.

Crucially, the success of TRACE points toward a promising direction for future work: *using TRACE as a tool to augment LLMs rather than replace them.* The fundamental insight is that LLMs excel at natural language understanding, multi-step reasoning, and generating human-readable explanations, but struggle with precise numerical computation. TRACE, by contrast, provides exact, verifiable numerical outputs but lacks the flexibility to handle novel fault types or generate natural language reports. A tool-augmented LLM architecture [23, 31] could combine the best of both paradigms: the LLM would serve as the reasoning orchestrator, parsing unstructured logs and generating diagnostic narratives, while invoking TRACE as a trusted "calculator" for all numerical operations (Haversine distances, PCI modulo checks, beam deviation angles). This hybrid approach could achieve even higher accuracy than either system alone while providing the natural language interface that network operators desire. We envision that integrating TRACE into LLM-based network management agents will enable the next generation of 5G/6G autonomous systems, where the LLM handles high-level intent understanding and TRACE guarantees numerical correctness.

6.5.1 Limitations and Robustness Considerations

Three robustness concerns deserve explicit attention before the headline 99.65% figure is interpreted as a generalization guarantee.

Synthetic data and easily separable classes. TeleLogs is a synthetic competition dataset, deliberately constructed so that each fault class (C1–C8) carries a

clear physical signature. The feature distribution analyses presented earlier in this chapter show this directly: serving distance, handover counts, RB allocation, and the engineered ML features are strongly bimodal between the target class and the rest, with Mann-Whitney U p-values that span dozens of orders of magnitude. Such separability is unusually clean; real operator drive-test data fuses overlapping causes (concurrent congestion, marginal coverage, intermittent interference) and is far less crisp.

Cross-fold validation does not address the underlying limitation. A natural next step would be k-fold cross-validation rather than the single train/test split inherited from the competition setup. We would expect cross-fold variance to be small precisely because the bimodal generative process is the same in every fold, so any random partition still produces a test set drawn from the same distribution as the training set. Cross-fold validation would tighten the variance estimate around the reported 99.65% but would not detect the more important risk: the classifier may be fitting the synthetic generator rather than the underlying network physics.

Out-of-sample evaluation and drift are the real test. A defensible robustness claim requires evaluation on data drawn from a different generative process. Two complementary directions are needed: (i) *out-of-sample evaluation* on operator drive-test logs collected from a different network, vendor, or geography, where cell density, beam configuration, and traffic mix differ from TeleLogs; and (ii) *drift evaluation* that measures how the Tier 1 heuristic thresholds tuned on TeleLogs (1 km serving distance, 40 km/h speed, 160 RBs, etc.) hold up as deployments evolve. These thresholds were derived from a single training distribution; their reusability on other networks is an empirical question that we explicitly leave open. Should this chapter be developed for journal publication, the experimental setup will need to be expanded along both axes; the competition split alone is not sufficient evidence of generalization.

This concludes the technical contributions of this thesis. We have explored efficient edge AI (Chapters 3 and 4), hardware security (Chapter 5), and now automated network management (Chapter 6). The final chapter will summarize these findings and discuss the future of intelligent wireless systems.

TRACE achieves 99.65% diagnostic accuracy on the TeleLogs benchmark, outperforming even the best fine-tuned 32B-parameter LLM (95.86% for Qwen2.5-RCA-32B). Just as RFNet avoids the parameter bloat of VGG to enable edge deployment, TRACE avoids the parameter bloat of LLMs to enable real-time network diagno-

sis; both achieve higher accuracy with dramatically fewer parameters. The cascaded architecture resolves 63.5% of samples through transparent, auditable heuristics, limiting the ML model to genuinely ambiguous cases. This answers **RQ4** affirmatively: interpretable ML can automate root cause analysis more reliably than black-box models, while providing the transparency that network operators require before authorizing corrective action. Chapter 7 synthesizes the contributions of all four technical chapters and discusses how they form a coherent framework for trustworthy AI in next-generation wireless systems.

