Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in ICT for Internet And Multimedia

Network attacks detection and classification via deep learning algorithms

Master Candidate

Saeed Rezaee

Student ID 2071538

Supervisor

Prof. Michele Rossi

University of Padova

Co-supervisor

Dr. Giovanni Perin

University of Brescia

University of Padova

Academic Year
2024/2025

"The people of this world are like the three butterflies in front of a candle's flame.

The first one went closer and said: I know about love.

The second one touched the flame lightly with his wings and said: I know how love's fire can burn.

The third one threw himself into the heart of the flame and was consumed. He alone knows what true love is."

— Attar of Nishapur

Abstract

This thesis investigates neural networkbased intrusion detection on large-scale, heterogeneous network-flow data. We curate a unified, leakage-aware corpus by harmonizing UNSW-NB15, CIC-IDS-2017, and CICDDoS-2019 (77M flows) under a common feature contract. Building on this foundation, we design a two-phase cascaded IDS. Phase 1 is a high-recall binary detector that separates benign from attack traffic; Phase 2 activates only on Phase 1 positives and assigns an attack family using a compact ontology that merges semantically overlapping labels (e.g., DNS/LDAP/SNMP, NET_BIOS/PORTMAP, SYN/UDP_LAG). Both phases use a regularized multilayer perceptron (MLP) tailored to tabular flow statistics, with group/stratified splits to prevent leakage, robust scaling fit on training data only, and imbalance-aware objectives. We calibrate scores (isotonic) and select thresholds at fixed false-positive budgets to report deployment-relevant metrics (ROC-AUC, PR-AUC, and TPR@FPR), including stress tests at 1% attack prevalence. Comparative baselines (1D-CNN and GRU) and a LightGBM reference demonstrate that MLPs provide superior precisionrecall, more stable calibration, and faster convergence on aggregated flow features. The resulting cascade offers high recall at low FPR with reproducible artifacts (models, scalers, calibration maps, thresholds) suitable for operational integration. Contributions include: (i) a unified, leakage-aware dataset and label ontology; (ii) a calibrated two-phase IDS with explicit operating points; (iii) an imbalance- and prevalence-aware evaluation protocol; and (iv) an HPC-ready, containerized pipeline for training at tens of millions of flows.

Sommario

Questa tesi studia limpiego di reti neurali per il rilevamento di intrusioni su dati di flusso di rete eterogenei e su larga scala. Viene costruito un corpus unificato e attento alle perdite dinformazione integrando UNSW-NB15, CIC-IDS-2017 e CICDDoS-2019 (77 milioni di flussi) con un contratto di feature comune. Su tale base progettiamo un IDS a due fasi in cascata: la Fase 1 è un rilevatore binario ad alta richiamata per distinguere traffico benigno da attacchi; la Fase 2 si attiva solo sui positivi della Fase 1 e classifica la famiglia di attacco utilizzando unontologia compatta che fonde etichette semanticamente sovrapposte (ad es. DNS/LDAP/SNMP, NET_BIOS/PORTMAP, SYN/UDP_LAG). Entrambe le fasi impiegano un MLP regolarizzato, adatto a statistiche tabellari di flusso, con partizionamenti per evitare leakage, scaling robusto calcolato solo sul training e strategie contro lo sbilanciamento. I punteggi sono calibrati (isotonic) e le soglie sono scelte a falsi positivi fissati, riportando metriche utili in esercizio (ROC-AUC, PR-AUC, TPR@FPR) e test di prevalenza all1%. Confronti con 1D-CNN e GRU, oltre a un riferimento LightGBM, mostrano che gli MLP offrono migliore precisionrecall, calibrazione più stabile e convergenza più rapida su feature aggregate. La cascata risultante garantisce alta richiamata a basso FPR con artefatti riproducibili (modelli, scaler, mappe di calibrazione, soglie) idonei allintegrazione operativa. I contributi includono: (i) dataset unificato e ontologia delle etichette; (ii) IDS a due fasi calibrato con punti operativi espliciti; (iii) protocollo di valutazione consapevole di squilibri e prevalenza; (iv) pipeline containerizzata e pronta per HPC per addestramenti su decine di milioni di flussi.

