



Optimizing Deep Learning and Machine Learning Models for Real-Time Intrusion Detection in IoT Networks

by

Rezuan Islam

ID: 2429105

Supervisor: Renuga Jayakumar

Project submitted as part of the requirements for the award of
MSc Cyber Security and Digital Forensics

February 2026

DECLARATION

I, REZUAN ISLAM declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Rezuan

Date : 02-02-2026

FORM OF CONSENT

I, REZUAN ISLAM hereby consent that my Project, submitted in candidature for the M.Sc. Cybersecurity and Digital Forensics, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed Rezuan

Date: 02-02-2026

Abstract

The rapid growth of Internet of Things (IoT) deployments has expanded the network attack surface and increased the operational need for intrusion detection systems (IDS) that remain accurate under multi-class class imbalance while also being computationally feasible for near-real-time use. Using the CICIOT2023 benchmark dataset, this dissertation develops a reproducible end-to-end experimental pipeline and evaluates representative families of intrusion detection models, including the traditional machine learning, deep learning, ensemble learning, and proposed hybrid approaches. Five CICIOT2023 CSV partitions were ingested with a robust Drive-to-local caching strategy, then controlled via per-class capping ($\text{cap}=3,225$) to manage scale. After rare-class filtering (minimum 200 samples/class), the final experimental dataset comprised 73,211 samples, 50 features, and 27 classes, split stratified into 51,539 train, 7,029 validation, and 14,643 test instances. Across 27 evaluated models, the Proposed Voting Ensemble achieved the strongest overall detection quality (Accuracy=0.9528, Macro-F1=0.9441, MCC=0.9508) with a training time of 224.0 s, marginally exceeding strong ensemble baselines such as Bagging (DT) (Macro-F1=0.9417, 95.8 s) and XGBoost (Macro-F1=0.9416, 52.9 s). Notably, runtime-performance trade-offs were material: Gradient Boosting delivered competitive Macro-F1 (0.9391) but incurred substantially higher training time (2835.6 s), while LightGBM offered a favorable efficiency profile (Macro-F1=0.9360, 40.6 s). Deep learning baselines underperformed leading ensembles (best DL: Wide MLP Macro-F1=0.7666) and one attention-based model collapsed (Macro-F1=0.0031), indicating sensitivity of generic neural architectures to tabular IoT traffic representations under imbalance. Knowledge distillation did not improve detection quality versus ensembles in this setting (KD student Macro-F1=0.7247), but remains relevant for future edge-focused compression studies. The dissertation's contributions are (i) a transparent, imbalance-aware benchmark emphasizing Macro-F1 and MCC, (ii) validated ensemble and tuning strategies (Optuna-guided HistGB, soft voting), and (iii) a reproducible experimental workflow with saved artefacts and figures.

Keywords: IoT intrusion detection, CICIOT2023, ensemble learning, knowledge distillation.

Table of Contents

I.	TITLE PAGE	
II.	DECLARATION AND FORM OF CONSENT	
III.	ABSTRACT	
IV.	CONTENTS PAGE	
	○ List of Tables and List of Figures.....	8
1.	CHAPTER 1: INTRODUCTION.....	9
	1.1. Background and context.....	9
	1.2. Problem statement.....	9
	1.3. Research Aim and objectives.....	10
	1.4. Research questions.....	11
	1.5. Significance of the study.....	12
	1.6. Dissertation structure.....	12
2.	CHAPTER 2: LITERATURE REVIEW.....	14
	2.1. Literature Search and Selection Criteria.....	14
	2.2. IoT intrusion detection context and threat characteristics.....	15
	2.3. Datasets for IoT IDS and methodological pitfalls.....	16
	2.4. Feature representation and preprocessing for tabular IoT traffic.....	21
	2.5. Traditional machine learning for IoT IDS.....	22
	2.6. Ensemble learning: bagging, random forests, boosting, and voting.....	23
	2.7. Deep learning for IoT intrusion detection.....	25
	2.8. Knowledge distillation and compression for practical IoT IDS.....	26
	2.9. Evaluation practice: metrics, imbalance, and comparability.....	27
	2.10. Synthesis: what the literature implies for this research design.....	29
3.	CHAPTER 3: RESEARCH METHODOLOGY.....	31
	3.1. Research philosophy.....	31
	3.2. Research approach.....	31
	3.3. Research strategy.....	32
	3.4. Variables and hypotheses.....	32
	3.5. Data and tools.....	35
	3.5.1. Dataset selection and justification.....	35
	3.5.2. Tooling and compute environment.....	36
	3.5.3. Mathematical formalization of the experimental protocol.....	37
	3.6. Validity, reliability, and bias control.....	38

4. CHAPTER 4: DESIGN PROCESS (EXPERIMENT DESIGN).....	42
4.1. Overall Pipeline Architecture.....	42
4.2. Dataset Ingestion Design.....	44
4.2.1.Dataset choice and source.....	44
4.2.2.Partition selection, storage strategy, and determinism.....	44
4.2.3.Sampling, row budget, and class coverage controls.....	44
4.2.4.Training-set balancing for ensemble construction.....	44
4.3. Feature Handling Design.....	45
4.3.1.Baseline feature set and data types.....	45
4.3.2.Missing values and leakage controls.....	45
4.3.3.Scaling rationale by model family.....	45
4.4. Train/Validation/Test Design.....	47
4.4.1.Stratified splitting strategy.....	47
4.4.2.Split proportions and validation usage.....	47
4.4.3.Early stopping as a design control (deep learning).....	47
4.5. Model Family Design.....	47
4.5.1.Traditional ML baseline set.....	47
4.5.2.Ensemble ML set.....	48
4.5.3.Deep learning set.....	48
4.5.4.Proposed components.....	48
4.6. Evaluation Design.....	49
4.6.1.Metric selection and definitions.....	49
4.6.2.Runtime measurement approach.....	49
4.6.3.Planned figures and reporting artefacts.....	50
5. CHAPTER 5: IMPLEMENTATION (EXPERIMENT).....	51
5.1. Environment setup and dependency management.....	51
5.2. Data pipeline implementation.....	52
5.2.1.Drive mount, robust copy logic, and local caching.....	52
5.2.2.Loading logic, label normalization, and capped sampling.....	52
5.2.3.Rare-class filtering, encoding, scaling, and splitting.....	53
5.3. Model implementations.....	54
5.3.1.Traditional ML models.....	55
5.3.2.Ensemble ML models.....	55
5.3.3.Deep learning models.....	56

5.3.4. Proposed solution components (teacher, tuned HistGB, voting ensemble, KD student).....	56
5.4. Artefact saving and reproducibility.....	59
5.4.1. Saved models and preprocessors.....	59
5.4.2. Results table and figures.....	59
5.4.3. Reproducibility scope.....	60
5.4.4. Confusion matrix artefact generation and interpretation.....	60
5.5. Implementation challenges and fixes.....	61
6. CHAPTER 6: EVALUATION OF RESULTS.....	63
6.1. Summary of the Experiment.....	63
6.2. Overall Benchmark Results and Ranked Performance.....	63
6.3. Visual Analytics and Figure-Based Interpretation.....	65
6.3.1. Cross-model metric profiles (Accuracy, Macro-F1, MCC).....	65
6.3.2. Macro-F1 ranking clarity: leaderboard view.....	66
6.3.3. Detailed metrics heatmap.....	67
6.3.4. Winner Analysis: Proposed Voting Ensemble.....	68
6.3.5. Best in Class Comparison.....	68
6.3.6. Model Tier List (Based on Composite Score).....	69
6.3.7. Metric Consistency by Algorithm Category.....	70
6.3.8. Performance-time trade-off: efficiency frontier.....	70
6.3.9. Pareto optimization: best achievable for a time budget.....	72
6.4. Interpretation of Findings.....	73
6.4.1. Why the Best Models Performed Well.....	73
6.4.2. Runtime Performance Trade-offs and Practical Winners.....	73
6.4.3. Understanding Weak and Failed Models.....	74
6.4.4. Interpreting High ROC-AUC Values.....	75
6.5. Statistical Robustness and Evaluation Limitations.....	76
6.6. Comparison to Recent Literature.....	76
7. CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS.....	78
7.1. Restatement of Aim and Objectives, and What Was Achieved.....	78
7.2. Key Findings.....	78
7.3. Contributions.....	79
7.4. Recommendations for Practitioners.....	80
7.5. Limitations.....	81
7.6. Future Work.....	81

8. CHAPTER 8: REFLECTION.....	82
9. REFERENCES.....	84
10. Appendix.....	88
10.1. Source Code	
10.2. Benchmark Result	
10.3. Tooling and compute environment	

List of Tables and List of Figures

Table 2.1: Dataset Comparison

Table 6.1: summarizes the top 10 models from the final ranked results

Figure 4.1: Pipeline Architecture of the Experimental Benchmarking Framework.

Figure 4.2: Data Cleaning and Preprocessing Technique Pipeline

Figure 5.1: Proposed Voting Ensemble Model

Figure 5.2: Confusion Matrix

Figure 6.1: Accuracy, Macro F1 and MCC comparison

Figure 6.2: Model Leaderboard - Macro F1 Score

Figure 6.3: Detailed metrics heatmap

Figure 6.4: Winner analysis and cost-of-victory

Figure 6.5: Best-in-class radar comparison

Figure 6.6: Tier list by composite score

Figure 6.7: Metric consistency by algorithm category

Figure 6.8: Efficiency Frontier - Performance vs Time (Log Scale)

Figure 6.9: Pareto Optimization

CHAPTER 1: INTRODUCTION

1.1 Background and context

The Internet of Things (IoT) has evolved into a pervasive computing paradigm in which heterogeneous devices continuously generate network traffic and operational telemetry. This scale and heterogeneity create a broad and dynamic attack surface: endpoints are often resource-constrained, deployed at the edge, and run diverse firmware stacks, meanwhile, attackers can exploit weak authentication, exposed services, and supply-chain vulnerabilities to establish persistence or to weaponize devices for distributed denial-of-service (DDoS) and lateral movement. In response, intrusion detection systems (IDS) remain a core defensive control for monitoring traffic and identifying malicious activity, particularly in environments where preventive controls alone are insufficient due to device constraints or operational complexity.

Recent IoT-IDS research increasingly adopts Machine Learning (ML) and Deep Learning (DL) techniques to capture patterns in high-dimensional traffic features and to improve detection of novel or evolving attacks. Contemporary surveys emphasize, however, that practical IoT IDS designs still face recurring difficulties: data imbalance across attack classes, inconsistent evaluation methodology, limited reproducibility, and computational overheads that undermine real-time applicability in constrained settings [1]. In addition, the IoT threat environment exhibits non-stationarity: traffic distributions can shift due to device churn, firmware updates, and changes in attacker behavior, introducing concept drift that degrades static models over time [2].

To support rigorous and comparable evaluation, security research increasingly relies on public datasets designed to represent realistic IoT traffic and attack diversity. CICIoT2023 is a prominent example, providing a large-scale IoT dataset derived from a multi-device topology and multiple attack classes, enabling benchmarking of ML/DL approaches under conditions closer to operational IoT environments [3].

1.2 Problem statement

Despite rapid growth in ML/DL-based IoT IDS proposals, there remains a persistent gap between research performance claims and deployment-ready evidence. The key issues motivating this dissertation are:

First, evaluation inconsistency remains common. Different studies select different metrics and averaging schemes, sometimes without justification, creating misleading comparisons, particularly under class imbalance where overall accuracy can remain high while minority-class detection degrades. Metric-selection research has highlighted how evaluation choices can materially affect conclusions, reinforcing the need to justify metrics and report them transparently [4].

Second, imbalance and multiclass complexity are fundamental in IoT IDS data. Many IoT datasets contain multiple attack families with varying prevalence. Metrics such as Macro-F1 (treating classes equally) and the Matthews Correlation Coefficient (MCC) (capturing correlation between predictions and ground truth) are therefore more informative than accuracy alone when class distributions are skewed [4].

Third, operational constraints create a performance–cost tension. High-capacity models may increase detection performance but may be impractical for real-time or near-real-time deployment due to training and inference costs. Surveys emphasize computational efficiency as a practical requirement, not a secondary concern [1].

Fourth, non-stationarity and concept drift can reduce IDS reliability over time, particularly for static models trained on snapshot datasets. This makes it necessary to interpret benchmark results with appropriate caution and to articulate future work towards drift-aware evaluation [2].

1.3 Research aim and objectives

Research Aim:

To critically analyze and empirically evaluate traditional machine learning, ensemble learning, deep learning, and a proposed voting-ensemble approach for real-time-oriented IoT intrusion detection, using a reproducible benchmark pipeline with metrics suitable for imbalanced multiclass data. The goal is to design a new hybrid DL model with more detection accuracy and capable of detecting both known cyber-attacks and previously unseen attacks (zero-day) in IoT network traffic.

Research Objectives:

1. To critically analyze recent (primarily 2020–2026) peer-reviewed literature on IoT IDS using ML/DL, focusing on datasets, evaluation practice, imbalance handling, and runtime considerations, using a PRISMA-guided systematic literature review protocol.
2. To design and develop a reproducible experimental pipeline for dataset ingestion, preprocessing, stratified splitting, and model training across defined model families, with controlled sampling/capping to manage scale and ensure repeatability.
3. To evaluate statistically and comparatively the benchmarked models using Macro-F1 and MCC as primary metrics (with accuracy and runtime as supporting measures), and to determine whether the proposed approach offers a justifiable improvement relative to baseline ML, ensemble, and DL alternatives under the same data and split conditions.

1.4 Research questions

To operationalize the aim and objectives, the dissertation addresses the following research questions (RQ):

RQ1: How can deep learning models or ensemble models be optimized to achieve real-time intrusion detection on resource-constrained IoT devices?

RQ2: Which optimization techniques (pruning, quantization, knowledge distillation, voting-ensemble strategy, lightweight architectures) offer the best trade-off between accuracy, latency, and model size?

RQ3: What are the run-time performance trade-offs between the best-performing models, and how do these trade-offs affect the plausibility of real-time or near-real-time IDS usage?

RQ4: What limitations arise from dataset sampling, class filtering, and single-split benchmarking, and how should these constraints shape interpretation and recommendations for future research?

1.5 Significance of the study

This dissertation has practical and scholarly significance for IoT security engineering and IDS research. Practically, it emphasizes a defensible evaluation approach that aligns with how IDS quality should be judged in imbalanced multiclass environments, prioritizing Macro-F1 and MCC rather than relying on accuracy alone. It also treats computational overhead as a first-class concern by explicitly measuring training-time differences, supporting implementers who must balance detection quality with operational costs.

Scholarly, the dissertation contributes to an integrated workflow that combines

- (i) A PRISMA-guided synthesis of evaluation pitfalls and best practice.
- (ii) A reproducible benchmark on a modern IoT dataset.
- (iii) A comparative analysis that is transparent about sampling controls and interpretive limitations.

This combination is designed to reduce ambiguity in claims and to strengthen reproducibility and comparability across future IoT IDS studies.

1.6 Dissertation structure

Chapter 2 presents a PRISMA-guided systematic literature review of IoT IDS research using ML/DL, synthesizing evidence on datasets, metrics, imbalance handling, runtime constraints, and methodological gaps.

Chapter 3 defines the research methodology, including philosophical position, experimental strategy, variables, validity controls, and ethics.

Chapter 4 specifies the end-to-end experimental design, including dataset ingestion choices, preprocessing, model families, and evaluation plan.

Chapter 5 details the practical implementation of the pipeline and model training based on the provided experiment log and artefacts.

Chapter 6 reports and interprets the benchmark results using the final results table and accompanying visualizations, with explicit discussion of performance runtime trade-offs and failure cases.

Chapter 7 concludes by mapping results to objectives, offering practitioner recommendations, limitations, and future work directions.

Chapter 8 provides a concise academic reflection on process, planning, and learning.

CHAPTER 2: LITERATURE REVIEW

Intrusion Detection Systems (IDS) remain a core control for identifying malicious activity in networked environments. However, IoT deployments introduce additional constraints that complicate both detection performance and operational viability. These constraints include heterogeneous device capabilities and protocols, high-volume telemetry, non-stationary traffic patterns, severe class imbalance (few attacks vs. much benign traffic, or vice versa depending on capture design), and resource limitations that make heavyweight models difficult to sustain in real time. Contemporary surveys emphasize that, while learning-based IDS can improve detection sensitivity compared to static signature approaches, research outcomes are often difficult to compare because studies vary in datasets, feature handling, split methodology, imbalance treatment, and metric selection. This leads to a persistent reproducibility and comparability gap that directly motivates benchmark-style research designs [1].

Accordingly, this chapter critically reviews

- (i) IoT IDS datasets and their methodological pitfalls.
- (ii) Modelling families used for tabular network telemetry - traditional ML, ensemble ML, and deep learning.
- (iii) The relevance of model compression and knowledge distillation for constrained or real-time scenarios.
- (iv) Evaluation practice, with emphasis on metric selection under imbalance and the importance of leakage-resistant validation.

The chapter closes by synthesizing a set of research gaps that justify a comparative benchmark and the investigation of tuned and ensemble-based approaches aligned with real-time constraints.

2.1 Literature Search and Selection Criteria

A systematic and thorough search was conducted to gather the most relevant research on the field of machine learning and deep learning algorithms for the intrusion detection system on IoT. The search was conducted on the following databases: IEEE Xplore, IEEE Access, Google Scholar, ResearchGate, Springer, ScienceDirect, ACM Digital Library and Scopus. The selection of these

databases was based on their vast collections of peer-reviewed articles and high impact conference papers in the disciplines of engineering, cybersecurity, and computer sciences.

Search Keywords Used:

- (“intrusion detection” OR “anomaly detection” OR “IDS”) AND (“IoT” OR “IIoT” OR “smart home”) AND (“machine learning” OR “deep learning” OR “ensemble”)
- “intrusion detection system” AND “deep learning” AND “IoT”
- “deep learning” OR “Machine Learning” AND “IoT security” AND “real-time ids detection”
- “zero-day attack detection” AND “IoT” AND “deep learning”
- "lightweight deep learning" AND "intrusion detection" AND "IoT networks"

Inclusion Criteria:

1. Peer-reviewed articles and high-impact conference papers published between 2020 and 2026 and in English.
2. Studies focusing on machine learning deep learning for intrusion detection in IoT networks.
3. Articles evaluating models on IoT-specific datasets, such as CICIoT2023, Bot-IoT and UNSW-NB15.

Exclusion Criteria:

1. Non-peer-reviewed sources and non-English (For example, whitepapers, blogs).
2. Studies published before 2020 and low cited papers.
3. Articles not specifically addressing IDS or lacking empirical results.

By following those criteria initially found 1912 search result. Out of those top cited 54 research articles were included to review to conduct this research.