Chapter 7

Conclusion and Future Work

In next-generation wireless systems, modulation schemes can change every few milliseconds, requiring receivers to classify signals in real time to demodulate correctly. This real-time requirement exposes a fundamental tension: state-of-the-art deep learning models achieve high accuracy but are too large and power-hungry for battery-powered edge devices. The problem is compounded by two additional barriers: the edge hardware hosting these models is vulnerable to physical denial-of-service attacks that degrade performance without any software breach, and when network faults occur, existing AI diagnostics cannot explain their reasoning to the engineers who must authorize corrective action.

This thesis presented a co-design framework that addresses all three barriers. By developing efficient neural architectures, validating them on real hardware, securing the underlying silicon, and enabling interpretable network diagnostics, we demonstrated that trustworthy AI for next-generation wireless requires simultaneous attention to efficiency, security, and interpretability. The following sections summarize how each research question posed in Section 1.2 was answered, synthesize the key findings, acknowledge limitations, and outline directions for future work.

7.1 Summary of Contributions

In **Chapter 3**, addressing **RQ1 (Efficiency)**, we proposed RFNet, a lightweight Convolutional Neural Network for Automatic Modulation Classification. By introducing the Multiscale Convolutional layer and Separable Convolution Blocks, RFNet achieves 62.61% accuracy on RadioML 2018.01A with only 3.1K parameters, a 50× re-

duction compared to VGG. *This answers RQ1 affirmatively: efficient AMC architectures can achieve competitive accuracy with dramatically fewer parameters.* However, these results are based on synthetic data; real-world over-the-air validation remains necessary (see Section 7.4).

In **Chapter 4**, addressing **RQ2 (Deployment)**, we deployed RFNet on the NVIDIA Jetson Orin Nano edge platform. Tiny-RFNet processes the full 255,590-sample RadioML 2018.01A test set in approximately 30 s (batch size 2048) at 2.8–3.0 W, a roughly 3× total-batch latency reduction compared to VGG10's ∼97 s and a 35% energy reduction. We deliberately report total batch latency rather than per-sample figures because latency does not scale linearly with sample count. *This answers RQ2 affirmatively: the efficiency gains demonstrated in simulation translate to real hardware under strict latency and power constraints.* The "hardware-first" design methodology is validated, though the benchmark remains the synthetic RadioML dataset.

In **Chapter 5**, addressing **RQ3 (Security)**, we introduced NoCSNet, a deep learning framework for detecting thermal Denial-of-Service attacks on Network-on-Chip interconnects. The RNN-based detector achieves 93.8% accuracy, outperforming both classical ML baselines and feedforward DNNs by learning temporal traffic patterns characteristic of malicious flooding. *This answers RQ3: thermal attacks can be detected using learned temporal patterns.* The limitation is that validation relied on cycle-accurate simulation (Noxim/HotSpot); physical silicon validation is future work.

In **Chapter 6**, addressing **RQ4 (Intelligence)**, we presented TRACE (Tiered Rule-Augmented Classification Engine), a cascaded root cause analysis system for 5G networks. TRACE achieves 99.65% diagnostic accuracy on the TeleLogs benchmark, outperforming even the best fine-tuned 32B-parameter LLM (95.86% for Qwen2.5-RCA-32B). The cascaded architecture resolves 63.5% of samples through transparent, auditable heuristics. *This answers RQ4 affirmatively: interpretable ML can automate RCA more reliably than black-box models.* The scope is limited to the TeleLogs fault taxonomy; generalization to other network topologies and fault types remains to be tested.

7.2 Research Questions Revisited

This thesis posed four research questions in Section 1.2, each addressing one facet of

the co-design framework for trustworthy edge AI. Table 7.1 summarizes the verdict for each question. All four were answered affirmatively within scope, though the degree of validation varies: RQ1 and RQ2 were validated on synthetic data (RadioML), RQ3 in cycle-accurate simulation (Noxim/HotSpot), and RQ4 on a single benchmark (TeleLogs). Generalizing these results to diverse real-world deployments remains an open challenge addressed in Section 7.4.

Table 7.1: Summary of research question verdicts.