Contents

List of Figures

List of Tables

List of Code Snippets

Acknowledgments

I would like to express my deepest gratitude to the **University of Padova** for the opportunity to broaden my knowledge in ICT. Studying here has not only shaped my technical abilities, but also given me the opportunity to pursue a better life and contribute - however modestly - to making the world a better place.

My sincere thanks go to **Prof. Michele Rossi** and **Dr. Giovanni Perin** for proposing this exciting thesis topic and for their guidance throughout the project. Their mentorship expanded my understanding of *Machine Learning* and *Deep Learning* in a profound and practical way that will greatly benefit my future career.

I am grateful to all my professors in the **ICT for Internet and Multimedia** program for the knowledge, rigor, and curiosity they instilled in me.

I owe everything to my **family**, especially my parents, whose unwavering support made this journey possible. They gave me every opportunity they could, despite the difficulties in our home country, so that I could pursue the life and education that they were denied.

Finally, my heartfelt thanks to my **friends**, whose presence over the last three years - through ups and downs - was the fuel that kept me moving toward my goals. Their support made this path much less difficult and far less frustrating; without them, this journey would have been immeasurably harder.

Thank you all.

1 Introduction

The rapid expansion of digital infrastructures and cloud-native services has been accompanied by a sustained rise in both the scale and sophistication of cyber threats. Traditional intrusion detection systems (IDS) built on signatures and static rules struggle to keep pace with polymorphic malware, encrypted traffic, and evolving adversarial behaviors [23, 22]. In particular, volumetric and application-layer Distributed Denial of Service (DDoS) campaigns continue to surge globally and remain an acute threat to availability; industry telemetry reported millions of attacks within a single quarter, underscoring the need for adaptive, data-driven defenses. *Eye-Net*, for instance, highlights this pressure and advocates deep learning with model compression for efficient DDoS detection on constrained devices [7].

Against this backdrop, machine and deep learning have emerged as promising approaches for network defense. Neural models learn discriminative patterns directly from network flow statistics and can generalize to attack variants that elude signature-based tools [23, 24]. Yet their practical value hinges on three factors: (i) *representative data* that capture benign diversity and multiple attack families without leakage, (ii) *architectures* that match the inductive biases of tabular flow features, and (iii) *methodology* that addresses class imbalance, calibration, and deployment-time operating points. Prior work such as Eye-Net demonstrates that careful feature selection, balancing, and MLP-based classifiers can deliver strong DDoS detection, even under quantization for edge scenarios; at the same time, many studies focus on narrow attack sets or single datasets, which limits generalization across environments [22, 28].

1.1 Problem Statement and Motivation

Despite steady progress, three gaps persist. First, widely used benchmarks often cover a limited subset of threats or differ in collection protocols, making cross-domain generalization brittle. Second, binary classification—while useful for triage—offers limited forensic value without attributing *which* family is active. Third, scaling learning pipelines to tens of millions of flows raises practical issues: leakage control across sources, skewed class priors, and robust probability calibration at low false positive rates (FPR).

This thesis tackles these challenges with a consolidated, leakage-aware corpus constructed by harmonizing *UNSW-NB15*, *CIC-IDS-2017*, and *CICDDoS-2019* [14, 19, 20]. Unlike DDoS-only studies (e.g., Eye-Net), our goal is broader: train a system that (i) detects attacks in mixed enterprise-like traffic and (ii) classifies the attack family, including reflection/exploitation DDoS variants and non-DDoS classes, within a single methodology [7].