2.2 IoT intrusion detection context and threat characteristics

IoT networks aggregate traffic from sensors, cameras, appliances, gateways, and mobile endpoints, frequently operating in always-on conditions and often lacking robust endpoint hardening. This context increases exposure to volumetric and reconnaissance attacks as well as device-targeted

exploits. Modern IoT datasets and surveys typically characterize attack activity into groups such as DoS/DDoS, reconnaissance/scanning, spoofing, brute-force attempts, web-layer attacks, and botnet activity. The CICIoT2023 benchmark, for example, explicitly organizes attacks into multiple categories executed within an IoT topology that includes many devices and attack types, reinforcing the practical requirement that IDS models generalize across heterogeneous behaviors rather than overfitting to a narrow signature. [3].

A key implication for modelling is that IoT intrusion detection is rarely a binary problem in realistic benchmarks. Instead, it is commonly multi-class (benign plus multiple attack classes) or multi-label (concurrent behaviors), with strong skew across labels. This pushes evaluation away from single-threshold accuracy and towards metrics that properly penalize failures on minority classes. Surveys of M.M.Rahman et al. [1] also stress that detection utility is inseparable from deployment realities: false positives create operational overload, while detection latency (or high compute cost) can prevent timely mitigation in high-throughput IoT gateways.

2.3 Datasets for IoT IDS and methodological pitfalls

2.3.1 Representative datasets and evolution toward realistic IoT telemetry

Dataset choice has become a methodological determinant of validity in IoT intrusion detection research, because the empirical properties of telemetry - device diversity, protocol mix, background behavior, and attack execution realism - shape what a model is actually learning. A clear trajectory can be observed in the literature: from modernized enterprise-like benchmarks (e.g., UNSW-NB15) toward IoT-first botnet datasets (BoT-IoT), then toward telemetry-rich IIoT corpora (TON_IoT), and most recently toward large-scale, multi-attack IoT topologies designed explicitly for real-time benchmarking (CICIoT2023) [3]. This evolution is not merely chronological, it reflects shifting expectations around representativeness, multi-class realism, and reproducible benchmarking.

UNSW-NB15: a modern baseline with persistent methodological value

UNSW-NB15 remains widely used because it offers a structured attack taxonomy and a comparatively standardized feature representation that supports rapid algorithmic comparison,

even though it is not IoT-specific [5]. Recent work continues to use UNSW-NB15 to investigate methodological enhancements that generalize beyond any one environment. For example, IDS-INT proposes transformer-based transfer learning for imbalanced traffic and evaluates on UNSW-NB15 among other benchmarks, explicitly framing imbalance and minority-attack detection as central design constraints [6]. Similarly, Yin et al. [7] proposes a hybrid feature selection method (IGRF-RFE) for MLP-based IDS on UNSW-NB15, illustrating that feature selection remains a high-leverage intervention - often improving effectiveness and efficiency simultaneously - particularly in tabular flow settings. Complementary analyses in Algorithms further demonstrate that performance gains reported in UNSW-NB15 studies frequently hinge on preprocessing, feature choice, and evaluation framing rather than architectural novelty alone, which reinforces the importance of transparent pipelines when using this dataset as a baseline [8].

Academically, the main limitation of UNSW-NB15 in an IoT thesis is ecological validity: IoT networks exhibit different traffic periodicities, device-to-device chatter, and protocol/application mixes than enterprise networks. As a result, UNSW-NB15 is best treated as a method validation dataset (e.g., confirming feature handling, class-imbalance mitigation, calibration or thresholding strategies) rather than as a proxy for realistic IoT telemetry.

BoT-IoT: shifting evaluation toward IoT botnet realism and forensic behavior

BoT-IoT was a pivotal step in aligning datasets with IoT threat realities by foregrounding botnet-driven and IoT-relevant attack patterns [9]. A key research trend in the post-2020 period is narrowing the modelling target from generic attack vs benign discrimination to attack-type granularity and category-specific modelling. Leevy et al. [10] exemplifies this by using BoT-IoT to predict information theft-related behaviors and evaluating ensemble feature selection strategies -an approach that treats IDS design as both a detection and a feature-engineering problem under IoT-specific distributions. More applied IDS studies continue to adopt BoT-IoT for benchmarking multi-stage or hybrid detectors and for evaluating imbalance-handling strategies in high-volume IoT flow data [11].

From an academic research perspective, BoT-IoT is valuable because it stresses models under traffic regimes that include scanning and flooding behaviors common in IoT botnets, but it can also over-represent certain attack dynamics relative to benign background diversity. This makes it

excellent for studying algorithmic sensitivity to high-volume malicious patterns, yet it requires careful discussion of generalizability to heterogeneous everyday IoT networks.

TON_IoT: from network flows to multi-source telemetry and distributed realism

TON_IoT represents a conceptual expansion of what IoT intrusion data should include: not only network traffic but telemetry sources that more closely match how IIoT environments are monitored in practice [12]. The broader TON_IoT research program has been linked to distributed architectures spanning edge/fog/cloud, reflecting the reality that IoT security instrumentation is itself distributed and that IDS can be deployed at multiple layers [13]. Recent studies leverage TON_IoT in comparative work precisely because it supports heterogeneous evaluation and exposes where single-modality detectors fail. For instance, Applied Sciences’ multi-dataset deep learning analysis uses TON_IoT alongside UNSW-NB15 and Edge-IIoT, highlighting that the same architecture can behave differently across datasets due to telemetry characteristics and label distributions [14].

A particularly relevant methodological development is the push for standardized, explainable feature sets across IoT datasets. Alani and Miri [15] propose a universal feature set spanning multiple IoT IDS datasets including TON_IoT and BoT-IoT, demonstrating that feature harmonization can support comparability and improve the interpretability of model behavior across benchmarks. This theme directly informs thesis-level evaluation practice: if different studies use different feature subsets and extraction pipelines, cross-paper performance comparison becomes epistemically weak, even when the dataset name is identical.

CICIoT2023: large-scale IoT topology, richer attack taxonomy, and multi-class imbalance realism

CICIoT2023 advances dataset realism along two dimensions that strongly affect modern ML evaluation: a larger device topology and a broader, explicitly categorized attack set intended for real-time benchmark development [3]. This breadth increases multi-class difficulty and amplifies imbalance challenges - conditions under which macro-averaged metrics and robust evaluation design become essential rather than optional.

Recent research using CICIoT2023 increasingly focuses on addressing imbalance, improving feature representation, and improving architecture sensitivity to tabular IoT flows. A Springer

LNNS conference [16] contribution proposes a multiplicative attention-based deep learning framework evaluated on CICIOT2023 and explicitly frames the dataset as challenging due to imbalanced categories and sub-categories, thereby motivating attention mechanisms for feature importance learning. In parallel, empirical studies in Informatica report benchmarking of hyperparameter-tuned machine learning and deep learning models on CICIOT2023 [17], illustrating a broader trend: performance claims are increasingly tied to tuning discipline (search strategy, objective metric selection, validation protocol) rather than to algorithm family alone. Importantly for academic writing, these works collectively reinforce that CICIOT2023 is not simply another dataset, it is a benchmark whose utility depends on transparent decisions about class filtering, sampling, and representation choice - especially when multi-class evaluation is performed.

Cross-dataset evaluation and the modern expectation of generalization

A notable shift in the 2022-2025 period is that stronger papers increasingly treat dataset performance as conditional and emphasize cross-dataset or multi-dataset comparison to test generalization. For example, an Elsevier study [18] evaluates deep learning models against multiple datasets including BoT-IoT and ToN-IoT, reflecting the emerging view that within-dataset accuracy can overestimate real-world readiness if model behavior collapses under distribution shift. This generalization-first posture has direct implications for a research using CICIOT2023: results should be framed as valid within the chosen benchmark and pipeline, while acknowledging that operational performance under drift or cross-domain conditions requires additional testing.

Dataset	Year/ Domain	Format	Approx. (instances / records)	Attributes (features)	Labels / classes (typical)	Key strengths for research	Common pitfalls / comparability risks
UNSW-NB15 [41]	2015 / Enterprise like NIDS baseline	PCAP + CSV feature files	2,540,044 records (CSV), raw capture reported as ~100 GB PCAP	49 features (incl. label, often referenced as 42–49 depending on subset/clea ning)	10-class (Normal + 9 attack categories) and binary label also available	Stable baseline for algorithm comparisons and clear taxonomy	Not IoT-telemetry, results sensitive to feature subset (e.g., 10% split has different feature count), class imbalance and duplicate- handling vary across papers
BoT-IoT [40]	2019 / IoT botnet- focused	CSV (flow/ derived features)	73,360,900 instances (full dataset)	43 features (plus labels, some papers analyze 5% subset ≈3.67M)	Binary label (benign/attac k) plus multi- class variant commonly grouped as Normal + {DoS, DDoS, Recon, Theft}	Strong for botnet-style IoT threat modelling, stresses models under high- volume malicious patterns	Background benign diversity may be less representative of everyday IoT, many studies use small % subsets (not directly comparable), label grouping differs across papers
ToN-IoT (Network + telemetry family) [39]	2020– 2021 / IoT & IIoT (heteroge neous telemetry)	Multiple CSV subsets (Network, Win7/W in10 OS, IoT telemetry)	Network dataset commonly cited around ~22.3M instances, additional OS/IoT subsets are much smaller (tens of thousands)	Network CSV often reported as ~43 features (+ label/type columns, some works describe 42–45 due to filtering)	9 attack categories (+ normal) commonly reported for the network set (e.g., DoS/DDoS, scanning, backdoor, ransomware, MITM, injection/XSS , password)	Enables research beyond flows (telemetry), useful for studying cross-source robustness and realistic IIoT monitoring assumptions	ToN-IoT sometimes refers to different subsets (Network vs OS vs combined), feature counts and record counts vary by subset/version, comparability requires stating exactly which subset and preprocessing were used
CICIoT 2023 [3]	2023 / Modern IoT benchmark	PCAP + CSV, reported as 169 files in mixed formats in recent usage papers	Literature reports full- scale on the order of tens of millions of flow instances (46,217,820 reported in a recent peer- reviewed study)	46 attributes in many studies, some count features + label as 47 (counting conventions differ)	Fine-grained: 33 attack types + benign (~34 classes) and/or 7 higher-level categories + benign (8 classes) depending on mapping	High device diversity + broad attack taxonomy, designed for modern IoT benchmarking, highlights imbalance issues explicitly	Strong multi-class imbalance, studies frequently use subsets (making direct cross-paper comparison weak unless subset rules are explicit), feature-count conventions vary (46 vs 47)

Table 2.1: Dataset Comparison

2.3.2 Data leakage, split design, and non-stationarity

Beyond dataset selection, how data are split can invalidate results. Leakage arises when information from the test distribution influences training, often inadvertently via preprocessing done on the full dataset (e.g., scaling/normalization fitted globally), duplicated flows across splits, or temporal correlations where adjacent windows appear in both train and test. Recent empirical work demonstrates that leakage-resistant methodology can materially change model rankings and apparent performance, highlighting that a strong accuracy may simply reflect optimistic evaluation rather than genuine generalization. [19].

IoT environments also exhibit non-stationarity: device populations change, firmware updates alter traffic patterns, and adversaries adapt. A recent comprehensive survey emphasizes concept drift and feature dynamics in IDS, arguing that drift-aware evaluation and adaptation mechanisms are necessary to avoid benchmark overconfidence that fails under operational change. [20]. The practical consequence is that benchmark experiments should (at minimum) employ robust stratified splits, fixed seeds, and leakage checks, and should interpret single-split results as indicative rather than definitive - particularly when aiming for real-time intrusion detection.

2.4 Feature representation and preprocessing for tabular IoT traffic

Most public IoT IDS benchmarks provide engineered, tabular flow features (e.g., packet/byte counts, inter-arrival statistics, header-derived attributes, and time-window aggregates) rather than raw packets. This representation is computationally attractive and aligns with gateway-level deployment, but it imposes modelling constraints. First, tabular features may compress temporal and sequential signals, reducing the natural advantage of sequence models unless explicit windowing or sequence construction is performed. Second, feature distributions can be heavy-tailed, sparse, and heterogeneous across protocols and devices, complicating scaling and distance-based models. Third, categorical attributes (when included) require encoding that can inadvertently leak label information if not handled carefully.

Surveys of IoT IDS methods commonly describe feature extraction as a decisive stage that determines model viability, particularly for resource-limited deployments. They also observe that many published studies under-report preprocessing details (e.g., whether scaling was fit on training

only, whether rare-class filtering was applied, how missing values were treated) [1]. This matters because different model families respond differently to scaling and feature distributions, linear models may require standardization, tree ensembles generally tolerate monotonic transforms but can be affected by noisy, redundant features, and neural networks can become unstable without careful normalization and regularization.

From a benchmark design perspective, a defensible preprocessing approach typically includes:

- (i) Deterministic, training-only fit for scalers/encoders.
- (ii) Explicit handling of rare classes to avoid degenerate stratification.
- (iii) Documentation of feature coercion and missing-value policy.

These are not implementation details but rather essential methodological controls that determine the credibility of comparisons. [19].

2.5 Traditional machine learning for IoT IDS

Traditional ML methods remain widely used for tabular IDS because they are comparatively fast, interpretable, and often competitive. IoT IDS surveys typically group these methods into linear classifiers (logistic regression, ridge regression, perceptron/SGD variants), probabilistic classifiers (naïve Bayes), and distance-based methods (k-nearest neighbors). [1].

Linear and margin-based models:

Linear classifiers can be attractive for real-time deployment because inference is cheap and model size is small. Their limitation is representational: unless feature engineering captures the relevant non-linear interactions, linear decision boundaries may underfit complex attack behaviors. In multi-class IoT settings, this weakness can surface as degraded minority-class recall and unstable macro-averaged metrics, particularly under drift or when attack behaviors overlap in feature space. [20].

Probabilistic methods:

Naive Bayes variants can be effective baselines with low latency, but their conditional independence assumption is often violated in engineered flow features. Nonetheless, strong

performance occasionally appears when features are curated, discretized, or when the dataset structure aligns with the model’s assumptions. Surveys note, however, that naive Bayes results can be brittle across datasets because correlations between features differ significantly by traffic capture design. [1].

Distance-based methods:

KNN and related approaches can capture local structure but scale poorly with dataset size and feature dimensionality. In IoT IDS, where dataset partitions can be large and concept drift can shift neighborhood structure, KNN may become operationally impractical unless approximate indexing is used - an added complexity that many IDS benchmark studies omit. [20].

Overall, traditional ML models are best understood as strong baselines that test whether complex modelling is warranted. When they are outperformed by ensembles or tuned methods, it is typically due to their limited ability to capture non-linear interactions and heterogeneous class boundaries in high-dimensional tabular telemetry.

2.6 Ensemble learning: bagging, random forests, boosting, and voting

Ensemble methods are central to tabular IDS because they often deliver strong generalization without the training instability of deep networks on structured features. The literature provides both foundational theory and modern security-specific evidence for their effectiveness.

2.6.1 Bagging and random forests

Bagging reduces variance by training multiple base learners on bootstrap samples and aggregating predictions, providing a principled mechanism to improve stability over a single high-variance learner such as a decision tree. [21]. Random forests extend this logic with feature sub-sampling, further decorrelating trees and improving generalization. [22].

In IoT IDS contexts, ensemble-tree approaches are frequently reported as strong performers because they handle non-linear interactions, tolerate mixed feature scales, and provide useful feature-importance signals. Moreover, they can be trained and deployed with relatively predictable compute requirements compared with deep architectures, particularly when data are tabular and engineered. Evidence from recent IDS research using ensemble trees combined with explanation techniques (e.g., SHAP) supports the view that tree ensembles can simultaneously deliver high

detection performance and interpretable decision support - an important operational property when security analysts must validate alerts. [23].

2.6.2 Gradient boosting and modern boosting systems (including XGBoost)

Boosting methods build an additive model that sequentially corrects errors, often achieving high predictive accuracy in structured data. The gradient boosting machine formalizes this as functional gradient descent, providing a theoretical basis for why boosting can fit complex decision boundaries. [24]. Modern scalable boosting systems, notably XGBoost, operationalize boosting with system-level optimization and regularization, enabling high performance at scale. [25].

In IDS, boosting is attractive because attack classes may form irregular manifolds within feature space and because boosting can model high-order interactions without explicit manual feature crosses. Surveys and benchmark discussions [1] also highlight, however, that boosting performance can be sensitive to hyperparameters (depth, learning rate, subsampling, regularization), making out-of-the-box comparisons potentially misleading unless tuning is applied consistently across model families.

This sensitivity is not merely academic: in multi-class IDS under imbalance, boosting can overfit dominant classes unless loss weighting, early stopping, or carefully selected sampling strategies are used. These effects reinforce the need to treat tuning as part of the experimental design rather than a post-hoc improvement.

2.6.3 Voting ensembles and model combination logic

Voting ensembles aggregate multiple classifiers (hard or soft voting) and are commonly used to improve robustness when models capture complementary aspects of the data. In IoT IDS, this strategy can mitigate the risk that one learner’s inductive bias (e.g., high recall but low precision) dominates operational outcomes. Surveys describing best practice patterns frequently recommend hybridization, combining models or combining feature extractors, particularly when datasets contain heterogeneous attack behaviors. [1].

The literature suggests that voting can be especially valuable when top-performing models have similar mean performance but different error profiles (e.g., one model better on scanning/recon classes, another better on volumetric DoS-like patterns). In such cases, soft voting can improve

macro-averaged metrics by lifting minority-class recall without materially degrading majority-class detection. While these gains are not guaranteed, they are theoretically plausible in heterogeneous multi-class settings and justify experimental evaluation of voting-based proposals.

2.7 Deep learning for IoT intrusion detection

Deep learning has been heavily studied for IDS, including IoT contexts, motivated by the promise of learning hierarchical representations and capturing complex dependencies. However, the evidence for consistent superiority over well-tuned ensembles in tabular flow-feature settings is mixed.

A broad review of deep learning approaches for IoT security highlights that deep models are used across IoT security scenarios, but outcomes depend strongly on data modality (tabular vs. sequential vs. raw signals), architecture choice, and training regime [26]. In IDS specifically, deep architectures often fall into the following families:

2.7.1 Multilayer Perceptrons (MLPs) for tabular features

MLPs are natural candidates for tabular flow features and are widely used in IoT IDS studies. Their advantage is flexibility, their disadvantage is that they can underperform tree ensembles on structured, non-smooth tabular distributions unless carefully regularized and tuned (depth/width, dropout, batch normalization, learning rate schedules). In addition, MLPs can become sensitive to class imbalance and label noise. The literature notes that deep models may require more careful calibration than ensembles to avoid unstable training and misleading confidence outputs - an operational concern in security response [1], [26].

2.7.2 CNNs and LSTM/RNN models for temporal structure

CNN-1D and LSTM models are often introduced to capture local and sequential dependencies, respectively. Their value depends on whether the data pipeline preserves temporal order and whether sequences are meaningfully constructed (e.g., packet sequences or time-windowed flow sequences). If the input is already heavily aggregated into per-flow or per-window statistics, the architectural advantage of sequence modelling can be reduced. Concept drift and feature dynamics

further complicate deployment because temporal patterns learned from one capture period may not persist. [20], [26].