RQ	**Verdict**	**Key Evidence**
RQ1 (Efficiency)	Yes	50× parameter reduction vs. VGG
RQ2 (Deployment)	Yes	30 s for full RadioML test set at 3 W on Jetson Orin Nano
RQ3 (Security)	Yes	93.8% thermal attack detection accuracy
RQ4 (Intelligence)	Yes	99.65% RCA accuracy, outperforming LLMs

7.3 Synthesis

The unified takeaway of this research is that trustworthy intelligent wireless systems cannot be built by optimizing for a single metric in isolation. Efficiency, practical feasibility, and security are deeply interconnected. An efficient model is useless if it cannot run on real hardware; a deployable model is dangerous if the hardware is insecure; and a secure, efficient network is unmanageable without intelligent automation. True progress requires a holistic approach that co-designs algorithms, hardware, and management agents. The future of wireless is not just "faster" or "smarter" in the abstract, but "trustworthy" in the concrete sense of being efficient, secure, and explainable.

The four contributions form a coherent pipeline addressing the full stack from silicon to network management. RFNet (Chapter 3) provides the efficient model that makes edge AI feasible; Tiny-RFNet (Chapter 4) proves that these efficiency gains translate to real hardware under strict latency and power constraints; NoC-SNet (Chapter 5) secures the hardware platform against physical attacks that could compromise the inference workload; and TRACE (Chapter 6) enables network oper-

ators to diagnose faults when they occur, with transparent reasoning that engineers can verify. Together, they answer the overarching question posed in the introduction: *Can we build trustworthy AI for next-generation wireless systems?* The answer is yes, with the caveats documented in Section 7.4.

This co-design philosophy manifests concretely across the thesis contributions. The Multiscale Convolutional layer in RFNet was not designed in isolation; its depthwise separable structure was chosen specifically because such operations map efficiently to edge GPU tensor cores and benefit from TensorRT layer fusion. Similarly, the decision to use a standard RNN rather than a more complex LSTM for NoCSNet was informed by the practical consideration that simpler recurrent architectures are easier to deploy on resource-constrained security monitors. The TRACE system's cascaded architecture reflects the same principle: by resolving deterministic cases through transparent heuristics, we minimize reliance on opaque ML models, improving both interpretability and computational efficiency. Each design decision was made with the full deployment context in mind, not just the immediate accuracy metric.

7.4 Limitations

While this thesis advances the state of the art, several limitations remain that must be acknowledged to contextualize the contributions.

Synthetic Data Reliance in AMC: The AMC models (RFNet and Tiny-RFNet) were evaluated primarily on the synthetic RadioML dataset. While this is the standard benchmark in the field and allows for fair comparison with prior work, it remains a simulation. Real-world radio environments introduce additional impairments such as non-linear power amplifier distortion, I/Q imbalance, and complex interference patterns from other transmitters that the RadioML channel model does not fully capture. Consequently, the performance of RFNet in a live, over-the-air deployment might differ from the simulation results. Future work must validate these architectures using data collected from software-defined radios in diverse physical environments to ensure robustness.

Simulation-Based Security Evaluation: The hardware security evaluation (NoCSNet) relied on cycle-accurate simulation using Noxim and HotSpot. While these are industry-standard tools for architectural exploration, they are approximations of physical silicon. Real thermal behavior is influenced by packaging, ambient temperature, and manufacturing process variations that simulation cannot perfectly

model. Furthermore, the ”attacker” in our study was a simulated traffic generator. A real-world exploit would need to bypass operating system protections and generate these traffic patterns from user-space code. Validating the thermal DoS threat on physical silicon (e.g., an FPGA or a test chip) is a necessary step to confirm the severity of the vulnerability.

Scope of Root Cause Analysis: The root cause analysis system (TRACE) was developed using the TeleLogs dataset, which represents a specific set of fault scenarios in a specific network topology. The “taxonomy of faults” (C1–C8) is not exhaustive; real networks face thousands of potential issues, from fiber cuts to software bugs in the baseband unit. More fundamentally, TeleLogs is a synthetic dataset whose classes are constructed to be easily separable: any random partition of the data is drawn from the same bimodal generative process and therefore risks overfitting to that process rather than to the underlying network physics. Cross-fold validation would tighten the variance estimate on the reported 99.65% accuracy but would not detect this risk; only *out-of-sample* evaluation on operator drive-test logs from a different network/vendor and explicit *drift* testing as deployments evolve can substantiate a generalization claim. Section 6.5.1 discusses this in detail. A truly universal RCA agent would need to be trained on a much broader corpus of real network logs and would likely require continuous learning capabilities to adapt to new fault modes as they emerge.