1.2 Approach Overview

We adopt a *two-phase cascaded* design at the flow level. Phase 1 is a high-recall binary detector that separates benign from attack traffic under explicit operating constraints (e.g., TPR@FPR targets). Phase 2 activates only on Phase 1 positives and performs multi-class attribution over a compact label ontology that merges semantically overlapping families (e.g., `NET_BIOS/PORTMAP`, `DNS/LDAP/SNMP`). Both phases share a harmonized feature contract derived from CICFlowMeter-style statistics. Following extensive comparisons, we use a *multilayer perceptron* (MLP) as the primary backbone: for tabular flow statistics without spatial or temporal topology, MLPs consistently outperform 1D-CNN/GRU variants in validation PR-AUC and calibration stability, while remaining efficient to train at scale [24].

Methodologically, the pipeline enforces leakage controls (grouped/stratified splits), robust scaling fit only on training data, class-imbalance remedies, and post-hoc probability calibration. Operating thresholds are selected on validation to meet target FPRs and then transferred to test [15, 28]. This produces deployment-ready metrics (e.g., TPR@FPR $\in \{10^{-3}, 10^{-2}, 5 \times 10^{-2}\}$) and enables prevalence-stress tests (e.g., 1% attack rate) that reflect real network conditions.

1.3 Contributions

- **Unified, leakage-aware corpus:** a harmonized dataset combining UNSW-NB15, CIC-IDS-2017, and CICDDoS-2019 with a consistent feature contract for flow-level learning and a principled label ontology for attack families.
- **Two-phase IDS with calibrated outputs:** a cascade of MLP detectors delivering high recall at low FPR (Phase 1) and reliable multi-class attribution (Phase 2), with post-hoc calibration to support thresholding and SOC automation.
- **Imbalance-aware training and realistic evaluation:** class-balanced objectives, stratified batching, and evaluation at fixed FPRs and shifted prevalences, including conditional assessment on Phase 1-passed traffic to quantify cascade effects.
- **Transparent, reproducible pipeline:** HPC-friendly implementation with artifact versioning (models, scalers, thresholds) and memory-efficient parquet loaders for tens of millions of flows (contextualized by the dataset scale in).

1.4 Positioning w.r.t. Prior Work

Eye-Net exemplifies a focused DDoS detector that couples MLPs with feature selection, data balancing, and quantization-aware training to achieve energy-efficient inference on IoT devices. We share the emphasis on MLP efficacy for flow statistics and on handling imbalance; however, our thesis extends beyond DDoS-only scopes by *(i)* unifying multiple corpora with leakage-aware splits, *(ii)* addressing both detection and *family-level* classification in a cascade, and *(iii)* centering evaluation on low-FPR operating points and calibrated probabilities for enterprise deployment. These choices aim to bridge the gap between high offline accuracy and operational utility [7].

2

Background and Related Work

2.1 Eye-Net — Low-Complexity DDoS Detection with MLP

Eye-Net proposes a pragmatic deep learning pipeline for network-based intrusion detection that explicitly targets flow-level distributed denial-of-service (DDoS) detection under deployment constraints. The work is situated in the line of research that uses CICFlowMeter-style tabular features rather than raw packets or payloads, and it focuses on building a compact multilayer perceptron (MLP) that remains accurate after model compression. The authors articulate three intertwined challenges that often appear in flow-based IDS research: (i) class imbalance between benign traffic and heterogeneous attack families, which can bias training and suppress minority-class recall; (ii) unnecessary variance from weak or redundant features that inflates model size and fosters overfitting; and (iii) limited attention to inference efficiency, which hampers practical adoption on commodity or edge hardware. To address these, Eye-Net integrates classical feature selection, targeted data balancing, and quantization-aware training (QAT) into a single, end-to-end recipe.