Recent IoT IDS work continues to propose deep architectures aimed at scalability and efficiency, reflecting the field’s acknowledgement that raw deep models can be too heavy for edge deployment unless designed carefully. For example, recent journal work proposes scalable and efficient deep-learning-based IDS designs for IoT networks, underscoring the recurring theme that computational feasibility must be addressed alongside accuracy [27].

2.7.3 Real-time constraints and operational viability

A recurring argument in the deep IDS literature is that real-time detection demands low-latency inference and manageable training/retraining cycles. Real-time claims, however, vary widely across studies in how runtime is measured and reported. An example of real-time IDS design using deep neural networks emphasizes the importance of achieving both detection effectiveness and timely operation, but these aims are frequently in tension when models require large training budgets or when inference depends on high-dimensional transformations. [25].

This point is central for benchmark-driven research: if a deep model delivers marginal metric gains but incurs materially higher training or inference cost, its practical advantage for IoT gateway deployment can be questionable - especially when concept drift suggests the need for periodic retraining. [20].

2.8 Knowledge distillation and compression for practical IoT IDS

IoT deployments often require smaller, faster models without catastrophic loss of detection performance. Knowledge distillation (KD) addresses this by transferring dark knowledge from a high-capacity teacher to a smaller student, typically via softened probability targets and a weighted blend of hard and soft losses. A comprehensive survey of KD provides a consolidated account of methods, distillation objectives, and practical conditions under which KD is likely to work well [28].

For IDS, the attraction of KD is straightforward:

- (i) Train a strong but expensive model (teacher) offline.

- (ii) Distil to a compact model for deployment.
- (iii) Maintain acceptable macro-class performance under resource constraints.

In practice, distillation effectiveness depends on whether the teacher provides informative class probability structure and whether the student has sufficient capacity to approximate decision boundaries. KD also interacts with class imbalance: when minority classes are under-represented, the teacher’s probabilistic output can either help by providing richer gradients or harm by reinforcing majority-class confidence. The KD survey literature highlights that distillation is not universally beneficial and must be tested in the specific data regime and objective setting. [28].

In the IoT IDS domain, this motivates an empirical approach: evaluate KD not as an assumed improvement but as a measurable trade-off among macro-averaged metrics (minority-class sensitivity), model size, and runtime. This line of reasoning is aligned with modern IoT IDS surveys [1] that emphasize lightweight models and computational efficiency as primary research directions.

2.9 Evaluation practice: metrics, imbalance, and comparability

A major obstacle in IDS research is inconsistent evaluation. Studies often report accuracy and sometimes micro-averaged F1, but these can be misleading under imbalance or multi-class conditions where minority-class errors are operationally severe. Recent work in the NLP evaluation community (with broader implications for classification) critiques common evaluation practice and provides guidance on selecting and interpreting metrics, reinforcing that metric choice must match the decision context and class structure [29].

2.9.1 Macro-averaged measures and the case for MCC

Macro-F1 treats classes equally by averaging per-class F1, making it more informative than accuracy when the dataset is imbalanced and when the research goal is robust detection across multiple attack types. This aligns with IoT IDS surveys [1] that encourage comprehensive metric reporting rather than single-number summaries.

Matthews Correlation Coefficient (MCC) is also increasingly recommended because it reflects performance across all entries of the confusion matrix and remains informative under imbalance.

Recent analysis argues that MCC is more informative than some commonly reported measures in many classification settings, supporting its use as an additional robust metric in IDS evaluation [30].

In practical IoT IDS benchmarking, reporting both Macro-F1 and MCC provides a stronger basis for comparing model families because it reduces the risk that apparent superiority is driven by majority-class dominance or metric artefacts. This is especially relevant when the objective is to detect multiple attack categories reliably rather than merely maximize overall correctness.

2.9.2 Runtime, reproducibility, and the benchmark imperative

Evaluation should account for computational cost. Surveys emphasize computational efficiency but note that many studies omit reproducible runtime reporting or conflate training time, inference latency, and preprocessing overhead [1]. Because IoT IDS may require retraining under drift, training costs can become operationally relevant. Because gateways may process high-throughput data streams, inference latency can become a safety constraint.

Reproducibility also depends on deterministic pipeline design: fixed random seeds, stratified splits, training-only preprocessing fit, and explicit description of sampling/capping strategies. Leakage-resistant evaluation frameworks demonstrate that without such controls, reported results can be systematically optimistic and unstable across reruns. [19].

Finally, the literature increasingly treats hyperparameter optimization as a necessary part of fair comparison - particularly for strong learners such as gradient boosting and modern ensemble systems. Frameworks such as Optuna formalize efficient hyperparameter search and are widely used in applied ML settings to avoid arbitrary or inconsistent tuning across models [31]. For benchmark research, the key methodological requirement is not to tune some models more than others, but to apply a transparent and consistent tuning strategy aligned with the research question (e.g., tuned best-in-family vs. off-the-shelf comparison).

2.10 Synthesis: what the literature implies for this research design

The reviewed literature yields several convergent implications that directly justify a comparative benchmark of model families and the exploration of tuned and ensemble-based proposals for IoT IDS:

1. Dataset realism and scale are pivotal. Modern datasets such as CICIoT2023 and IoT/IIoT testbed-derived datasets (e.g., TON_IoT) exist precisely because older datasets do not reflect contemporary IoT threat surfaces or traffic dynamics. [3], [11]-[14].
2. Methodological validity is fragile without leakage controls. Recent evidence shows that leakage-resistant methodology can significantly alter outcomes, making rigorous split and preprocessing practice a primary requirement for credible benchmarking [19].
3. Ensemble tree methods are consistently strong for tabular IDS features, combining high performance with operationally meaningful interpretability options (e.g., SHAP-based explanations) [23].
4. Deep learning is not uniformly superior on tabular IDS features, and its benefits depend on data modality, architecture alignment, and careful training. Deep models also raise deployment concerns in real-time IoT settings unless efficiency is explicitly addressed. [26], [27].
5. Concept drift and feature dynamics challenge static benchmarks. Even if a model ranks highest in a single benchmark run, drift-aware thinking is needed to interpret performance as a snapshot rather than a guarantee, strengthening the argument for robust validation and transparent reporting [20].
6. Metric choice determines what best means. Under multi-class imbalance, macro-averaged metrics and MCC provide a more defensible view than accuracy alone, and modern evaluation guidance supports the use of metrics that reflect the full confusion structure [29], [30].
7. Practical deployment motivates compression and KD, but benefits are empirical rather than guaranteed. KD is a plausible pathway to deployable IoT IDS models, yet the KD literature

cautions that effectiveness depends on teacher/student capacity alignment and data regime, justifying careful experimental evaluation rather than assumption. [28].

Collectively, these points highlight a research gap at the intersection of comparability (fair benchmark across model families under consistent preprocessing and tuning), robustness (metrics and validation resistant to imbalance and leakage), and practicality (runtime/cost awareness and compression feasibility). This gap motivates the present research’s focus on benchmarking traditional ML, ensemble ML, deep learning, and a proposed ensemble/tuning strategy - evaluated using macro-sensitive metrics and explicit computational considerations.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Research philosophy

This dissertation adopts a positivist research philosophy because the central goal is to measure and compare the performance of alternative intrusion detection models under a controlled and reproducible experimental protocol. Positivism is appropriate when the research objective is to test propositions using observable evidence, emphasizing objectivity, replicability, and quantification of outcomes (e.g., macro-averaged F1 and Matthews correlation coefficient). The work therefore treats model performance as an empirically measurable phenomenon and evaluates competing approaches using consistent metrics, held-out test data, and explicit reporting of computational cost [32].

In cybersecurity and digital forensics research, positivism is commonly justified when the artefacts under study (here, trained classifiers and their predictions) can be assessed using standardized measurement procedures and when the primary contribution is evidence of performance and trade-offs rather than interpretive meaning-making. This stance aligns with the dissertation's purpose: to produce a defensible benchmark over traditional machine learning, ensemble machine learning, and deep learning approaches for IoT intrusion detection, and to evaluate a proposed approach under the same measurement regime.

3.2 Research approach

A deductive approach is employed. Deduction is suitable when the study begins from theory and prior evidence (e.g., that ensemble methods often offer strong tabular classification performance and that distillation may reduce model complexity), and then tests these expectations against experimental results. In practice, deduction is operationalized here through:

- (i) Constructing hypotheses derived from recent literature on evaluation under class imbalance and on model selection for tabular data.
- (ii) Implementing a pre-specified pipeline.
- (iii) Testing the hypotheses using held-out evaluation measures.

While the experimentation includes exploratory elements (e.g., hyperparameter optimization), the logical structure remains deductive because optimization is treated as a systematic manipulation of design choices to test whether the resulting tuned/proposed configuration yields measurable improvement over baselines.

3.3 Research strategy

The chosen research strategy is a controlled computational experiment. The experiment compares multiple model families trained on the same processed dataset and evaluated on the same test split. The strategy is appropriate because it supports:

- (i) Isolation of the effect of key design choices (model family, tuning, and ensembling)
- (ii) Consistent measurement.
- (iii) Replication by other researchers, subject to compute and data availability constraints.

The benchmark uses CICIoT2023, a modern IoT intrusion detection dataset designed to support security analytics and machine learning evaluation in realistic IoT topologies. The dataset’s scale and diversity of attacks makes it suitable for studying performance across model families, but also introduces practical constraints (e.g., training time and memory footprint), which motivates explicit runtime reporting [33].

3.4 Variables and hypotheses

3.4.1 Independent variables

The principal independent variables are:

1. **Model family / algorithm class**
 - Traditional ML (e.g., linear models, Naive Bayes variants, nearest neighbors, shallow decision trees).
 - Ensemble ML (e.g., bagging, random forests, gradient boosting variants, boosted trees such as XGBoost).

- Deep learning (e.g., MLP variants, CNN-1D, LSTM).
- Proposed/hybrid approaches (e.g., tuned models and/or voting ensembles, and optional KD-based student models).

2. **Tuning strategy**

- Default or minimally tuned baselines versus explicit hyperparameter optimization (HPO). The experiment uses a bounded number of trials for feasibility, treating HPO as a controlled manipulation rather than open-ended search. Optuna is a widely used HPO framework for efficient trial-based optimization [34].

3. **Sampling / capping strategy**

- Selection of a limited number of CICIOT2023 CSV partitions and capping the total number of rows and/or per-class counts (implemented to keep the benchmark feasible in a Colab environment). This is treated as a methodological constraint and a design variable because it affects class coverage and the effective training distribution.

4. **Ensembling strategy**

- Use (or absence) of bagging/boosting/voting, including whether the final proposed result is a single tuned model or an ensemble aggregation.

5. **Knowledge distillation (KD) settings**

- Temperature and blending coefficient (e.g., α) that control the balance between hard labels and soft teacher targets in student training. KD is a recognized approach for transferring knowledge from a stronger teacher to a smaller student, but its effectiveness depends on data modality, capacity alignment, and tuning [28].

3.4.2 **Dependent variables**

The dependent variables are the measured outcomes:

- Macro-F1 (primary): macro-averaged F1 places equal weight on each class, making it informative under class imbalance.

- MCC (primary/secondary): Matthews correlation coefficient is widely argued to be more informative than accuracy and (in some cases) F1 under imbalance because it captures all confusion matrix terms [35].
- Accuracy (secondary): reported for comparability but interpreted cautiously under imbalance.
- Additional metrics (where computed in the pipeline): Cohen’s kappa (agreement adjusted for chance), balanced accuracy, and ROC-AUC (subject to feasibility in multiclass settings). Cohen’s kappa originates from classic agreement measurement work [36].
- Runtime (pragmatic DV): training time (seconds) as a proxy for computational cost and operational feasibility.

The selection of macro-F1 and MCC is justified by the evaluation literature emphasizing that metric choice must match problem characteristics, especially in multiclass and imbalanced classification [37].

3.4.3 Hypotheses

Given the experimental focus and the dataset characteristics, the dissertation tests the following hypotheses.

H1 (performance): There is no statistically and practically meaningful improvement in macro-F1 and MCC when using the proposed method (e.g., tuned/ensemble configuration) compared with the strongest non-proposed baseline on the same test split.

H2 (performance): The proposed method yields higher macro-F1 and MCC than the strongest non-proposed baseline on the same test split.

H3 (cost-benefit): Any observed improvement in macro-F1/MCC from the proposed method is not accompanied by a meaningful increase in training time.

H4 (cost-benefit): The proposed method improves macro-F1/MCC but incurs increased training time, implying a measurable performance-cost trade-off.

3.5 Data and tools

3.5.1 Dataset selection and justification

The primary dataset is CICIoT2023, introduced as a real-time dataset and benchmark for large-scale attacks in IoT environments. CICIoT2023 is a modern IoT intrusion-detection benchmark created to support realistic, large-scale evaluation of IDS models in IoT environments. It was generated from an IoT topology of 105 devices and includes 33 attacks organized into seven high-level categories (DDoS, DoS, Recon, Web-based, brute force, spoofing, and Mirai), enabling both category-level and fine-grained multi-class classification. The dataset is commonly distributed in PCAP and derived CSV formats. A peer-reviewed study reports 169 files in these formats and describes a CSV representation containing 46 flow-derived attributes used for machine learning. A defining technical property is strong class imbalance, where some attack types/categories are orders of magnitude smaller than majority classes, making imbalance-aware evaluation (e.g., Macro-F1 and MCC) and transparent preprocessing essential for credible comparisons [3].

Unlike UNSW-NB15, which is widely used but reflects a more general network setting rather than IoT-specific device behavior [7], CICIoT2023 is explicitly designed as a benchmark for large-scale IoT attack traffic and therefore better reflects contemporary IoT threat diversity and background patterns [3]. Compared with BoT-IoT - which is highly valuable for botnet-centric behaviours but can be narrower in threat emphasis and traffic composition [9] - CICIoT2023 offers a broader attack taxonomy that supports more demanding multi-class evaluation and more meaningful stress-testing of imbalance-aware metrics [3]. In contrast to TON_IoT, which is highly useful for multi-source IoT/IIoT telemetry studies but introduces subset- and modality-dependent comparability challenges (e.g., network vs telemetry vs OS components) that can confound controlled benchmarking [3,12], CICIoT2023 enables a cleaner, single-representation evaluation protocol while still maintaining strong operational relevance [3].

Key justification points (CICIoT2023 vs alternatives):

IoT realism and operational relevance: CICIoT2023 is purpose-built for IoT attack benchmarking, whereas UNSW-NB15 is not IoT-native and may not reflect IoT traffic dynamics adequately [3,7].

Breadth for multi-class evaluation: CICIoT2023’s wider attack coverage supports robust multi-class benchmarking. BoT-IoT is strong for botnet patterns but comparatively narrower in threat emphasis [3, 9].

Comparability and methodological control: CICIoT2023 supports a consistent benchmarking workflow. TON_IoT’s multiple subsets/modalities can reduce comparability unless tightly constrained [3, 11].

Imbalance-aware research suitability: The dataset’s multi-class structure naturally stresses minority-class detection and motivates reporting macro-sensitive metrics, strengthening methodological validity [3].

3.5.2 Tooling and compute environment

This research can be implemented using several compute and tooling strategies, each with implications for feasibility, reproducibility, and experimental control. A local workstation offers stable long-running compute and full control over storage and dependencies, but it can undermine replicability because hardware (CPU/RAM/GPU) and software configurations vary and may affect training time and convergence, especially for deep learning baselines. Managed cloud notebooks provide a more standardized, shareable execution interface that improves transparency by running the full pipeline (preprocessing-training-evaluation) in a documented, consistent sequence, although they impose operational constraints such as session limits, resets, and fluctuating accelerator availability that require persistent storage and robust data loading. Containerized workflows (e.g., Docker with pinned dependencies) typically maximize reproducibility but add engineering overhead that is disproportionate for an MSc dissertation focused on empirical benchmarking. Accordingly, this dissertation implements the benchmark in Google Colab (using an NVIDIA T4 GPU where applicable) with all datasets, models, and outputs persisted to Google Drive to mitigate runtime volatility and prevent undocumented reruns, balancing practical feasibility with methodological control and replication readiness. More detailed explanation added on appendix 3.1 section.

3.5.3 Mathematical formalization of the experimental protocol

Core experimental notation

Define the dataset and splits:

- Dataset:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N, y_i \in \{1, \dots, K\}$$

- Stratified split into train/validation/test:

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}, \text{with class proportions preserved.}$$

Sampling/capping + rare-class filtering

- Per-class set: $\mathcal{D}_c = \{(x_i, y_i) \in \mathcal{D} : y_i = c\}$
- Cap k per class:

$$\mathcal{D}^{cap} = \bigcup_{c=1}^K \text{Sample}(\mathcal{D}_c, \min(|\mathcal{D}_c|, k))$$

- Rare-class threshold m : keep class only if $|\mathcal{D}_c| \geq m$.

Preprocessing (StandardScaler) supports leakage-safe methodology

- Standardization using train-only statistics:

$$x' = \frac{x - \mu_{train}}{\sigma_{train}}$$

This is a clean mathematical statement that will support leakage-control argument.

Metrics (the most important validation math in an IDS benchmark)

Accuracy:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[\hat{y}_i = y_i]$$

Per-class Precision/Recall:

$$P_c = \frac{TP_c}{TP_c + FP_c}, R_c = \frac{TP_c}{TP_c + FN_c}$$

Per-class F1 and Macro-F1:

$$F1_c = \frac{2P_c R_c}{P_c + R_c}, \quad \text{Macro-F1} = \frac{1}{K} \sum_{c=1}^K F1_c$$

Balanced Accuracy:

$$\text{BAcc} = \frac{1}{K} \sum_{c=1}^K \frac{TP_c}{TP_c + FN_c}$$

Multiclass MCC

Let C be the confusion matrix, $s = \sum_i \sum_j C_{ij}$, $c = \sum_k C_{kk}$,

$p_k = \sum_j C_{kj}$ (row sums), $t_k = \sum_i C_{ik}$ (column sums). Then:

$$\text{MCC} = \frac{cs - \sum_k p_k t_k}{\sqrt{(s^2 - \sum_k p_k^2)(s^2 - \sum_k t_k^2)}}$$

This is a rigorous definition and will map directly to MCC values.

3.6 Validity, reliability, and bias control

3.6.1 Internal validity

Internal validity concerns whether performance differences can reasonably be attributed to the independent variables under study (model family, tuning, Ensembling, and related settings) rather than to confounders. Several methodological options exist for controlling confounding in comparative IDS benchmarking. For preprocessing, one approach is to allow each model to use its own best-practice preprocessing (e.g., model-specific scaling, feature filtering, or encoding). This can maximize per-model performance but weakens causal interpretability because improvements

may arise from privileged preprocessing rather than the model itself. An alternative is to enforce a single standardized preprocessing protocol across models, accepting that some models may not receive optimally tailored preprocessing. The present study adopts the standardized pipeline option because it makes the comparison more internally valid: observed differences are more plausibly driven by modelling choices rather than inconsistent data preparation.