Generalization and Domain Adaptation: A broader limitation shared across all contributions is the question of generalization to new domains. Just as Large Language Models require fine-tuning or retrieval augmentation to adapt to specific domains like telecommunications, our methods would require exploratory data analysis (EDA) and threshold re-tuning when deployed to networks with different configurations, cell densities, or vendor equipment. However, this limitation must be contextualized: unlike LLMs, which face fundamental challenges with precise numerical reasoning (as demonstrated by Mistral Large 3’s arithmetic errors even when given ground truth), our lightweight approaches can be rapidly adapted through standard ML workflows. Furthermore, the environmental footprint of domain adaptation is dramatically lower: retraining a decision tree or recalibrating heuristic thresholds requires orders of magnitude less compute than fine-tuning a multi-billion parameter language model, making our approach more sustainable for widespread deployment.

7.5 Future Work

Future research should focus on closing the loop between these domains. One promising direction is the development of ”self-defending” edge AI systems. Rather than treating security and inference as separate tasks, a unified agent could dynamically adjust the workload or routing of the Tiny-RFNet model in response to a thermal attack detected by NoCSNet. This would create a system that degrades gracefully under attack rather than failing catastrophically.

Another avenue is the expansion toward autonomous network management. The reasoning capabilities demonstrated in Chapter 6 could be extended to closed-loop 6G management systems. In this vision, diagnostic systems like TRACE would not only identify faults but also autonomously reconfigure the network to optimize for energy efficiency and security in real-time.

Perhaps the most exciting direction is the integration of TRACE as a tool within LLM-based network management agents. As demonstrated in Chapter 6, LLMs struggle with the precise numerical computations required for RCA, while TRACE excels at exactly these operations. A tool-augmented LLM architecture could combine the natural language understanding and flexible reasoning of foundation models with the numerical precision and domain expertise of TRACE. The LLM would serve as the high-level reasoning orchestrator, capable of understanding operator intent, parsing diverse log formats, and generating human-readable reports, while invoking TRACE as a trusted computational backend for all distance calculations, modulo operations, and threshold comparisons. This synergy could enable 5G/6G network agents that not only match TRACE's 99.65% accuracy on structured benchmarks but also generalize to novel fault types and provide the conversational interface that human operators require. By building TRACE-augmented LLMs, we can unlock the full potential of foundation models for telecommunications while guaranteeing the numerical correctness that network operations demand.

By connecting the high-level reasoning of hybrid ML systems with the low-level efficiency of models like RFNet, we can move closer to the vision of a fully autonomous, self-optimizing, and self-securing wireless ecosystem. Ultimately, the goal is not just faster or smarter networks in the abstract, but *trustworthy* systems that operators can deploy with confidence, knowing they will perform efficiently, resist physical attacks, and explain their decisions when things go wrong.

Appendix A

Supplementary Material

This appendix provides additional technical details, dataset schemas, and supplementary experimental results that support the main chapters of this thesis.

A.1 Dataset Specifications

A.1.1 RadioML 2018.01A Dataset

The RadioML 2018.01A dataset [18] was used for evaluating RFNet (Chapter 3) and Tiny-RFNet (Chapter 4). Table A.1 summarizes the dataset specifications.

A.1.2 TeleLogs Dataset Schema

The TeleLogs dataset [16] used for TRACE (Chapter 6) contains user-plane drive-test measurements and cell engineering parameters. Tables A.2 and A.3 detail the schema.

A.1.3 NoCSNet Dataset

The NoCSNet dataset (Chapter 5) was generated using Noxim 3.0 (cycle-accurate NoC simulator) coupled with HotSpot 6.0 (thermal simulator). Table A.4 summarizes the simulation parameters.

Table A.1: RadioML 2018.01A dataset specifications.

Property	**Value**
Total samples	2,555,904
Samples per class per SNR	4,096
Signal length	1,024 complex samples (I/Q)
Number of classes	24
SNR range	−20 dB to +30 dB (2 dB steps)
Modulation Classes	
Digital	OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-DSB-WC, AM-DSB-SC
Analog	AM-SSB-WC, AM-SSB-SC, FM, GMSK, OQPSK
Channel Model	
Fading	Rician (K=4) and Rayleigh
Impairments	Frequency offset, sample rate offset

Table A.2: TeleLogs user-plane data schema (per-second measurements).