The dataset foundation is CICDDoS2019, a modern, labeled corpus of reflection and application-layer DDoS attacks represented as bidirectional flows with dozens of statistical features (inter-arrival times, packet/byte counts, flag counts, window sizes, and per-flow rates). Consistent with best practices, the pipeline first removes identifier-like metadata and degenerate columns (e.g., fields that are constant or near-empty) and applies simple minmax normalization so that gradient-based training is well-conditioned across heterogeneous scales. Feature selection is performed with a univariate ANOVA F-test over the training split, ranking attributes by their between-class discriminative power. Rather than chasing very high dimensionality, Eye-Net argues for a compact subset (on the order of a few dozen features) that stabilizes training and yields predictable latency; in practice, keeping roughly thirty well-ranked features offers a strong balance between separability and complexity. Class imbalance is mitigated during training with synthetic oversampling of minority classes (SMOTE), which expands decision boundaries for underrepresented attacks without altering the validation/test distributions. The classifier itself is a small MLP with one or two hidden layers and rectified linear activations, trained with a modern optimizer and early stopping; the same backbone is used in two regimes, a binary model (benign vs. DDoS) and a multiclass model over several DDoS families.

A distinctive contribution of Eye-Net is the emphasis on *quantization-aware training*. Instead of training in full precision and quantizing post hoc, the method inserts quantization effects during training so that weight tensors (and, in some configurations, activations) learn to be robust to 8-bit integer representations. The paper presents a simple layer-wise procedure: progressively quantize portions of the network, briefly freezing those parts while allowing the remainder to adapt, and then continue fine-tuning end-to-end. The result is a model that preserves most of its predictive performance while attaining substantial reductions in memory footprint and arithmetic cost, which directly translates to faster inference and reduced energy on CPUs and low-power devices. Importantly, the authors do not rely on exotic architectures or specialized operators; the gains come from careful preprocessing, disciplined feature selection, and training-time awareness of the deployment precision.

Findings and Relevance to This Thesis

Empirically, Eye-Net shows that a small, well-regularized MLP trained on a curated feature subset can achieve state-of-the-art accuracy for binary DDoS detection on CICDDoS2019, and that a similar backbone remains competitive in the more demanding multiclass setting when two practical interventions are applied: class rebalancing during training and, where justified by feature-space overlap, conservative merging of nearly indistinguishable attack families.

The latter is motivated by confusion analyses (e.g., systematic misclassification between closely related services such as Portmap and NetBIOS), and the paper treats it as a data-centric modeling decision rather than a purely algorithmic fix. After quantization-aware training to 8-bit precision, the models retain high precision and recall with only modest degradation relative to full-precision baselines, while achieving substantial reductions in model size and compute.

For a thesis that aims to train a two-stage cascade on a merged flow corpus (UNSW-NB15, CIC-IDS-2017, CICDDoS2019), Eye-Net is valuable in several concrete ways. First, it validates the *binary-then-multiclass* design with the same family of tabular features and a compact MLP backbone: a high-recall Stage 1 anomaly filter followed by a Stage 2 attack-type classifier is both effective and operationally simple. Second, it offers a disciplined, reproducible preprocessing pathdrop identifiers and degenerate fields, normalize, rank features with ANOVA, and cap dimensionality around a small, stable setwhich is directly transferable to large, merged tables and helps contain training time. Third, it demonstrates that minority-class performance in multiclass attack recognition is primarily a *data* problem before it is a *model* problem: targeted oversampling and taxonomy hygiene (merging or regrouping highly similar attacks) reduce systematic confusions that persist even with stronger networks. Finally, it establishes a deployment-oriented baseline: quantization-aware training can be applied to MLPs with minimal engineering overhead to produce compact INT8 models that are easier to serve on CPU-only nodes, enabling low-latency inference without resorting to heavier hybrids.

In summary, Eye-Net is a strong exemplar of data-centric, efficiency-aware deep learning for flow-based IDS. Its results support three design choices in this thesis: (1) adopt a small MLP baseline with explicit feature selection as the primary workhorse for both stages; (2) treat class balance and label taxonomy as first-class levers for multiclass accuracy, guided by confusion analysis; and (3) include deployment metrics (model size, throughput, latency) alongside F1/PR-AUC, leveraging quantization-aware training to meet real-world serving constraints. These lessons complement the thesiss broader goal of harmonizing multiple flow datasets and building a cascade that first detects intrusions and then assigns plausible attack families with robust per-class performance.