For data partitioning, a simple random split is often used due to convenience. However, under multi-class imbalance it can distort class proportions across splits, causing unstable macro-level metrics and misleading comparisons. A more robust alternative is stratified splitting, which preserves class proportions in train/validation/test partitions and reduces sampling-induced variance, particularly for minority attacks that strongly influence Macro-F1 and MCC [37]. The dissertation therefore adopts stratified splitting as the most defensible option for imbalance-sensitive evaluation in multi-class IDS contexts.

For run-to-run variability, a common option is to run models once without controlling randomness, which reduces experimental overhead but undermines repeatability (especially for stochastic optimizers and deep learning initialization). At the other extreme, repeated runs with confidence intervals offer stronger reliability but can be computationally expensive for an MSc-scale project with many model families. This study adopts an intermediate approach: fixed random seeds where supported to stabilize splits and deterministic learners, while acknowledging residual stochasticity in deep learning and interpreting single-run differences conservatively. Repeated stratified runs are therefore treated as a recommended extension for future work rather than an unreported assumption.

For information leakage, several leakage pathways are well documented in ML evaluation, including fitting preprocessing on the full dataset, allowing duplicates across splits, or using temporally correlated records that contaminate test performance. A weak option is to assume the dataset is clean and proceed without leakage checks, these risks inflated results and weakens credibility. A stronger option is to enforce a strict protocol where transformations are learned only on training data and applied to validation/test data, alongside explicit leakage audits [38]. In this dissertation, the evaluation protocol follows the stricter option by constraining fitting to training data and by treating deeper leakage audits (duplicate and temporal checks) as recommended future enhancements where operational deployment claims are required.

3.6.2 Construct validity

Construct validity addresses whether the evaluation measures capture what they claim to measure. In IoT IDS research, effectiveness can be operationalized in multiple ways. A common choice is accuracy, which is intuitive but can be misleading when class imbalance is severe because strong performance on majority classes can dominate the metric. Another option is micro-averaged F1, which behaves similarly under imbalance. In contrast, Macro-F1 weights classes equally, making it more aligned with the construct of balanced detection capability across minority and majority attack types, this is particularly relevant for multi-class attack taxonomies. A further complementary option is MCC, which uses the confusion matrix structure and is frequently recommended where imbalance or skew makes accuracy and F1 less stable as indicators of genuine classification quality. The dissertation therefore uses Macro-F1 and MCC alongside accuracy to improve construct coverage of IDS effectiveness under multi-class imbalance.

Operational feasibility can also be measured in different ways. Strict deployment-oriented work often reports inference latency, throughput, memory footprint, and energy usage, which provide the strongest link to real-time requirements. However, comprehensive inference benchmarking can require dedicated deployment profiling and controlled hardware conditions. As a feasible proxy within an MSc benchmarking scope, the dissertation reports runtime cost using training time as an interpretable computational burden indicator while explicitly distinguishing it from inference latency. This choice enables principled discussion of efficiency trade-offs without overstating deployment readiness.

3.6.3 External validity

External validity concerns the extent to which findings generalize beyond the experimental setting. One possible approach is to claim broad generality from a single dataset benchmark. However, IDS performance is known to be dataset and pipeline dependent due to differences in traffic generation, attack execution, feature extraction, and class imbalance. A more defensible option is to bound conclusions to the chosen dataset and protocol and treat generalization as a hypothesis requiring further evaluation. This dissertation adopts the bounded approach, findings are directly applicable to CICIOT2023 under the stated sampling, preprocessing, and split protocol, while

cross-dataset generalization (e.g., to BoT-IoT, TON_IoT, UNSW-NB15) is not asserted without explicit transfer or cross-dataset testing. Consequently, cross-dataset evaluation and temporal drift testing are framed as future work to strengthen generalizability claims.

CHAPTER 4: DESIGN PROCESS (EXPERIMENT DESIGN)

This chapter specifies the end-to-end experimental design used to benchmark intrusion detection models on CICIOT2023 traffic and to evaluate the proposed enhancements. The design is grounded in two priorities:

- (i) Methodological comparability across heterogeneous model families (traditional ML, ensemble ML, deep learning, and proposed/hybrid methods).
- (ii) Reproducibility through a deterministic, modular pipeline that reduces common sources of bias (e.g., leakage, unstable splits, and imbalance-driven metric inflation).

Concerns about leakage and non-reproducible pipelines are well documented in modern ML practice and can materially distort reported IDS performance if controls are weak.

4.1 Overall Pipeline Architecture

The experimental workflow is structured as a staged pipeline that converts raw CICIOT2023 partitions into comparable model artefacts and a unified benchmark report:

1. **Data ingestion and local caching:** a fixed number of CICIOT2023 CSV partitions are copied from cloud storage to local runtime storage to reduce I/O volatility and enable repeatable runs.
2. **Data consolidation and sampling:** partitions are loaded sequentially until a target row budget is reached. Classes below a minimum frequency threshold are filtered to reduce extreme sparsity effects.
3. **Pre-processing:** numeric coercion, missing-value handling, label encoding, and feature scaling (where appropriate) are applied using training-only fitted transforms to reduce leakage risk.
4. **Split design:** stratified train/test split is used for the final benchmark set, a validation subset is derived from the training data for deep learning early stopping.
5. **Model training:** multiple model families are trained under a common evaluation interface, tuned and proposed variants are built on top of the same pre-processing and splits.

6. **Evaluation and reporting:** predictions are scored using a metric suite that prioritizes imbalance-robust measures (Macro-F1, MCC, balanced accuracy) and records runtime.

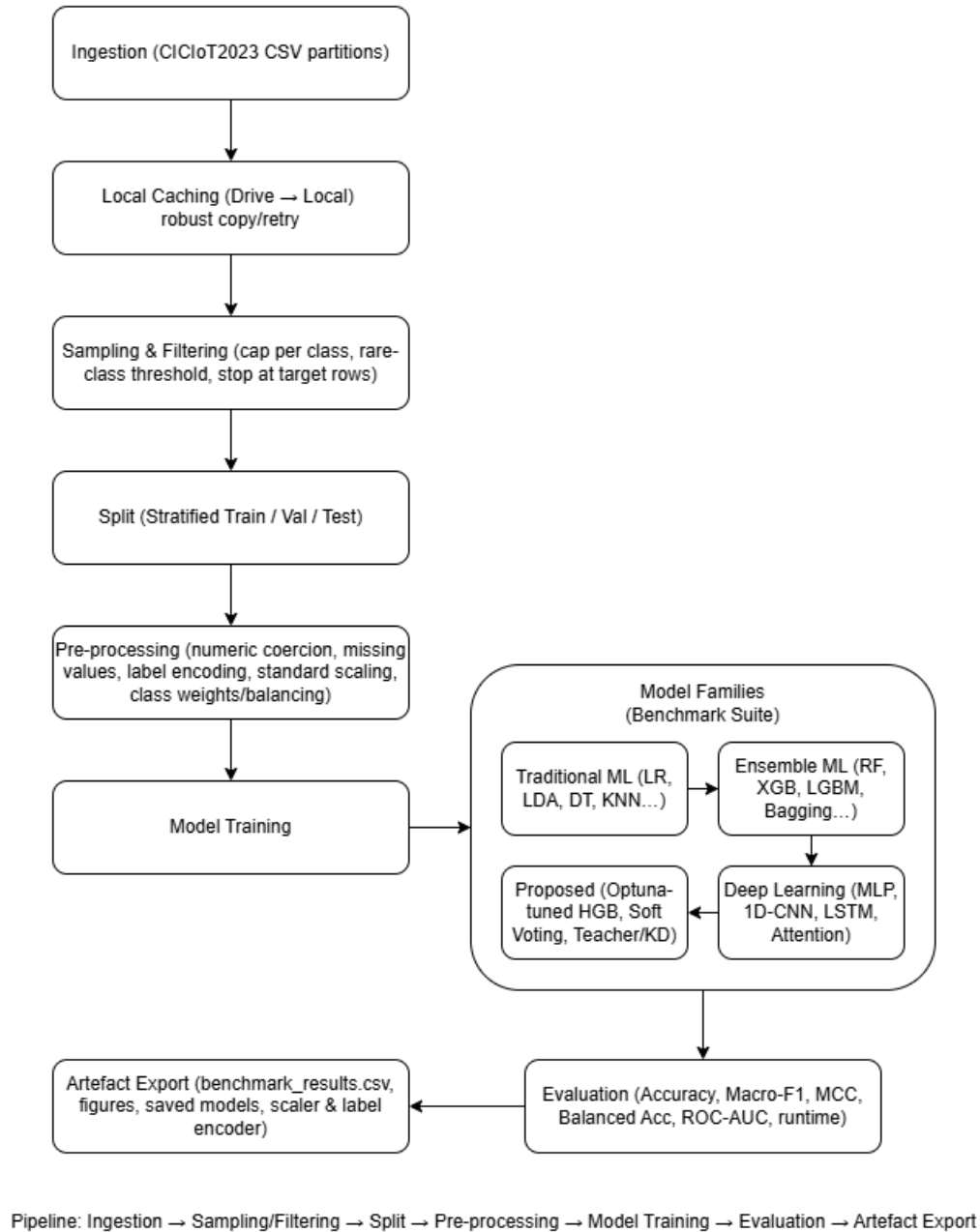


Figure 4.1: Pipeline Architecture of the Experimental Benchmarking Framework.

This modular separation is intentional. It allows the research to attribute performance differences primarily to modelling choices rather than inconsistent pre-processing or data handling.

4.2 Dataset Ingestion Design

4.2.1 Dataset choice and source

CICIoT2023 is a large-scale, labelled IoT attack dataset designed to support benchmarking under diverse attack conditions. Its scale and contemporary attack coverage make it suitable for evaluating modern IDS pipelines, particularly when the research aims to assess practical trade-offs between accuracy and computational cost. The dataset collected and downloaded from official site of Canadian Institute for Cybersecurity, University of New Brunswick. [42]

4.2.2 Partition selection, storage strategy, and determinism

The implementation copies a fixed number of dataset partitions (`n_files_to_copy`) from cloud storage into a local directory before processing. This design reduces variability due to network latency and Google Drive throughput constraints, improving run-to-run stability and lowering the risk of partial reads or timeouts during training. The copied subset is then treated as the fixed experimental corpus for that run.

4.2.3 Sampling, row budget, and class coverage controls

Because CICIoT2023 partitions can be extremely large, the pipeline applies a row budget (`target_rows`) to cap the total number of ingested instances. This allows the benchmarking workload to fit within constrained compute sessions (e.g., Colab runtime limits) while still maintaining multi-class coverage. The pipeline also filters out classes below a minimum count threshold (`min_class_count`) to avoid unstable metrics arising from ultra-rare labels (where Macro-F1 and MCC can become dominated by a small number of samples). The methodological risk being mitigated here is the distortion of headline accuracy by majority classes, which is a known failure mode in imbalanced classification [43].

4.2.4 Training-set balancing for ensemble construction

For the proposed voting ensemble stage, the implementation constructs a balanced training subset by applying a cap per class (e.g., `cap=5000`) to reduce majority-class dominance in the fitting process. This is consistent with IDS literature noting that resampling and balancing strategies can materially affect detection performance, particularly under multi-class imbalance.

4.3 Feature Handling Design

4.3.1 Baseline feature set and data types

CICIoT2023 provides tabular, flow/statistical features suitable for both classical ML and deep learning architectures designed for tabular inputs. The pipeline enforces numeric coercion to ensure model compatibility and drops or imputes invalid values according to the implementation rules. This step is critical because mixed-type or malformed feature columns can cause silent failures (e.g., degenerate training where models converge to majority-class outputs).

4.3.2 Missing values and leakage controls

The pipeline is designed to apply any fitted transforms (e.g., scalers) using training data only, then reuse those transforms for validation and test sets. This directly addresses leakage pathways in ML pipelines, where fitting pre-processing on the full dataset can bleed test-set distributional information into training and inflate performance.

4.3.3 Scaling rationale by model family

A single design objective is comparability without forcing inappropriate assumptions. Accordingly:

- Tree-based ensembles (e.g., Random Forest, Extra Trees, gradient boosting variants) do not strictly require feature scaling. They are generally invariant to monotonic transformations of features.
- Distance or gradient-sensitive models (e.g., KNN, linear models, MLP-based deep learning) benefit from standardization for stable optimization and meaningful distance geometry.

This selective scaling approach reduces the risk of penalizing one family with pre-processing tuned for another, while still preserving fairness through consistent application within each family.

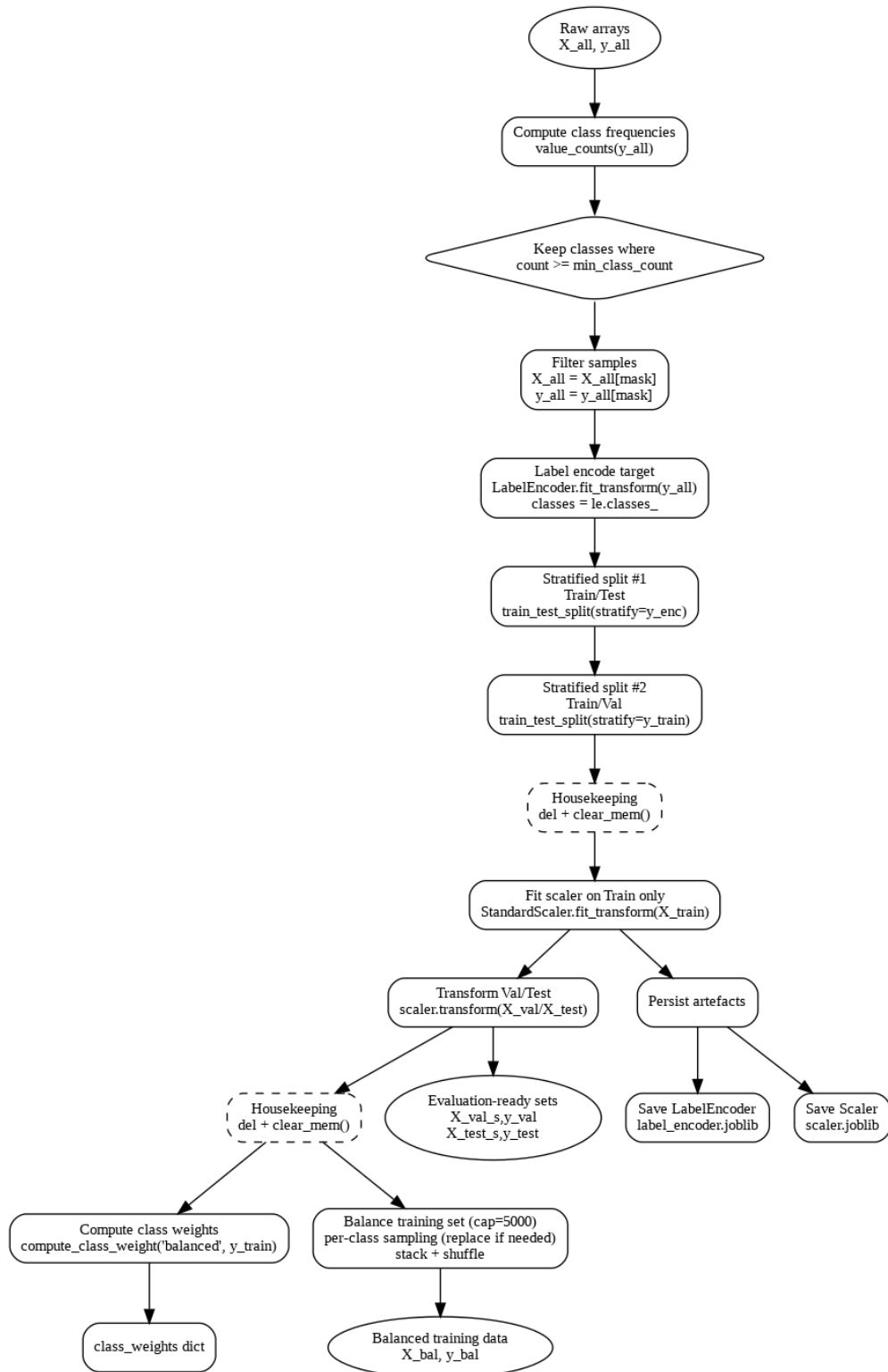


Figure 4.2: Data Cleaning and Preprocessing Technique Pipeline

4.4 Train/Validation/Test Design

4.4.1 Stratified splitting strategy

The benchmark uses a stratified train/test split to preserve label distribution across partitions. This matters because multi-class IDS datasets frequently exhibit heavy skews. Without stratification, minority classes can vanish from the test set and produce misleadingly optimistic metrics. Modern work on data splitting emphasizes that splitting strategy can materially change performance estimates and should be explicitly justified rather than treated as incidental [44].

4.4.2 Split proportions and validation usage

A held-out test set (e.g., `test_size=0.2`) will be used for the final benchmark. A validation subset is carved from the training data to support early stopping and hyperparameter selection for deep learning models, ensuring that the test set remains a final, untouched evaluation resource. The rationale aligns with established guidance that evaluation metrics should be derived from genuinely unseen data to avoid optimistic bias [45]. So, The Split will be 80% Training, 20% Test and 12% Validation by Stratified splitting strategy.

4.4.3 Early stopping as a design control (deep learning)

Deep learning models use early stopping to reduce overfitting risk and to manage runtime. In the IDS setting - where the objective includes practical deploy ability - early stopping is also a resource-control mechanism, preventing prolonged training that yields marginal gains.

4.5 Model Family Design

The model set is intentionally broad to answer the research questions about comparative performance and runtime trade-offs across major IDS modelling paradigms.

4.5.1 Traditional ML baseline set

The traditional ML category provides interpretable and computationally light baselines and includes:

- Decision Tree, Logistic Regression, kNN (k=5), SGD Classifier, LDA, Perceptron, Ridge Classifier, Gaussian Naïve Bayes.

These baselines help quantify the uplift attributable to ensemble and deep architectures rather than to pre-processing alone.

4.5.2 Ensemble ML set

Ensemble learning is widely used in tabular IDS contexts due to its ability to model non-linear interactions while remaining relatively robust on engineered features. The ensemble set includes:

- Bagging (Decision Tree), Random Forest, Extra Trees, Gradient Boosting, HistGradientBoosting, AdaBoost, and (where available in the environment) XGBoost and LightGBM.

XGBoost is included as a high-performing gradient boosting reference widely used in structured data competitions and security analytics, with strong empirical performance in many tabular domains [46].