Field	**Type**	**Description**
Timestamp	datetime	Measurement timestamp
Longitude	float	GPS longitude (degrees)
Latitude	float	GPS latitude (degrees)
Speed	float	Vehicle speed (km/h)
Serving_PCI	int	Physical Cell Identity of serving cell
SS_RSRP	float	SS-RSRP of serving cell (dBm)
SS_SINR	float	SS-SINR of serving cell (dB)
DL_Throughput	float	Downlink MAC throughput (Mbps)
Scheduled_RBs	int	Allocated Physical Resource Blocks
Neighbor_PCIs	list[int]	Up to 5 strongest neighbor PCIs
Neighbor_RSRPs	list[float]	Corresponding neighbor RSRPs (dBm)

Table A.3: TeleLogs engineering parameters schema (per-cell configuration).

Field	**Type**	**Description**
gNodeB_ID	string	Base station identifier
Cell_ID	int	Cell identifier within gNodeB
PCI	int	Physical Cell Identity (0–1007)
Longitude	float	Tower longitude (degrees)
Latitude	float	Tower latitude (degrees)
Antenna_Height	float	Antenna height above ground (m)
Azimuth	float	Antenna azimuth angle (degrees)
Mech_Downtilt	float	Mechanical downtilt (degrees)
Digital_Tilt	float	Electrical/digital tilt (degrees)
Beam_Scenario	string	Beam configuration identifier

Table A.4: NoCSNet simulation parameters.

Parameter	**Value**
Topology	6×6 mesh (36 nodes)
Routing algorithms	XY (deterministic), Odd-Even (adaptive)
Traffic patterns	Uniform Random, Transpose, Bit-Reversal
Injection rates	0.01–0.10 flits/cycle/node
Simulation cycles	1,500,000 per scenario
Flit size	32 bits
Buffer depth	4 flits per input port
Thermal Model (HotSpot)	
Technology node	22 nm
Ambient temperature	45°C
DTM threshold	85°C
Thermal sampling	Every 10,000 cycles
Attack Model	
Attacker nodes	1–4 per scenario
Attack intensity	5×–10× normal injection rate
Attack duration	100,000–500,000 cycles

A.2 Supplementary Experimental Results

A.2.1 RFNet Per-Class Accuracy

Table A.5 presents the per-class classification accuracy of RFNet128 on the RadioML 2018.01A dataset at high SNR ($\geq$ 10 dB).

Table A.5: RFNet128 per-class accuracy at SNR $\geq$ 10 dB on RadioML 2018.01A.

Class	**Accuracy (%)**	**Class**	**Accuracy (%)**
OOK	99.2	16QAM	87.4
4ASK	98.7	32QAM	82.1
8ASK	96.3	64QAM	78.5
BPSK	99.8	128QAM	71.2
QPSK	99.1	256QAM	65.8
8PSK	97.6	AM-DSB-WC	99.4
16PSK	91.2	AM-DSB-SC	98.9
32PSK	84.7	AM-SSB-WC	97.3
16APSK	89.3	AM-SSB-SC	96.8
32APSK	85.1	FM	99.7
64APSK	79.4	GMSK	98.2
128APSK	72.8	OQPSK	97.9

A.2.2 Tiny-RFNet Deployment Specifications

Table A.6 details the NVIDIA Jetson Orin Nano specifications used for Tiny-RFNet evaluation.

A.2.3 TRACE Threshold Derivation

The deterministic thresholds used in TRACE's Tier 1 heuristics (Chapter 6) were empirically derived through analysis of the TeleLogs training set. Table A.7 summarizes these thresholds and their justification.

A.3 Code and Data Availability

To support reproducibility, the following resources are publicly available:

Table A.6: NVIDIA Jetson Orin Nano Developer Kit specifications.