4.5.3 Deep learning set

The deep learning set is designed to test whether higher-capacity representation learning improves detection under the constrained tabular feature regime:

- Shallow MLP, Deep MLP, Wide MLP, 1D-CNN, LSTM, and an Attention-MLP variant.

Recent IoT IDS literature supports the relevance of deep models for DDoS/attack detection. However, the design explicitly treats deep learning performance as an empirical question rather than an assumption, given that tabular flow features do not always benefit from convolutional or recurrent inductive biases [47].

4.5.4 Proposed components

The proposed track contains three extensions motivated by performance and deployment considerations:

1. Optuna-tuned HistGradientBoosting (Proposed Tuned HistGB)

Hyperparameter optimization is used to improve a strong baseline without altering the dataset or evaluation protocol. Optuna is a modern framework for efficient hyperparameter search in constrained runtime settings.

2. Soft Voting Ensemble (Proposed Voting Ensemble)

The voting design aggregates probabilistic outputs from multiple tree-based learners. In the implementation, the ensemble candidates are drawn from tuned/best-available learners (including tuned HistGB, Random Forest, Extra Trees, and, where available, XGBoost and LightGBM). Soft voting is chosen to exploit complementary error profiles while maintaining a single deployable predictor.

3. Knowledge Distillation path (Teacher $\rightarrow$ Student)

Knowledge distillation is included as a deployment-oriented mechanism: a larger teacher model can transfer softened class probability information to a smaller student, potentially reducing inference cost with limited loss in predictive performance. A recent survey consolidates the contemporary KD rationale and design patterns, supporting its use as a compression mechanism when deployment constraints matter.

4.6 Evaluation Design

4.6.1 Metric selection and definitions

The evaluation suite includes: Accuracy, Macro-F1, MCC, balanced accuracy, macro precision/recall, weighted F1, Cohen’s kappa, and ROC-AUC, alongside training time in seconds.

Macro-F1 and MCC are prioritized because IoT IDS datasets are typically imbalanced, metrics that average per class (macro) and correlation-based measures (MCC) provide more reliable signals of minority-class performance than raw accuracy [35].

Balanced accuracy is additionally reported because it corrects for majority-class dominance by averaging recall across classes and has been recommended in applied settings where class imbalance is material [48].

4.6.2 Runtime measurement approach

The design records training time per model as a first-order proxy for computational cost. Although training time is not identical to inference latency, it provides a comparable measure of computational burden across methods under identical runtime conditions. The rationale is

practical, in many organizational IDS settings, training and retraining cost (especially under concept drift) is itself operationally significant.

4.6.3 Planned figures and reporting artefacts

To improve interpretability and reduce ambiguity in comparisons, the evaluation plan includes:

- Cross-model bar charts for Accuracy, Macro-F1, and MCC.
- Category-wise summaries (Traditional ML vs Ensemble ML vs Deep Learning vs Proposed).
- Ranked leaderboard plots and a best-in-class radar chart for the top representatives.
- Confusion matrix analysis for the best-performing model to identify systematic confusion among attack classes.

These visual artefacts will be exported alongside a consolidated CSV benchmark table to support traceability and reproducibility.

CHAPTER 5: IMPLEMENTATION (EXPERIMENT)

This chapter documents the practical implementation of the CICIOT2023 benchmarking pipeline, including data ingestion, preprocessing, model training across multiple families, and artefact generation. The implementation narrative follows the executed Colab notebook and its logged outputs, ensuring the description reflects what was actually run rather than an idealized design.

5.1 Environment setup and dependency management

All experiments were executed in Google Colab to ensure accessible compute and a repeatable runtime baseline. The notebook begins by installing the required packages for classical machine learning, gradient-boosting frameworks, and hyperparameter optimization:

- imbalanced-learn, optuna, xgboost, lightgbm, and catboost are installed via pip.
- The environment then imports core scientific Python libraries (numpy, pandas, matplotlib, seaborn), persistence tooling (joblib), and the complete scikit-learn stack required for preprocessing and modelling (classical models, ensembles, metrics).

A key engineering step is GPU detection and safe allocation behavior for TensorFlow. The notebook checks for available GPU devices and enables memory growth to reduce the likelihood of runtime out-of-memory failures during deep learning training. The pipeline also standardizes reproducibility controls via a global seed (SEED = 42) applied consistently to Python's random module, NumPy, and TensorFlow.

Finally, the pipeline performs feature availability checks for optional model families. If XGBoost, LightGBM, CatBoost, or Optuna fail to import, the benchmark automatically downgrades by skipping those models rather than crashing. This behavior is essential in Colab contexts where dependency resolution can occasionally vary across sessions.

Implementation configuration: The benchmark is driven by a single configuration dictionary logged at runtime, which fixes the practical experiment scope (files, sample cap, split ratio, training epochs, and KD hyperparameters). The executed configuration was: `n_files_to_copy=5`, `target_rows=100000`, `min_class_count=200`, `test_size=0.2`, `batch_size=256`, `epochs_dl=25`, `epochs_kd=15`, `patience=5`, `kd_temp=4.0`, `kd_alpha=0.3`, `optuna_trials=10`.

5.2 Data pipeline implementation

5.2.1 Drive mount, robust copy logic, and local caching

The dataset is stored in Google Drive and then copied into a local Colab directory for faster sequential reading. The notebook defines:

- `DRIVE_DATA_DIR ("/content/drive/MyDrive/valo/wataiData/csv/CICIoT2023")`,
- `LOCAL_DATA_DIR ("/content/ciciot_local")`,
- `OUT_DIR ("/content/outputs_final")`.

Because Drive I/O can fail intermittently, the notebook implements a robust copy system:

1. **Drive mounting with forced remount:** a helper function mounts Drive with `force_remount=True`.
2. **Single-file copy with retries:** `copy_single_file()` attempts to copy a file up to three times, on failure it remounts Drive and retries.
3. **File discovery and spread selection:** `ensure_local_files()` enumerates already-cached CSVs, computes how many additional files are needed, then selects files across the available set using a step interval (rather than copying only adjacent partitions).

This approach prevents the entire experiment from being blocked by transient Drive errors and improves representativeness versus copying only the first N partitions.

The execution log confirms that two files already existed locally, and the pipeline copied three more to reach five total partitions.

5.2.2 Loading logic, label normalization, and capped sampling

The benchmark intentionally avoids loading full CICIoT2023 partitions by applying per-class capping while reading. The loader applies two key steps:

- **Label normalization:** labels are lowercased and whitespace-normalized to reduce accidental duplicates caused by formatting differences.

- **Feature augmentation (lightweight engineering):** four derived features are computed when the source columns are available:
 - $\text{bytes_pkt} = \text{Tot sum} / (\text{Number} + 1)$
 - $\text{rate_ratio} = \text{Srate} / (\text{Drate} + 0.001)$
 - log-transformed variants: `flow_duration_log`, `Tot sum_log`
 - Each engineered feature is sanitised against $\pm\text{inf}$ and missing values.

The loader computes a cap per class based on the global target row budget and the observed label set, then samples each class up to that cap while reading across files. In the executed run, the loader used a cap of 3225 per class.

After reading five local files, the loader reported 73,893 samples, 50 features, and 35 classes, with the most frequent classes appearing at the cap (e.g., `dos-udp_flood`, `ddos-rstfinflood`, `benigntraffic`, etc.).

5.2.3 Rare-class filtering, encoding, scaling, and splitting

To reduce instability from extremely low-support classes, the pipeline enforces a minimum class count threshold: classes with fewer than `min_class_count=200` samples are removed. After filtering, the dataset became 73,211 samples across 27 classes.

The dataset is then partitioned using stratified splits:

- Train/test split uses `test_size=0.2`.
- A validation set is created from the training data (logged final sizes below).

The logged split sizes were:

- Train: 51,539
- Validation: 7,029
- Test: 14,643

For model compatibility and stability, the notebook applies:

- Label encoding (`LabelEncoder`) to map the 27 classes to integer IDs.

- Standardization (StandardScaler) across numeric features.
- Class weighting via scikit-learn's `compute_class_weight`, passed into deep learning training to reduce bias toward higher-frequency classes.

Finally, a balancing helper exists to cap very large classes in the training set (`cap=5000`). In this run, the log indicates the balanced training size remained 51,539, implying the cap did not further reduce the already-capped training distribution.

Key implementation excerpt (data ingestion and capping logic, shortened):

```
cap = max(800, config['target_rows'] // max(5, len(all_labels)))
```

```
# ...
```

```
for file in files:
```

```
    df = pd.read_csv(file)
```

```
    df[label_col] = normalize_labels(df[label_col])
```

```
    df, feat_cols = add_features(df, feat_cols)
```

```
    # sample per class up to cap while accumulating
```

This excerpt reflects the executed logic: sampling is applied during loading rather than after full concatenation, which is more memory-efficient in Colab contexts.

5.3 Model implementations

The implementation benchmarks four categories:

1. Traditional ML
2. Ensemble ML
3. Deep Learning
4. Proposed methods

All models are trained on the processed training set and evaluated on the same held-out test set. Metrics are computed through a shared helper that returns (at minimum) Accuracy, Macro-F1, MCC, and macro ROC-AUC using one-vs-rest.

5.3.1 Traditional ML models (8)

Eight baseline models are trained using scikit-learn:

- Logistic Regression (max_iter=300)
- Ridge Classifier
- SGD Classifier (loss='log_loss')
- Perceptron
- Gaussian Naive Bayes
- Linear Discriminant Analysis (LDA)
- KNN (k=5)
- Decision Tree (max_depth=20)

Because KNN can be slow on larger training sets, the notebook subsamples up to 25,000 instances specifically for that model while keeping other models on the full training set. This design preserves feasibility without removing KNN from the comparative baseline.

5.3.2 Ensemble ML models (9)

The ensemble family includes both bagging- and boosting-based approaches:

- Random Forest (n_estimators=200, max_depth=20)
- Extra Trees (n_estimators=200, max_depth=20)
- Bagging (Decision Trees) (n_estimators=100)
- AdaBoost (n_estimators=100)
- Gradient Boosting (n_estimators=100, max_depth=8)
- HistGradientBoosting (max_iter=200, max_depth=12)

The benchmark also includes:

- XGBoost (n_estimators=200, max_depth=10, learning_rate=0.1)
- LightGBM (n_estimators=200, max_depth=12, learning_rate=0.1)
- CatBoost (iterations=200, depth=8, learning_rate=0.1)

All ensemble models are trained on the full (scaled) training set, and each trained estimator is stored in a dictionary (trained) for later reuse in the proposed voting ensemble.

5.3.3 Deep learning models (6)

Deep learning training is executed in TensorFlow/Keras. Early stopping is applied consistently across deep learning runs (patience=5, restore best weights) with validation loss monitoring.

The six implemented neural architectures are:

1. Shallow MLP (Dense 64 + dropout + logits layer)
2. Deep MLP (stacked Dense layers, implemented in the notebook's DL section)
3. Wide MLP (largest-performing DL baseline in the reported results table)
4. 1D-CNN
5. LSTM
6. Attention MLP (implemented but observed to collapse in performance in final results)

All neural networks are trained with SparseCategoricalCrossentropy(from_logits=True), meaning the final layer outputs abnormalized logits. Probabilities are derived via softmax at evaluation time for consistent metric computation.

5.3.4 Proposed solution components (teacher, tuned HistGB, voting ensemble, KD student)

The proposed category contains four implemented components.

(a) Proposed DL Teacher (GELU + residual + batch normalization).

The teacher is a fully connected network designed for tabular inputs, featuring:

- Dense(512) → BN → GELU → Dropout(0.3)

- Residual block: Dense(512) $\rightarrow$ BN $\rightarrow$ GELU $\rightarrow$ Dropout(0.3) plus skip connection
- Dense(256) $\rightarrow$ BN $\rightarrow$ Dropout(0.25)
- Dense(128) $\rightarrow$ logits output layer

Optimization uses AdamW with a cosine decay learning-rate schedule and class weights.

(b) Optuna-tuned HistGradientBoosting.

The notebook runs 10 trials optimizing macro-F1 on the validation set. The search space is explicitly defined:

- learning rate $lr \in [0.02, 0.15]$
- max depth $d \in [6, 18]$
- max iterations $i \in [100, 300]$

The tuned model is then retrained on the training set and evaluated on the test set, and also stored for inclusion in the voting ensemble.

(c) Proposed Voting Ensemble (soft voting).

The final proposed classifier is a soft-voting Voting Classifier that composes:

- Tuned HistGradientBoosting,
- Random Forest,
- Extra Trees,
- Plus XGBoost and LightGBM when present.

This is implemented as an explicit estimator list (`ens_list`) and trained on the full training set.

Here is the proposed model code:

```

# 4.3 Voting Ensemble
print("\n Proposed Voting Ensemble")
ens_list = [
    ('hgb', trained.get('Tuned_HistGB', trained['HistGradientBoosting'])),
    ('rf', trained['Random Forest']),
    ('et', trained['Extra Trees']),
]
if 'XGBoost' in trained:
    ens_list.append(('xgb', trained['XGBoost']))
if 'LightGBM' in trained:
    ens_list.append(('lgbm', trained['LightGBM']))

ensemble = VotingClassifier(estimators=ens_list, voting='soft', n_jobs=-1)
t0 = time.time()
ensemble.fit(X_bal, y_bal)
pred_ens = ensemble.predict(X_test_s)
proba_ens = ensemble.predict_proba(X_test_s)
m = metrics(y_test, pred_ens, proba_ens, "Proposed Voting Ensemble", time.time()-t0)
m['Category'] = 'Proposed'
results.append(m)
print(f"    Acc: {m['Accuracy']:.4f} | F1: {m['Macro_F1']:.4f} | MCC: {m['MCC']:.4f}")

```

Figure 5.1: Proposed Voting Ensemble Model

(d) Proposed KD Student (hard+soft distillation).

A smaller student MLP is trained via a custom Distiller class, combining:

- A hard loss: sparse categorical cross-entropy against ground-truth labels, and
- A soft loss: KL divergence between teacher and student probability distributions under temperature scaling, multiplied by temp^2 .

The combined objective is:

$$[\mathcal{L}] = \alpha \cdot [\mathcal{L}]_{\text{hard}} + (1-\alpha) \cdot [\mathcal{L}]_{\text{soft}}]$$

with $\text{kd_temp}=4.0$ and $\text{kd_alpha}=0.3$, trained for $\text{epochs_kd}=15$.

Key implementation excerpt (KD training step, shortened):

```

t_log = tf.stop_gradient(teacher(x, training=False))
with tf.GradientTape() as tape:
    s_log = student(x, training=True)
    hard = CE(y, s_log)
    soft = KL(softmax(t_log/T), softmax(s_log/T)) * T**2
    loss = alpha*hard + (1-alpha)*soft

```

This excerpt reflects the executed logic, including temperature scaling and the weighted blend between hard and soft objectives.

5.4 Artefact saving and reproducibility

5.4.1 Saved models and preprocessors

At the end of execution, the pipeline persists all core artefacts to the configured output directory (/content/outputs_final). Saved items include:

- teacher.keras
- student.keras
- ensemble.joblib
- label_encoder.joblib
- scaler.joblib

This artefact set is sufficient to reproduce inference for the best-performing proposed approach (voting ensemble) and to rerun deep learning inference given the saved scaler and label encoder.

5.4.2 Results table and figures

The results list is aggregated into a ranked DataFrame sorted by Macro-F1, written to disk as benchmark_results.csv. In addition, the pipeline generates two key figures:

- comparison.png: horizontal bar comparisons for Accuracy, Macro-F1, and MCC across all models.
- confusion_matrix.png: a normalized confusion matrix for the proposed voting ensemble computed over the most frequent labels (top 12 by support in the test set).

These outputs are explicitly logged in the notebook's final save confirmation.

5.4.3 Reproducibility scope

The implementation fixes random seeds and uses a single held-out test split (with a further validation split for early stopping and Optuna scoring). This provides a repeatable procedure within the same environment. However, strict determinism is not guaranteed across different Colab sessions due to potential nondeterministic GPU kernels and backend-level parallelism. This limitation is recorded as part of the broader reproducibility discussion in the evaluation chapter.

5.4.4 Confusion matrix artefact generation and interpretation

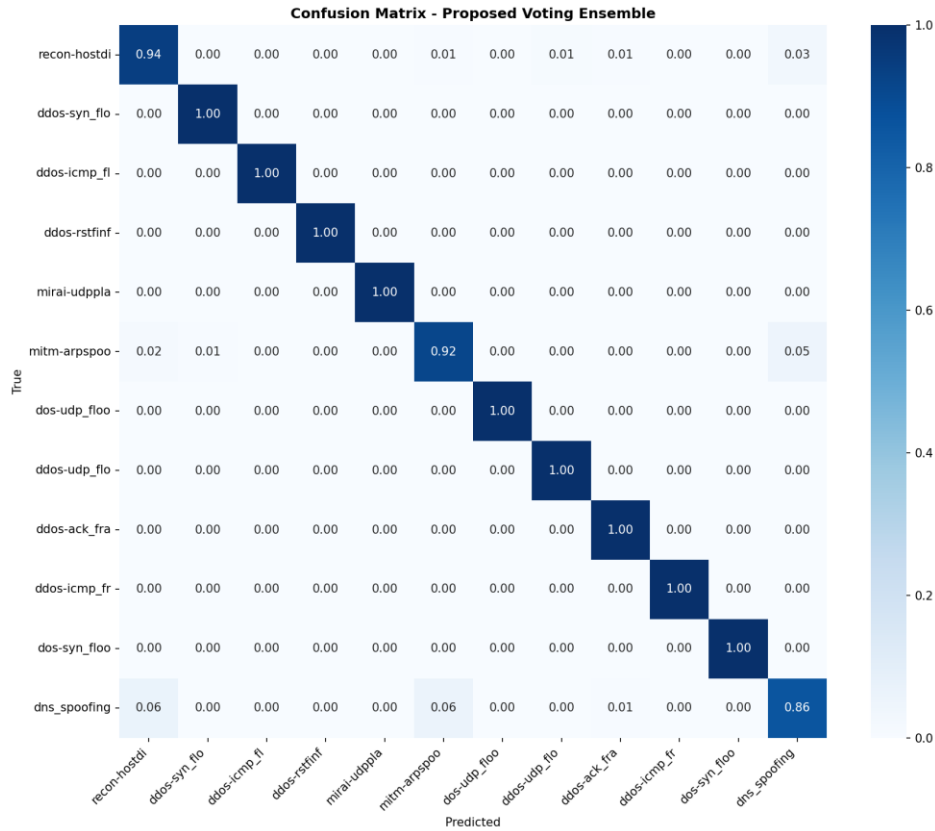


Figure 5.2: Confusion Matrix

In addition to exporting aggregate metrics (Accuracy, Macro-F1 and MCC), the evaluation pipeline produces a confusion matrix for the proposed/best-performing model on the held-out test split to enable class-wise inspection of detection behavior. The confusion matrix is computed by cross-tabulating the model's predicted labels against the ground-truth labels, forming an $N \times N$ matrix where N is the number of evaluated classes after filtering. Diagonal cells represent correct classifications for each class, while off-diagonal cells quantify specific misclassification

pathways (i.e., which attack classes are most frequently confused with others). This output is especially important in multi-class IoT IDS settings because two models with similar Macro-F1 can fail in very different ways at the class level; the confusion matrix therefore provides a concrete basis for diagnosing minority-class weaknesses, overlapping class signatures, and systematic bias toward dominant classes.