Component	**Specification**
GPU	NVIDIA Ampere architecture, 1024 CUDA cores 32 Tensor Cores (3rd generation)
CPU	6-core Arm Cortex-A78AE v8.2 64-bit
Memory	8 GB LPDDR5 (shared CPU/GPU)
Storage	512 GB NVMe SSD
Software Stack	
JetPack	5.1.2
CUDA	11.4
TensorRT	8.5.2
Python	3.8
TensorFlow	2.11
Power Modes	
15W mode	Max GPU/CPU clocks
7W mode	Reduced clocks for power savings

Table A.7: TRACE Tier 1 threshold derivation and justification.

Root Cause	**Threshold**	**Justification**
C2: Over-Shooting	$d \geq 1.0$ km	Urban 5G inter-site distance is 200–500 m; 1 km indicates non-dominant distant cell
C5: Ping-Pong	HOs ≥ 2	Stable connection should have ≤ 1 HO per 4-second window
C6: PCI Mod 30	Ratio > 0.5	DMRS collision present in majority of disruption window
C7: Speed	$v > 40$ km/h	Doppler spread exceeds channel coherence time for pedestrian-optimized networks
C8: Low RBs	RBs < 160	100 MHz channel has 273 PRBs; <160 indicates scheduler-limited throughput

A.3.1 Datasets

- **RadioML 2018.01A:** `https://www.deepsig.ai/datasets`
- **TeleLogs:** `https://huggingface.co/datasets/netop/TeleLogs`
- **TeleLogs-CoT:** Chain-of-thought augmented version available at: `https://huggingface.co/datasets/tecnicolaude/Telelogs-CoT`

A.3.2 Publications

The work presented in this thesis has been published as:

- **RFNet:** M. Chegini, P. Shiri, and A. Baniasadi, "RFNet: Fast and Efficient Neural Network for Modulation Classification of Radio Frequency Signals," *ITU Journal on Future and Evolving Technologies*, vol. 3, no. 2, pp. 261–272, 2022.
- **Tiny-RFNet:** M. Chegini, M. Abdollahi, A. Baniasadi, and A. Patooghy, "Tiny-RFNet: Enabling Modulation Classification of Radio Signals on Edge Systems," in *Proc. IEEE CPSNA*, 2024.
- **NoCSNet:** M. Abdollahi, M. Chegini, M. H. Hesar, S. Javadinia, A. Patooghy, and A. Baniasadi, "NoCSNet: Network-on-Chip Security Assessment Under Thermal Attacks Using Deep Neural Network," in *Proc. IEEE/ACM NoCArc Workshop*, 2024.

A.3.3 Software Requirements

Table A.8 lists the software dependencies for reproducing the experiments.

Table A.8: Software dependencies for experimental reproduction.

Component	**Version**	**Purpose**
Python	3.8+	Runtime environment
TensorFlow	2.11+	RFNet, Tiny-RFNet training
PyTorch	2.0+	NoCSNet model training
scikit-learn	1.2+	Classical ML baselines
InterpretML	0.4+	Explainable Boosting Machine
pandas	2.0+	Data processing
numpy	1.24+	Numerical operations
Noxim	3.0	NoC simulation
HotSpot	6.0	Thermal simulation
TensorRT	8.5+	Edge inference optimization

Bibliography

[1] Meisam Abdollahi, Mohammad Chegini, Mahdi Hasanzadeh Hesar, Samaneh Javadinia, Ahmad Patooghy, and Amirali Baniasadi. NoCSNet: Network-on-chip security assessment under thermal attacks using deep neural network. In *Proceedings of the 17th IEEE/ACM International Workshop on Network-on-Chip Architectures (NoCArc)*, 2024.

[2] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70–78, 2002.

[3] Subodha Charles and Prabhat Mishra. A survey of network-on-chip security attacks and countermeasures. *ACM Computing Surveys*, 54(5):1–36, 2021.

[4] Mohammad Chegini, Meisam Abdollahi, Amirali Baniasadi, and Ahmad Patooghy. Tiny-RFNet: Enabling modulation classification of radio signals on edge systems. In *2024 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*. IEEE, 2024.

[5] Mohammad Chegini, Pouya Shiri, and Amirali Baniasadi. RFNet: Fast and efficient neural network for modulation classification of radio frequency signals. *ITU Journal on Future and Evolving Technologies*, 3(2):261–272, September 2022.

[6] Yuxuan Chen, Rongpeng Li, Zhifeng Zhao, Chenghui Peng, Jianjun Wu, Ekram Hossain, and Honggang Zhang. NetGPT: A native-AI network architecture beyond provisioning personalized generative services. *IEEE Network*, 2023.