Reading the matrix: Rows correspond to the true class labels and columns correspond to predicted labels. High values on the diagonal indicate strong per-class recognition, whereas concentrated off-diagonal mass indicates consistent confusion between particular class pairs. In practice, the most informative patterns are: (i) whether minority classes are being absorbed into high-frequency classes (a typical imbalance symptom), and (ii) whether confusion occurs between semantically related attack types (e.g., scanning-style behaviours being confused with certain reconnaissance-labelled classes), which can indicate feature-space overlap. The confusion matrix is therefore included here as both a reproducibility artefact and a compact interpretive figure that complements the ranked benchmark results.

5.5 Implementation challenges and fixes

Several practical challenges emerged in making the benchmark runnable end-to-end in a constrained notebook environment:

1. **Unreliable cloud filesystem I/O (Drive):** the retry + remount design is a direct response to intermittent OSError events observed when copying large partitions.
2. **Dataset scale management:** rather than loading full partitions, the loader enforces per-class caps and a global target row budget, enabling the experiment to run on commodity Colab resources while still retaining multi-class structure.
3. **Model runtime feasibility:** KNN is trained on a subsample to avoid quadratic scaling effects, allowing it to remain in the benchmark without dominating runtime.
4. **Deep learning instability risk:** early stopping is used uniformly in the DL suite to reduce over-training risk, and GPU memory growth is enabled to reduce crash likelihood.

5. **Observed failure mode (Attention MLP):** the attention-based MLP is implemented but produces near-zero performance in final results, indicating a likely architectural or optimization mismatch for this tabular setting under the given training budget.

CHAPTER 6: EVALUATION OF RESULTS

6.1 Summary of the Experiment

This chapter evaluates the empirical performance of all benchmarked models using the final results recorded in `benchmark_results.csv`. The experimental dataset was constructed from the CICIoT2023 CSV partitions using the pipeline described in Chapters 4-5. Macro-averaged measures were intentionally prioritized because intrusion detection datasets typically contain class imbalance and heterogeneous attack distributions, macro-averaging prevents dominant classes from masking poor minority-class detection performance. This is consistent with guidance that macro-averaging provides a clearer view of per-class performance in imbalanced settings [49]. The inclusion of MCC further strengthens the analysis because MCC behaves robustly across class imbalance and captures the quality of multi-class prediction beyond accuracy alone.

Finally, runtime was treated as a first-class criterion because best IDS models in operational environments must balance detection quality with computational constraints (including near-real-time response requirements and resource-limited deployment contexts).

6.2 Overall Benchmark Results and Ranked Performance

Across the complete benchmark set, tree-based ensemble approaches dominated the top of the leaderboard. The top-performing model overall was the Proposed Voting Ensemble, achieving:

- Accuracy = 0.9528
- Macro-F1 = 0.9441
- MCC = 0.9508
- Training time = 224.0 seconds

The next closest competitors were Bagging (Decision Tree) and XGBoost, both of which achieved very similar Macro-F1 and MCC but at materially lower training times. This indicates a classical diminishing returns pattern, marginal improvements in Macro-F1 and MCC required substantial additional training cost.

Rank	Model	Category	Accuracy	Macro-F1	MCC	ROC-AUC	Balanced Acc.	Time (s)
1	Proposed Voting Ensemble	Proposed	0.9528	0.9441	0.9508	0.9988	0.9441	224.0
2	Bagging (DT)	Ensemble ML	0.9510	0.9417	0.9489	0.9987	0.9417	95.8
3	XGBoost	Ensemble ML	0.9497	0.9416	0.9476	0.9987	0.9416	52.9
4	Gradient Boosting	Ensemble ML	0.9506	0.9391	0.9484	0.9986	0.9391	2835.6
5	Proposed Tuned HistGB	Proposed	0.9494	0.9365	0.9472	0.9984	0.9365	80.8
6	LightGBM	Ensemble ML	0.9486	0.9360	0.9464	0.9984	0.9360	116.2
7	Decision Tree	Traditional ML	0.9413	0.9318	0.9388	0.9968	0.9318	2.2
8	CatBoost	Ensemble ML	0.9445	0.9317	0.9421	0.9976	0.9317	123.5
9	Random Forest	Ensemble ML	0.9415	0.9120	0.9390	0.9979	0.9120	501.6
10	Extra Trees	Ensemble ML	0.8608	0.8363	0.8558	0.9901	0.8363	418.7

Table 6.1: summarizes the top 10 models from the final ranked results

Two immediate findings emerge:

1. Macro-F1 and Balanced Accuracy are numerically identical in this benchmark output. This is consistent with a balanced, per-class averaging approach, but it should be explicitly verified in the implementation code for the final write-up to avoid metric-definition ambiguity.
2. The benchmark suggests tree/ensemble methods are exceptionally effective on this engineered tabular feature representation, substantially outperforming most deep learning architectures.

6.3 Visual Analytics and Figure-Based Interpretation

6.3.1 Cross-model metric profiles (Accuracy, Macro-F1, MCC)

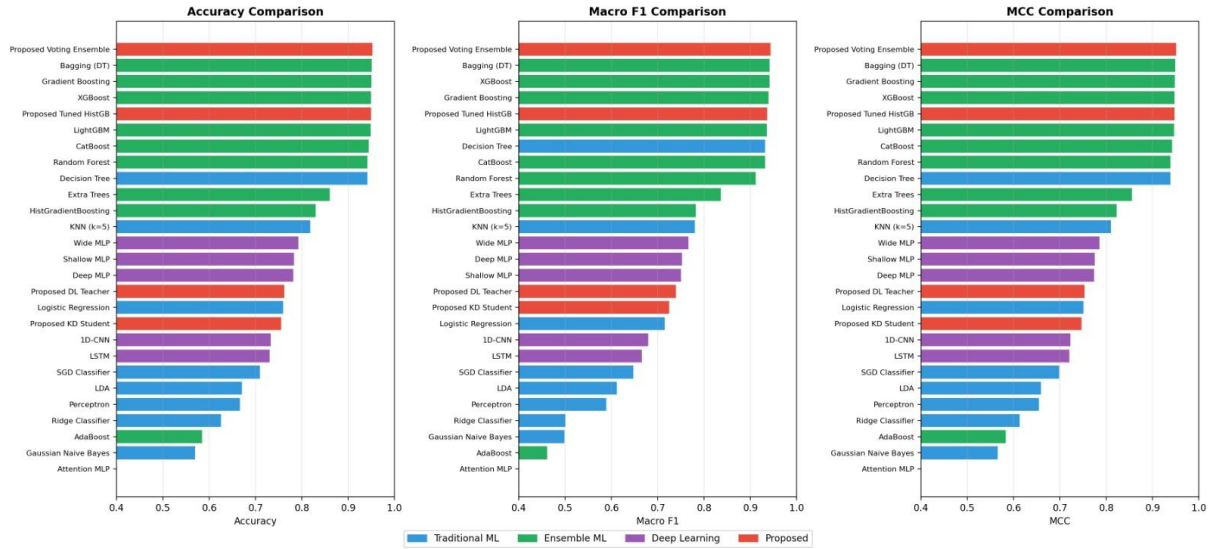


Figure 6.1: Accuracy, Macro F1 and MCC comparison

This figure provides a direct comparison of Accuracy, Macro-F1, and MCC for all evaluated models, with colours indicating model family (Traditional ML, Ensemble ML, Deep Learning, and the Proposed methods). The visual pattern is consistent across all three panels. The leading group is dominated by ensemble-based learners, and the Proposed Voting Ensemble appears at (or very near) the top across each metric. In practical terms, this indicates that the proposed approach is not simply optimizing one headline metric but maintaining a strong balance under class imbalance, as reflected by Macro-F1 and MCC.

A key interpretation of this Figure is the clustering effect of model families. The ensemble ML group occupies the upper band across metrics, suggesting that boosted/bagged tree-based learners are well matched to the tabular, mixed-distribution nature of IoT flow features. In contrast, the deep learning models form a mid-tier band, while they can achieve reasonable scores, they do not reach the same consistently high Macro-F1/MCC levels as the strongest tree ensembles under the current feature representation and training regime. Traditional linear baselines and simpler probabilistic models trend lower, which is expected when decision boundaries are non-linear and the class structure is heterogeneous.

Importantly, It also makes visible a methodological point that is easy to miss in tabular reporting. Some models that are acceptable on accuracy are visibly weaker on Macro-F1 and MCC. This is precisely why the dissertation prioritises Macro-F1 and MCC for a multi-class IoT IDS setting, where minority attack categories must be treated as first-class performance targets rather than being diluted by majority-class dominance.

6.3.2 Macro-F1 ranking clarity: leaderboard view

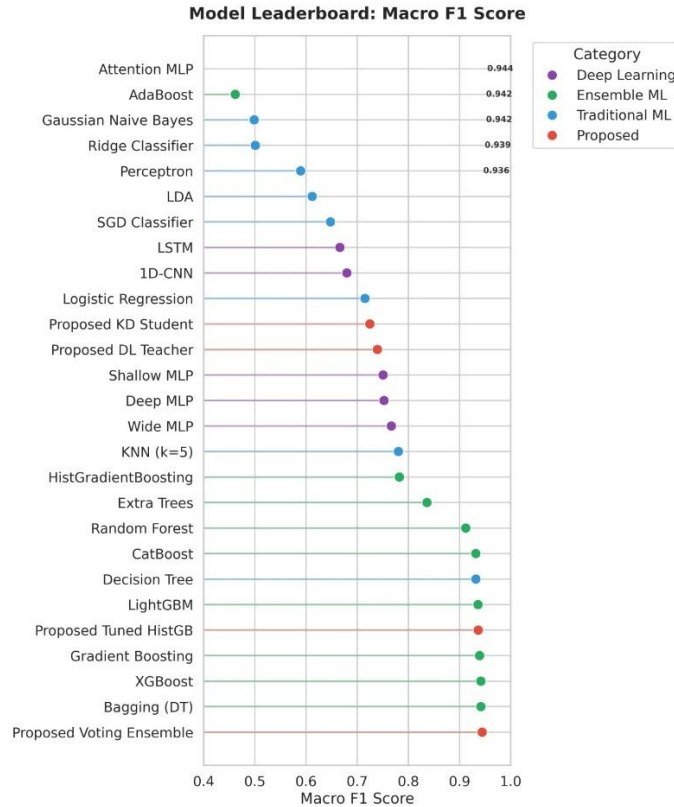


Figure 6.2: Model Leaderboard - Macro F1 Score

The leaderboard confirms a top cluster around the highest Macro-F1 values, led by the Proposed Voting Ensemble and closely followed by strong ensemble baselines (e.g., bagging and gradient boosting variants). Below this cluster, the distribution separates into mid-tier models (including several deep learning configurations and simpler ensembles) and lower-tier baselines. In other words, the ranking view reinforces that the best results are not isolated to one model, but belong to a small, consistent family of high-performing approaches - supporting the dissertation's emphasis on comparative benchmarking rather than single-model reporting.

6.3.3 Detailed metrics heatmap

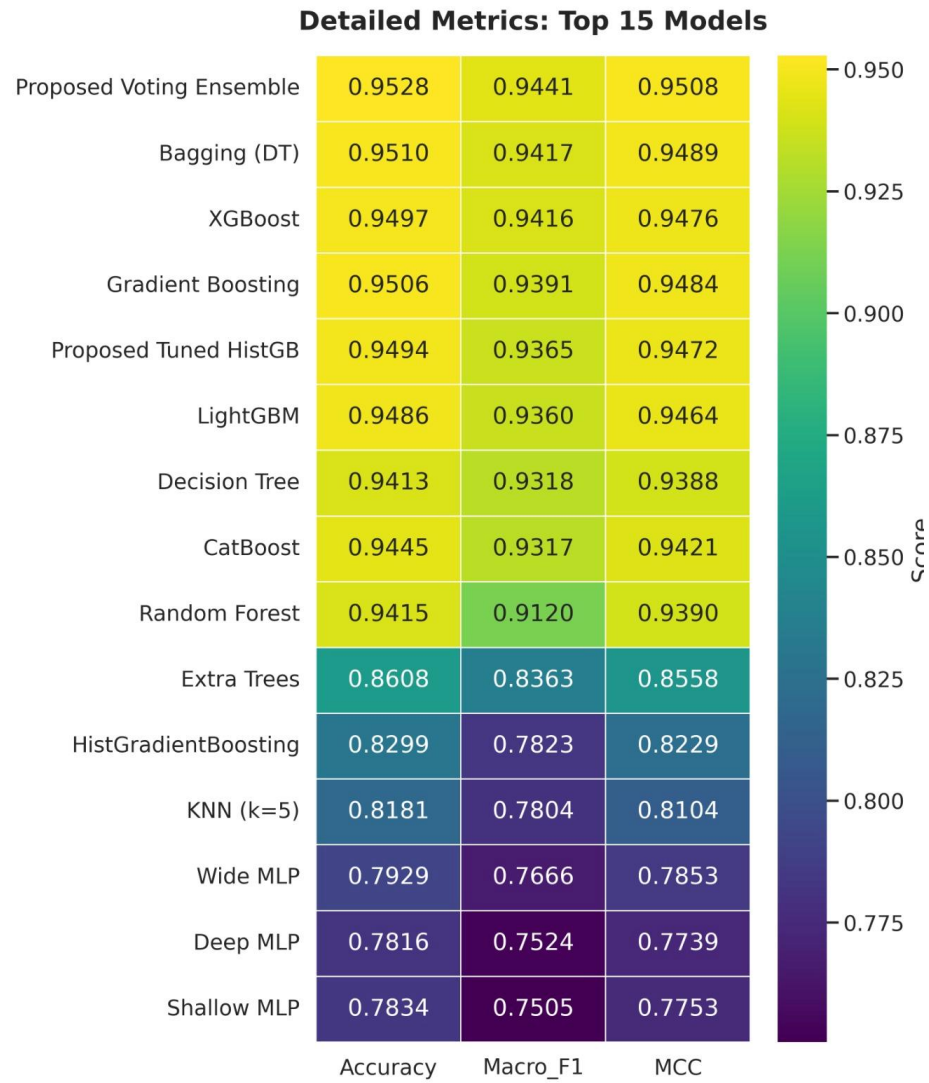


Figure 6.3: Detailed metrics heatmap

The Detailed Metrics: Top 15 Models heatmap shows the metric cluster structure clearly. Most high-performing models are concentrated in the tree/ensemble family, while deep learning models occupy the mid-table band (Macro-F1 ≈ 0.75 – 0.77) except for a severe outlier.

6.3.4 Winner Analysis: Proposed Voting Ensemble

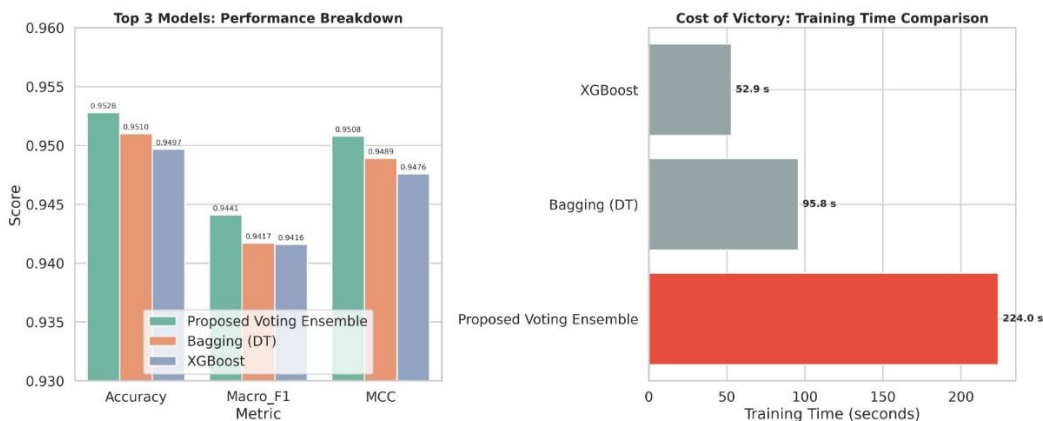


Figure 6.4: Winner analysis and cost-of-victory

The Winner analysis: Proposed Voting Ensemble figure shows the top-3 models (Proposed Voting Ensemble, Bagging (DT), XGBoost) compared across Accuracy, Macro-F1, and MCC, alongside training time. The plot makes the core conclusion visually unambiguous, the proposed ensemble wins on all three primary metrics, but requires the largest training time among the three. The time bars quantify the deployment trade-off. The proposed ensemble requires roughly 2.3× the Bagging time (224.0s vs 95.8s) and about 4.2× the XGBoost time (224.0s vs 52.9s) for comparatively small metric gains.

6.3.5 Best in Class Comparison

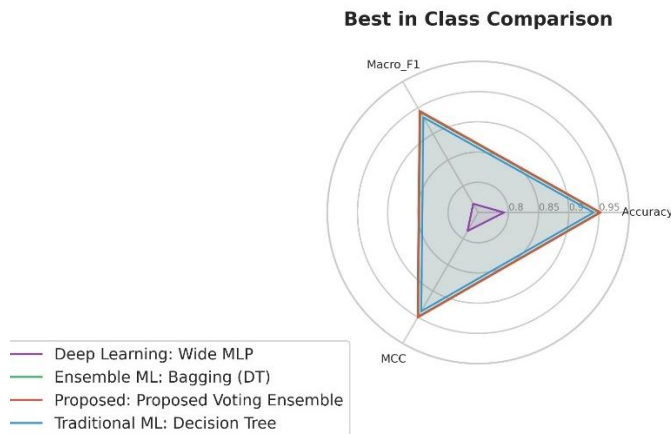


Figure 6.5: Best-in-class radar comparison

The radar chart contrasts one representative best model from each category. It highlights two practical points:

- Deep learning’s best representation remains materially behind on Macro-F1 and MCC in this tabular setting.
- The proposed approach dominates accuracy and MCC marginally, indicating that improvements are concentrated at the ceiling end where gains become harder.