[7] William J Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (DAC)*, pages 684–689, 2001.

[8] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[9] Christopher A Harper, Mitchell A Thornton, and Eric C Larson. Automatic modulation classification with deep neural networks. *Electronics*, 12(18):3962, 2023.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. Presented at NIPS 2015 Deep Learning Workshop.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] Thien Huynh-The, Cam-Hao Hua, Quoc-Viet Pham, and Dong-Seong Kim. MCNet: An efficient CNN architecture for robust automatic modulation classification. *IEEE Communications Letters*, 24(4):811–815, 2020.

[13] Thien Huynh-The, Quoc-Viet Pham, Tien-Hoa Nguyen, Zhu Han, and Dong-Seong Kim. Automatic modulation classification: A deep architecture survey. *IEEE Access*, 9:142950–142971, 2021.

[14] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[15] Joonho Kong, Johnsyk John, Eui-Young Chung, Sung Woo Chung, and Jie Hu. On the thermal attack in instruction caches. *IEEE Transactions on Dependable and Secure Computing*, 9(2):161–172, 2010.

[16] NetOp. TeleLogs: A 5g network troubleshooting dataset. `https://huggingface.co/datasets/netop/TeleLogs`, 2025. Accessed: 2026-02-24.

[17] Timothy J O'Shea, John Corgan, and T Charles Clancy. Convolutional radio modulation recognition networks. In *International Conference on Engineering Applications of Neural Networks*, pages 213–226. Springer, 2016.

[18] Timothy J O'Shea, Tamoghna Roy, and T Charles Clancy. Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018.

[19] Sreeraj Rajendran, Wannes Meert, Domenico Giustiniano, Vincent Lenders, and Sofie Pollin. Deep learning models for wireless signal classification with distributed low-cost spectrum sensors. *IEEE Transactions on Cognitive Communications and Networking*, 4(3):433–445, 2018.

[20] Cezar Reinbrecht, Altamiro Susin, Lilian Bossuet, Georg Sigl, and Johanna Sepúlveda. Timing attack on NoC-based systems: Prime+Probe attack and NoC-based protection. *Microprocessors and Microsystems*, 52:556–565, 2017.

[21] Mohamed Sana, Nicola Piovesan, Antonio De Domenico, Yibin Kang, Haozhe Zhang, Merouane Debbah, and Fadhel Ayed. Reasoning language models for root cause analysis in 5g wireless networks. *arXiv preprint arXiv:2507.21974*, 2025.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[23] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[24] Safal Shrestha, Minwu Kim, and Keith Ross. Mathematical reasoning in large language models: Assessing logical and arithmetic errors across wide numerical ranges. *arXiv preprint arXiv:2502.08680*, 2025.

[25] Chamara Sudusinghe, Subodha Charles, and Prabhat Mishra. Denial-of-service attack detection using machine learning in network-on-chip architectures. In

Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip, pages 1–8, 2021.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erber, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[27] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[28] TecnicoLaude. TeleLogs-CoT: Chain-of-thought augmented TeleLogs dataset. `https://huggingface.co/datasets/tecnicolaude/Telelogs-CoT`, 2025. Accessed: 2026-02-24.

[29] Ning Wang, Yuan Liu, Long Ma, Yang Yang, and Huaxia Wang. Multidimensional CNN-LSTM network for automatic modulation classification. *Electronics*, 10(14):1649, 2021.

[30] Nathan E West and Timothy J O'Shea. Deep architectures for modulation recognition. In *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–6, 2017.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[32] Fuxin Zhang, Chunbo Luo, Jian Xu, Yang Luo, and Fu-Chun Zheng. Deep learning based automatic modulation recognition: Models, datasets, and challenges. *Digital Signal Processing*, 129:103650, 2022.

[33] Zhi Zhou, Xu Chen, En Li, Liangze Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[34] Hang Zou, Qiyang Zhao, Yu Tian, Lina Bariah, Faouzi Bader, Thierry Lestable, and Merouane Debbah. TelecomGPT: A framework to build telecom-specific

large language models. *IEEE Journal on Selected Areas in Communications*, 2024.