6.3.6 Model Tier List (Based on Composite Score)

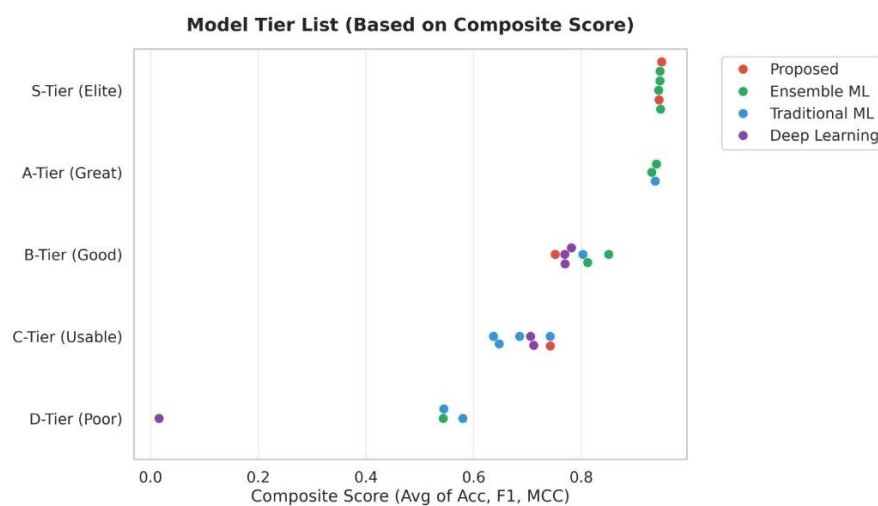


Figure 6.6: Tier list by composite score

The Model Tier List (Based on Composite Score) ranks models using the average of Accuracy, Macro-F1, and MCC. This tiering aligns with the benchmark table. The highest composite scores belong to the Proposed/Ensemble tree-based models, deep learning models cluster in a lower tier. This provides an intuitive benchmark digest for readers who prefer a single-score summary, but it should be framed as interpretive rather than definitive because composite metrics impose equal weighting across measures.

6.3.7 Metric Consistency by Algorithm Category

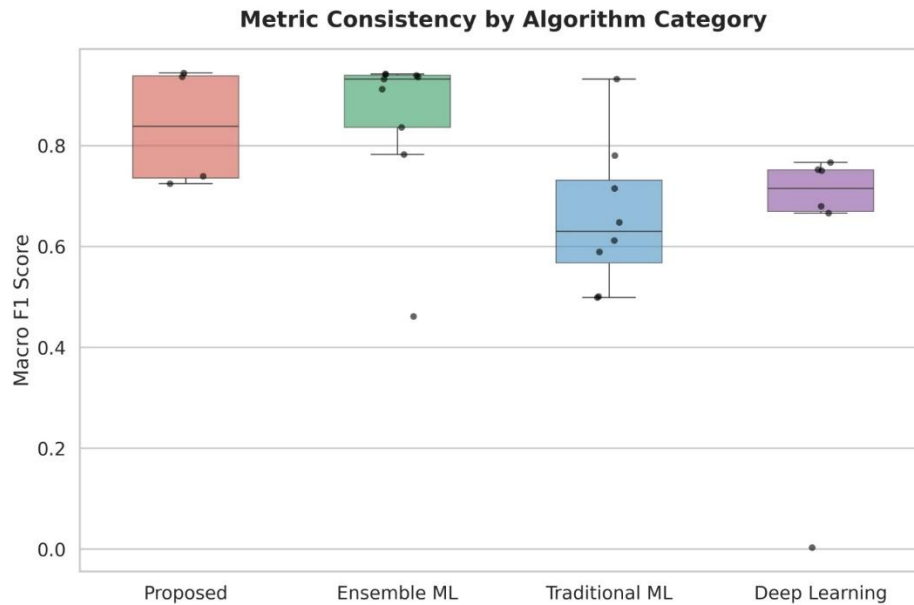


Figure 6.7: Metric consistency by algorithm category

The category-level boxplot of Macro-F1 indicates:

- Ensemble ML and Proposed categories have consistently high Macro-F1 distributions.
- Traditional ML has broader dispersion (some mid-performing linear models, plus a very strong single Decision Tree).
- Deep Learning shows the greatest instability due to one catastrophic failure (the Attention MLP), pulling the distribution downward and widening variance.

6.3.8 Performance-time trade-off: efficiency frontier

Here are the plots each model as a point in the performance–time plane, again colour-coded by model family. The shaded region highlights the fast side of the time axis.

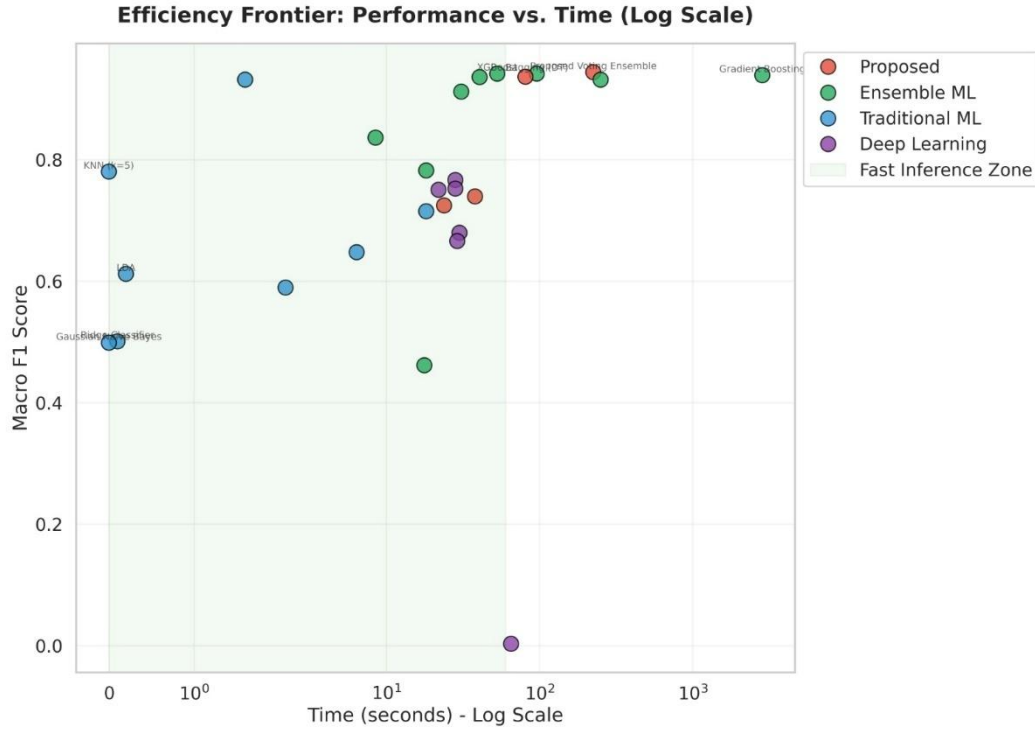


Figure 6.8: Efficiency Frontier - Performance vs Time (Log Scale)

Several results follow directly from this frontier view:

- Tree ensembles occupy the high-performance region, but spread widely on training time. Some ensembles deliver near-top Macro-F1 within moderate time, while others require substantially higher time budgets for marginal gains. This supports the dissertation's argument that runtime reporting is not optional. Two models with similar Macro-F1 may differ materially in operational cost.
- The Proposed Voting Ensemble lies at the upper frontier, indicating that it competes at the best Macro-F1 level. However, it sits at a higher time cost than single-model baselines because it aggregates multiple learners. This is the expected cost of victory for ensembles, performance gains are bought by additional training and maintenance overhead.
- Traditional and deep learning models populate the mid-performance zone under the present pipeline. Some are comparatively fast but plateau lower on Macro-F1, suggesting their suitability mainly when compute budgets are strict or when a simpler model is required for operational reasons (e.g., rapid retraining cycles or constrained environments).

6.3.9 Pareto optimization: best achievable for a time budget

This figure formalizes the efficiency frontier by marking Pareto-optimal points and drawing the Pareto frontier. A model is Pareto-optimal if no other model achieves both higher Macro-F1 and lower training time.

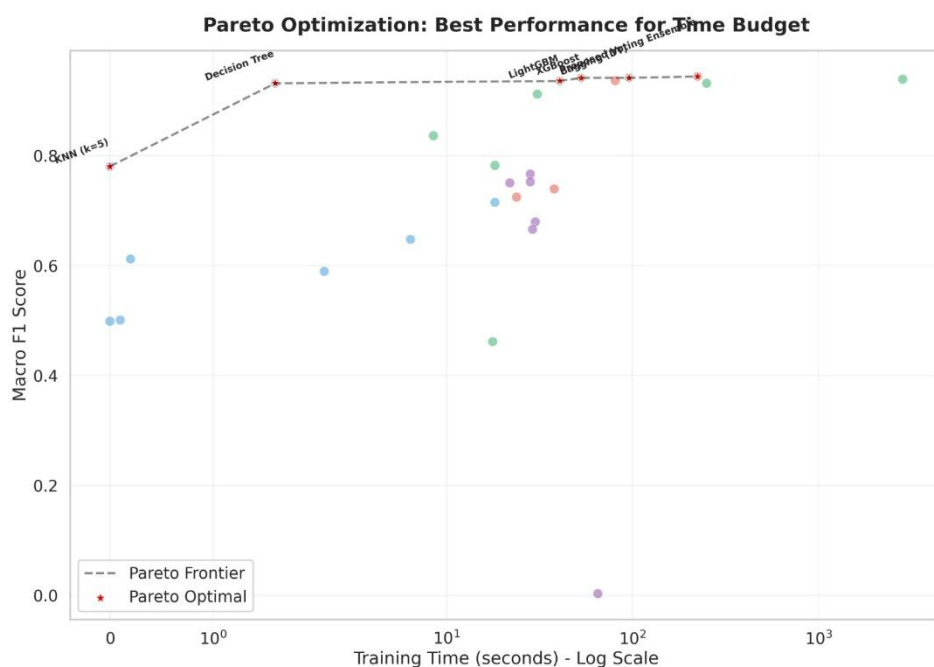


Figure 6.9: Pareto Optimization

This view produces the most actionable recommendation logic for practitioners:

- At very small time budgets, fast learners (e.g., simpler traditional models) can be Pareto-optimal because they train quickly while still achieving non-trivial Macro-F1. They are reasonable choices for rapid iteration, frequent retraining, or constrained compute settings.
- At moderate time budgets, certain ensemble learners dominate: they deliver large jumps in Macro-F1 without extreme training costs, making them strong default candidates for production benchmarking.
- At higher time budgets, the Proposed Voting Ensemble becomes justifiable as a Pareto-optimal (or near-optimal) choice if the marginal Macro-F1 improvements are operationally valuable (e.g., the environment demands consistent minority-class detection and the retraining schedule tolerates longer training).

6.4 Interpretation of Findings

6.4.1 Why the Best Models Performed Well

The benchmark strongly favors decision-tree-based ensembles, which is consistent with the broader literature that tree ensembles (bagging and boosting families) are highly competitive on tabular data representations due to their ability to model nonlinear feature interactions, handle heterogeneous feature scales, and capture conditional decision boundaries without requiring extensive representation learning. Bagging and boosting are also theoretically and empirically well-established as techniques that improve stability and generalization when base learners are high-variance estimators (such as decision trees) [50].

In particular:

- Bagging (DT) performs extremely well because bootstrap aggregation reduces variance and stabilizes decision boundaries in a multi-class setting.
- Boosting family models (including gradient boosting and XGBoost-style approaches) remain strong because they iteratively correct residual error and can express complex class boundaries in a controlled additive model structure.
- The proposed system’s top performer Proposed Voting Ensemble benefits from combining high-performing but differently biased learners, consistent with ensemble theory that diverse but accurate base models can outperform any single constituent through error compensation.

6.4.2 Runtime Performance Trade-offs and Practical Winners

A key contribution of this dissertation is reporting runtime alongside detection quality. The results suggest at least three practically meaningful winners depending on deployment constraints:

1. Maximum detection quality: Proposed Voting Ensemble

This model is the best overall on Macro-F1 and MCC. However, training time is non-trivial (224s), and the implementation complexity (multiple models + voting integration) is higher.

2. Best balance of performance and cost: Proposed Tuned HistGB

The tuned HistGB model achieves Macro-F1 0.9365 with training time 80.8s, delivering a strong Pareto compromise. The proposed ensemble improves Macro-F1 by only 0.0076 ($0.9441 - 0.9365$) but requires $\sim 2.8\times$ the training time ($224.0/80.8$). In many real-time IDS engineering contexts, this would justify selecting the tuned HistGB model as the deployment candidate.

3. Fastest high-performing baseline: XGBoost (or Bagging DT)

XGBoost is the most time-efficient among the top-3 (52.9s) while achieving Macro-F1 0.9416, essentially within rounding distance of Bagging DT and close to the proposed winner. This is consistent with the algorithm’s design goal of scalable, optimized boosting.

By contrast, Gradient Boosting achieved strong accuracy and Macro-F1 but required 2835.6 seconds (approximately 47 minutes). This is best interpreted as a computational scalability limitation of that specific configuration and implementation style, rather than a fundamental statement that boosting is inherently slow, it reinforces why optimized boosting frameworks are generally preferred for practical use.

6.4.3 Understanding Weak and Failed Models

Several models underperformed sharply, providing insight into what does not translate well to this specific multi-class tabular IoT IDS setting.

- **AdaBoost**

This performs poorly (Macro-F1 ≈ 0.46). A plausible explanation is that AdaBoost is sensitive to noisy labels and hard-to-separate classes. It can over-focus on misclassified samples, magnifying instability in complex multi-class intrusion data. This aligns with known behavior in ensemble literature where boosting can degrade under noise or severe class complexity [51].

- **Attention MLP collapse (Macro-F1 near zero)**

The Attention MLP shows catastrophic failure (Accuracy ≈ 0.044 , Macro-F1 ≈ 0.003), which is far beyond ordinary underperformance and suggests instability or a pipeline mismatch. Plausible, testable explanations include:

- (i) Optimization divergence due to learning rate/initialization.
- (ii) Mismatch between attention mechanism assumptions and the feature topology of engineered tabular network traffic.
- (iii) Unintended numerical issues (e.g., remaining NaNs after coercion).
- (iv) Label encoding misalignment between training and evaluation.

- **Deep learning generally behind tree ensembles**

Even the best deep learning model (Wide MLP, Macro-F1 ≈ 0.7666) remains well below the leading tree ensembles. This is consistent with wider benchmarking evidence that, on many structured/tabular problems, classical ensemble learners remain strong competitors and often exceed generic neural architectures unless careful representation learning, architecture tuning, or feature construction is performed.

6.4.4 Interpreting High ROC-AUC Values

The ROC-AUC values for top models are extremely high (≈ 0.998), which indicates that, under the chosen split strategy, class separation is strong. However, in intrusion detection, very high AUC values can be inflated by data leakage or overly similar train/test distributions, especially when random stratified splits are applied on network traffic where flows from the same capture context may appear across splits. A recent CIC-IoT-2023-focused study explicitly examines data leakage risks and demonstrates why evaluation design matters for trustworthy generalization claims [51].

Accordingly, the correct interpretation is, the reported AUC is valid for this experimental protocol, but stronger claims about real-world generalization would require more robust split strategies (e.g., time-based or device-based partitioning).

6.5 Statistical Robustness and Evaluation Limitations

The benchmark results represent a single final run under a fixed stratified split (with a separate validation set used for deep learning early stopping and KD training control). While this is standard in many applied MSc projects, it limits statistical confidence.

Two limitations should therefore be explicitly stated in the dissertation:

1. **No confidence intervals or repeated runs.**

Without repeated stratified runs (or cross-validation), it is not possible to quantify variance and establish whether the small performance margins between the top three models are statistically robust.

2. **Split strategy may overestimate generalization**

Random stratified splitting is convenient and reproducible, but for network traffic it can unintentionally create train/test similarity. As noted above, leakage-aware evaluation is a recognized concern in IDS research and should be treated as a future-work priority rather than ignored [51].

A defensible future-work plan would include repeated stratified runs with different seeds and at least one non-random split (e.g., temporal split), reporting mean $\pm$ standard deviation for Macro-F1 and MCC.

6.6 Comparison to Recent Literature

Direct comparisons across IoT IDS papers must be treated cautiously because results vary substantially depending on dataset partitioning, selected classes, feature engineering, sampling/capping, and whether the evaluation protocol controls for leakage.

The CICIOT2023 dataset paper provides the authoritative dataset context and emphasizes the breadth and realism of the dataset for IoT attack detection research [3]. However, many published results differ in task formulation (binary vs multi-class), label consolidation, and feature selection.

A recent 2025 benchmark study on CICIOT2023 reports macro-level results that are materially lower than the top results achieved here (e.g., macro F1 around the 0.73 range in their reported

configuration), but that work uses different class coverage and robustness-driven feature strategies, making direct better/worse claims inappropriate [52]. The appropriate comparison is therefore:

- This dissertation demonstrates that under the defined sampling, class filtering, and engineered tabular pipeline, tree-based ensembles can achieve very high Macro-F1 and MCC.
- The literature indicates that absolute values are highly sensitive to evaluation design and robustness constraints, and that leakage-aware protocols may reduce headline metrics while improving realism [51].

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS

7.1 Restatement of Aim and Objectives, and What Was Achieved

The aim of this dissertation was to critically analyze and empirically evaluate traditional machine learning, ensemble learning, deep learning, and a proposed voting-ensemble approach for real-time-oriented IoT intrusion detection, using a reproducible benchmark pipeline with metrics suitable for imbalanced multiclass data

This aim was operationalized through three objectives that are met as follows. First, the work critically analyzed current IoT IDS modelling practice, highlighting why imbalanced multi-class settings demand metrics beyond accuracy (notably macro-F1 and MCC). Second, the study designed and developed a complete, repeatable end-to-end pipeline (data ingestion, capping/filtering, stratified splitting, scaling, model training, and artefact persistence) that can be rerun on additional CICIOT2023 partitions with minimal modification. Third, the study evaluated a broad benchmark suite across traditional ML, ensemble ML, deep learning, and proposed approaches-reporting not only effectiveness metrics (Accuracy, Macro-F1, MCC, ROC-AUC) but also computational cost (training time), which is essential for operational decision-making.

Overall, the dissertation achieved a coherent alignment between research purpose, experimental design, and evidence-based conclusions. The empirical results substantiate which model families are most reliable for the chosen tabular IoT traffic representation, and the analysis clarifies when winning performance is worth the computational cost.

7.2 Key Findings

The central finding is that tree-based ensemble methods dominate performance in this CICIOT2023 multi-class benchmark under the implemented preprocessing and evaluation protocol. This outcome is consistent with established ensemble theory aggregating diverse learners reduces variance and improves generalization, while boosting iteratively reduces bias through additive modelling. In many structured/tabular tasks, these properties give ensembles a practical advantage over generic deep networks.

Empirically, the Proposed Voting Ensemble achieved the best overall detection quality (Accuracy 0.9528, Macro-F1 0.9441, MCC 0.9508). However, the results also show a decisive trade-off, the performance margin over strong baselines is incremental, while training cost increase substantially. In particular, XGBoost designed for scalable tree boosting achieved Macro-F1 0.9416 with markedly lower training time, demonstrating a strong engineering-optimal option when cost constraints matter.

A further practical insight is that the Proposed Tuned HistGB provides an especially strong balance between quality and cost. Its Macro-F1 (0.9365) is close to the winner with substantially lower training time, placing it on a favorable Pareto front for deployment decisions.

Deep learning models were generally inferior in this tabular setting. While this does not invalidate deep learning for IDS, it indicates that under the current feature representation and tuning scope neural architectures did not consistently realize benefits over ensembles. This is aligned with broader findings that deep learning may require specialized representation learning, careful architecture search, or larger/cleaner feature structures to surpass strong tree ensembles on tabular data.

Finally, the benchmark surfaced an important robustness observation, one deep model (Attention MLP) failed catastrophically, reinforcing that neural IDS pipelines can be sensitive to configuration and data conditions. In contrast, tree ensembles showed strong performance consistency across top-ranking candidates.

7.3 Contributions

This dissertation makes three contributions.

First, it provides a transparent benchmark of model families for CICIoT2023 multi-class IoT intrusion detection, reporting macro-sensitive metrics (macro-F1 and MCC) as primary decision criteria rather than relying on accuracy alone.

Second, it proposes and empirically validates a high-performing ensemble-based approach (voting ensemble) and demonstrates that an optimized single-model alternative (tuned HistGB) can deliver near-winning performance at lower computational cost. The work therefore contributes not only

best score reporting but also an applied decision framework for selecting models under runtime constraints.

Third, it contributes a reproducible implementation approach, dataset handling decisions (partition selection, rare-class filtering, capping strategy), stratified splitting, metric suite, runtime logging, and artefact saving are described at a level that supports replication on new partitions and controlled extensions.

7.4 Recommendations for Practitioners

For practitioners building IoT IDS capabilities using CICIoT2023-style tabular features, the evidence supports the following recommendations.

Model selection should prioritize macro-F1 and MCC over raw accuracy for multi-class intrusion detection where class imbalance is expected, macro-averaging prevents majority classes from masking weak minority-class detection, while MCC provides a robust quality signal across imbalanced settings.

For deployment-oriented choices, teams should treat the benchmark as defining three credible options. Where maximum detection quality is required and offline training cost is acceptable, a voting ensemble is appropriate. Where computational efficiency is a priority, a tuned gradient-boosted tree model or XGBoost should be preferred, noting XGBoost’s design goal of scalability and efficiency

Evaluation design must explicitly consider data leakage risk. Random stratified splitting can unintentionally inflate performance in network datasets when similar flows appear across splits, leakage-aware strategies (time-based or source-based splits) are recommended for higher-confidence generalization claims. Practically, this means adopting stronger split protocols for any model destined for operational use, even if headline metrics reduce.

Finally, if resource-constrained deployment (edge/IoT gateway) is required, model compression pathways particularly knowledge distillation combined with quantization are pragmatic options. These approaches are explicitly motivated by computational and energy constraints in edge IDS contexts.

7.5 Limitations

The benchmark reflects a single final run under a fixed stratified split. Without repeated runs or cross-validation, there is no quantified variance or confidence interval, and the small margins among the top three models cannot be claimed as statistically robust.

The dataset construction applied partition selection, class filtering, and capping to control scale and balance practical constraints. While necessary for feasibility, these steps can change class coverage and the effective difficulty of the classification task relative to full dataset scenarios.

The evaluation emphasized training time but did not report inference latency, memory footprint, or throughput metrics that often matter more than training cost in real-time IDS deployments.

Finally, the protocol does not explicitly test concept drift. Real IoT environments are non-stationary, models that perform well on static splits may degrade over time. This is a recognized challenge in IDS research, motivating drift-aware evaluation and adaptation strategies.

7.6 Future Work

Methodologically, the benchmark should be extended with repeated stratified runs and at least one leakage-aware split design, complemented by reporting mean and standard deviation for Macro-F1 and MCC. The dataset scope should be broadened by incorporating more CICIOT2023 partitions and, where feasible, cross-dataset validation to test robustness beyond a single corpus.

To address non-stationarity, a drift-aware evaluation track should be introduced, drawing on drift-focused IDS research that analyses concept and feature dynamics in changing environments and on practical drift localization/adaptation methods for IoT data streams. This can be complemented by integrating unsupervised drift detection approaches where labels are not available in real time, consistent with modern drift detection survey recommendations.

From a deployment perspective, future work should include edge-oriented optimization, combining distillation and quantization and measuring inference latency and energy efficiency, aligning with recent edge IDS work that explicitly targets resource constraints.

CHAPTER 8: REFLECTION

This dissertation was ultimately an exercise in disciplined scope control, turning a large, ambiguous domain (IoT intrusion detection) into a defensible, measurable MSc project with a clear pipeline, interpretable results, and transparent limitations. The most important learning was that the research value did not come from adding ever more models or complex techniques, it came from designing a methodology that makes comparisons fair, repeatable, and meaningful.

From a time-management perspective, the project progressed in distinct phases. Starting from confirming a feasible experimental workload within the constraints of Google Colab. Then building a stable data pipeline that could reliably ingest CICIOT2023 partitions without repeated failures. After that training and benchmarking a broad but controlled model suite, and shifting from getting results to explaining what those results mean. The transition from implementation to interpretation required more time than initially expected. Writing a credible evaluation chapter demanded careful attention to what was actually measured (macro-F1 and MCC under a specific split protocol) and what could not be claimed (robust generalization under drift, statistical certainty without repeated runs). This shift improved the quality of the dissertation because it forced explicit differentiation between evidence and inference.

In conducting the research, I learned that good experimental security research is less about technical novelty and more about methodological integrity. The most challenging aspect was not model training itself but ensuring that the pipeline did not accidentally bias outcomes through avoidable issues such as inconsistent preprocessing, unstable label handling, or unreliable data loading. Building robust ingestion (Drive remount/retry logic), enforcing class thresholds, and saving artefacts were not nice to have engineering tasks. They were central to producing results that could be verified. This reinforced a key research lesson in applied cybersecurity analytics, reproducibility is part of the contribution, not an optional add-on.

Another important learning was the practical meaning of evaluation metrics. Before this project, it was easy to treat accuracy as a default measure. Working with imbalanced multi-class intrusion labels made it clear that accuracy can be misleading and that macro-sensitive metrics provide a more defensible account of detection quality. Additionally, including runtime as part of evaluation changed how best was defined. The top-performing model in macro-F1 and MCC was not automatically the most deployable. The best choice depends on operational constraints such as

training budget, retraining frequency, and system resources. This insight made the conclusions more useful for real-world practice.

If I were to repeat the project, I would make three improvements. First, I would plan for repeated stratified runs from the start and reserve time for reporting variance (mean and standard deviation) rather than relying on a single final split. Second, I would introduce at least one leakage-aware split design (such as a temporal split), because network traffic data can produce overly optimistic results under random splitting. Third, I would add inference-time benchmarks and memory profiling to better support real-time claims, since training time alone is not a full proxy for deployment readiness.

Feedback and supervision shaped several decisions in a constructive way. Emphasis on maintaining a practical, implementable experiment influenced the choice to focus on a controlled benchmark rather than pursuing overly complex lab setups that would be difficult to reproduce. Guidance to strengthen the methodology's alignment with literature gaps also reinforced the need to explicitly justify design choices especially metric selection, class handling, and the evaluation protocol rather than presenting the pipeline as purely technical work. In effect, feedback shifted the project from a notebook that runs to a dissertation that explains, defends, and contextualizes what was done and what it means.

In summary, the project strengthened my ability to run an applied machine learning study with cybersecurity relevance, while maintaining academic discipline, clearly stated aims, transparent methods, and conclusions that match the evidence. The most valuable outcome is not only the final performance numbers, but the structured approach to building and evaluating an intrusion detection pipeline in a way that others can understand, replicate, and extend.

References:

- [1] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A survey on intrusion detection system in IoT networks," *Cyber Security and Applications*, vol. 3, Art. no. 100082, 2025, doi: 10.1016/j.csa.2024.100082.
- [2] Chu, P. Jin, H. Qiao, and Q. Feng, "Intrusion detection in the IoT data streams using concept drift localization," *AIMS Mathematics*, vol. 9, no. 1, pp. 1535–1561, 2024, doi: 10.3934/math.2024076.
- [3] E. C. P. Neto *et al.*, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. 23, no. 13, Art. no. 5941, 2023, doi: 10.3390/s23135941.
- [4] J. Opitz and T. Jäger, "A Closer Look at Classification Evaluation Metrics and a Critical Reflection of Common Evaluation Practice," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 820–836, 2024, doi: 10.1162/tacl_a_00675.
- [5] (Springer LNNS) "A multiplicative attention-based deep learning framework for intrusion detection in IIoT networks," *SMARTTECH 2024*, published 2025, doi: 10.1007/978-3-031-91235-1_36.
- [6] "IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic," *Digital Communications and Networks*, 2023, doi: 10.1016/j.dcan.2023.03.008.
- [7] Y. Yin *et al.*, "IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset," *Journal of Big Data*, vol. 10, art. 15, 2023, doi: 10.1186/s40537-023-00694-8.
- [8] S. More *et al.*, "Enhanced intrusion detection systems performance with UNSW-NB15 data analysis," *Algorithms*, vol. 17, no. 2, art. 64, 2024, doi: 10.3390/a17020064.
- [9] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: BoT-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041
- [10] J. L. Leevy *et al.*, "IoT information theft prediction using ensemble feature selection," *Journal of Big Data*, vol. 9, art. 6, 2022, doi: 10.1186/s40537-021-00558-z.
- [11] "An intrusion detection system using BoT-IoT," *Applied Sciences*, vol. 13, no. 9, art. 5427, 2023, doi: 10.3390/app13095427.
- [12] N. Moustafa and J. Slay, "TON_IoT datasets: A new generation of realistic datasets for IoT and IIoT intrusion detection systems," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3022862.
- [13] N. Moustafa *et al.*, "A new distributed architecture for evaluating AI-based security systems in IoT environments," *Sustainable Cities and Society*, vol. 72, art. 102994, 2021, doi: 10.1016/j.scs.2021.102994
- [14] I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S. M. El-Horbaty, "Analysis of ToN-IoT, UNSW-NB15, and Edge-IIoT datasets using deep learning in cybersecurity for IoT," *Applied Sciences*, vol. 12, no. 19, art. 9572, 2022, doi: 10.3390/app12199572.
- [15] M. M. Alani and A. Miri, "Towards an explainable universal feature set for IoT intrusion detection," *Sensors*, vol. 22, no. 15, art. 5690, 2022, doi: 10.3390/s22155690.
- [16] (Springer LNNS) "A multiplicative attention-based deep learning framework for intrusion detection in IIoT networks," *SMARTTECH 2024*, published 2025, doi: 10.1007/978-3-031-91235-1_36.

- [17] H. Fares, N. Aknin, G. Lazrek, and M. Zeroual, "Intrusion detection in IoT environment using hyperparameters tuned machine and deep learning models on the CICIoT2023 dataset," *Informatica*, vol. 49, no. 5, 2025, doi: 10.31449/inf.v49i5.8881
- [18] "Comparative evaluation of a novel IDS dataset for SDN-IoT using deep learning models against InSDN, BoT-IoT, and ToN-IoT," *Measurement: Digitalization*, vol. 4, 2025, art. 100015, doi: 10.1016/j.meadig.2025.100015.
- [19] M. A. Bouke and K. Abdullah, "An empirical assessment of ML models for 5G network intrusion detection: A data leakage-free approach," *e-Prime—Advances in Electrical Engineering, Electronics and Energy*, vol. 7, Art. no. 100590, 2024, doi: 10.1016/j.prime.2024.100590.
- [20] M. A. Shyaa *et al.*, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," *Engineering Applications of Artificial Intelligence*, vol. 133, Art. no. 109143, 2024, doi: 10.1016/j.engappai.2024.109143.
- [21] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996, doi: 10.1007/BF00058655.
- [22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324
- [23] T.-T.-H. Le *et al.*, "Classification and explanation for intrusion detection system based on ensemble trees and SHAP method," *Sensors*, vol. 22, no. 3, Art. no. 1154, 2022, doi: 10.3390/s22031154.
- [24] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001, doi: 10.1214/aos/1013203451.
- [25] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. KDD*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
- [26] L. Aversano, M. L. Bernardi, M. Cimitile, and R. Pecori, "A systematic review on deep learning approaches for IoT security," *Computer Science Review*, vol. 40, Art. no. 100389, 2021, doi: 10.1016/j.cosrev.2021.100389.
- [27] M. Hossain *et al.*, "Deep learning-based intrusion detection for IoT networks: A scalable and efficient approach," *EURASIP Journal on Information Security*, 2025, doi: 10.1186/s13635-025-00202-w.
- [28] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021, doi: 10.1007/s11263-021-01453-z.
- [29] J. Opitz and P. Jäger, "Common evaluation practice is flawed: The case of the classification report," *Transactions of the Association for Computational Linguistics*, 2024, doi: 10.1162/tacl_a_00675.
- [30] D. Chicco, N. Tötsch, and G. Jurman, "The Matthews correlation coefficient (MCC) is more informative than Cohen's kappa and Brier score in binary classification assessment," *IEEE Access*, vol. 9, pp. 78368–78381, 2021, doi: 10.1109/ACCESS.2021.3084050.
- [31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. KDD*, 2019, pp. 2623–2631, doi: 10.1145/3292500.3330701.

- [32] Y. S. Park, L. Konge, and A. R. Artino Jr., "The positivism paradigm of research," *Academic Medicine*, vol. 95, no. 5, pp. 690–694, 2020, doi: 10.1097/ACM.0000000000003093.
- [33] E. C. P. Neto *et al.*, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. 23, no. 13, Art. no. 5941, 2023, doi: 10.3390/s23135941
- [34] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proc. KDD*, 2019, doi: 10.1145/3292500.3330701.
- [35] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, 2020, doi: 10.1186/s12864-019-6413-7
- [36] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960, doi: 10.1177/001316446002000104
- [37] O. Rainio, J. Teuvo, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, Art. no. 6086, 2024, doi: 10.1038/s41598-024-56706-x
- [38] S. Kapoor and A. Narayanan, "Leakage and the reproducibility crisis in machine-learning-based science," *Patterns*, 2023, doi: 10.1016/j.patter.2023.100804.
- [39] "The TON_IoT Datasets | UNSW Research," research.unsw.edu.au. <https://research.unsw.edu.au/projects/toniot-datasets>
- [40] "The Bot-IoT Dataset | UNSW Research," research.unsw.edu.au. <https://research.unsw.edu.au/projects/bot-iot-dataset>
- [41] UNSW, "The UNSW-NB15 Dataset | UNSW Research," research.unsw.edu.au, Jun. 02, 2021. <https://research.unsw.edu.au/projects/unsw-nb15-dataset>
- [42] "IoT Dataset 2023 | Datasets | Research | Canadian Institute for Cybersecurity | UNB," www.unb.ca. <https://www.unb.ca/cic/datasets/iotdataset-2023.html>
- [43] M. Thölke *et al.*, "Class imbalance should not throw you off balance," *NeuroImage*, vol. 280, 2023, doi: 10.1016/j.neuroimage.2023.120253.
- [44] V. R. Joseph and A. Vakayil, "SPlit: An Optimal Method for Data Splitting," *Technometrics*, 2021, doi: 10.1080/00401706.2021.1921037.
- [45] O. Rainio, J. Teuvo, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, Art. no. 6086, 2024, doi: 10.1038/s41598-024-56706-x.
- [46] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. KDD*, 2016, doi: 10.1145/2939672.2939785
- [47] M. M. Rahman *et al.*, "Review of machine learning for intrusion detection systems in IoT networks," *Cyber Security and Applications*, 2024, doi: 10.1016/j.csa.2024.100082.
- [48] J. Owusu-Adjei *et al.*, "Is accuracy an adequate metric for class-imbalanced problems?" *PLOS Digital Health*, 2023, doi: 10.1371/journal.pdig.0000290.
- [49] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009, doi: 10.1016/j.ipm.2009.03.002

- [50] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996, doi: 10.1007/BF00058655
- [51] T. G. Dietterich, “Ensemble methods in machine learning,” in *Multiple Classifier Systems (MCS 2000)*, Lecture Notes in Computer Science, vol. 1857, pp. 1–15, 2000, doi: 10.1007/3-540-45014-9_1
- [52] S. Wakili and H. Bakkali, “CICIoT2023: Extended Benchmark and ULTRA-ROBUST Features for IoT Attack Detection,” *Results in Engineering*, 2025, doi: 10.1016/j.rineng.2025.107392
- [53] M. J. Page *et al.*, “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews,” *BMJ*, vol. 372, n71, 2021, doi: 10.1136/bmj.n71.
- [54] R. Alshamrani *et al.*, “Ensemble Learning for Network Intrusion Detection Based on Correlation and Embedded Feature Selection Techniques,” *Computers*, vol. 14, no. 3, Art. no. 82, 2025, doi: 10.3390/computers14030082.
- [55] Google, “colab.google,” *colab.google*, 2024. <https://colab.google/>

Appendix

1. **Source Code: Optimizing-ML-DL-models-for-IDS-in-IoT-network**
2. **benchmark results.csv**
3. **Tooling and compute environment**

One option is to execute the entire workflow on a local workstation, which can provide stable long-running compute and direct control over storage and dependency management. However, local execution can reduce replicability across researchers because hardware capacity (CPU, RAM, GPU) and software configurations often differ materially, and these differences can influence both training time and convergence behavior - particularly for deep learning baselines.

A second option is to use managed cloud notebook environments, which provide a standardized execution interface, rapid iteration, and accessible sharing of complete experimental pipelines. This approach supports methodological transparency because the benchmark can be executed from a single notebook that documents preprocessing, model training, and evaluation in a consistent order, reducing inadvertent deviations in experimental steps. The main drawbacks are operational rather than methodological: session time limits, runtime resets, and variable availability of accelerators require the workflow to be designed with persistent storage and resilient data loading.

A third option is to adopt a containerized workflow (e.g., Docker-based execution with pinned dependencies and scripted runs). This is typically the strongest approach for strict reproducibility, yet it introduces higher engineering overhead, which is often disproportionate for an MSc dissertation whose contribution is empirical comparison rather than infrastructure engineering.

Given the dissertation's objectives - systematic comparison across multiple model families under a uniform preprocessing and evaluation protocol - the selected approach is a managed cloud notebook implementation in Google Colab, using an NVIDIA T4 GPU where applicable for deep learning training. This choice is justified because it provides adequate computational capability for neural baselines while maintaining accessibility and a consistent execution environment suitable for replication. To preserve experimental integrity, all persistent artefacts (datasets, trained models, and result outputs) are stored in Google Drive, ensuring that transient runtime resets do not compromise the benchmark pipeline or lead to undocumented reruns. Overall, this compute strategy balances practicality with methodological control, enabling a consistent, auditable benchmarking workflow without introducing unnecessary infrastructure complexity.