2.2 OptMLP-CNN for DDoS Detection in SDN

This work introduces an optimized hybrid deep-learning detector, OptMLP-CNN, tailored for Distributed Denial of Service (DDoS) detection in Software-Defined Networking (SDN) environments. The motivation is that SDN architectures, with centralized control and dynamic reconfiguration, have distinctive traffic dynamics and failure modes that both complicate detection and raise the cost of false alarms. The proposed approach embraces three complementary ideas: (i) a data-centric feature pipeline that selects a compact, highly-informative subset of flow attributes; (ii) a hybrid model that fuses a multilayer perceptron (MLP) with a convolutional neural network (CNN) to exploit complementary inductive biases; and (iii) rigorous model optimization through SHAP-based feature selection and Bayesian hyperparameter tuning, with Adam for training. The authors evaluate on two public datasets to anchor both general Internet traffic and SDN-specific regimes: CICDDoS-2019 and InSDN.

The feature pipeline begins with standard preprocessing (normalization, formatting, and optional up-sampling if class distribution is skewed) followed by SHAP-based selection. Rather than relying on correlation heuristics alone, SHAP scores quantify each features marginal contribution to the models predictions, yielding a ranked subset that is then sanity-checked via correlation matrices to identify redundancy and interactions. The goal is to retain only features that materially drive separability while curbing overfitting and reducing compute. In practice, the selected subset is small (on the order of a few tens of features), and the paper visualizes SHAP summary plots and correlation matrices for both CICDDoS-2019 and InSDN to justify the retained attributes.

The architecture merges an MLP branch (effective on tabular, per-flow statistics) with a CNN branch (useful when projecting features into structured representations or when exploiting local receptive fields). The combined model is trained end-to-end with Adam; a Bayesian optimizer searches key hyperparameters (e.g., widths, depths, learning rates, regularization) to balance bias and variance and improve generalization. The training and evaluation routine is conventional but thorough: hold-out testing, standard classification metrics (accuracy, precision, recall, F1, AUC), and ablations that vary activations and input-weight initializations. Conceptually, this stacked design argues that even with tabular flow features, local structure and cross-feature patterns can benefit from shallow convolutional processing when properly optimized.

Findings and Relevance to This Thesis

On CICDDoS-2019, OptMLP-CNN reaches approximately 99.95% accuracy with strong precision/recall/F1 (around 0.999), and on the SDN-specific InSDN dataset it attains roughly 99.98% accuracy with similarly elevated precision/recall/F1 and near-perfect AUC. A detailed metric table reports Accuracy $\approx$ 0.9995, Precision $\approx$ 0.9990, Recall $\approx$ 0.9998, F1 $\approx$ 0.9994, AUC $\approx$ 0.9979 on CICDDoS-2019, and Accuracy $\approx$ 0.9998, Precision $\approx$ 0.9999, Recall $\approx$ 0.9997, F1 $\approx$ 0.9998, AUC $\approx$ 0.9996 on InSDN. The paper also compares against a range of baselines (e.g., DNN, ELM, Gradient Boosting and other optimized ML variants) across several datasets (NSL-KDD, UNSW-NB15, CIC-IDS-2017, CICDDoS-2019, InSDN), arguing that the optimized hybrid and SHAP-driven feature selection tend to match or outperform prior scores while keeping the feature set compact and the model stable.

For a thesis that adopts a two-stage cascade (binary intrusion detection followed by attack-type classification) on a merged flow corpus (UNSW-NB15, CIC-IDS-2017, CICDDoS-2019), this work is useful in several ways. First, it is a strong reference for *feature selection under deployment constraints*: SHAP offers an interpretable, data-driven method to prune to a small, high-yield subset, which is critical when harmonizing heterogeneous flow schemas from multiple sources. Second, it demonstrates that *model capacity can be focused rather than inflated*: by coupling a compact MLP with a small CNN branch and then using Bayesian tuning, the detector attains near-ceiling accuracy without excessive depth/width, which complements an MLP-first design ethos and translates to faster training on large merged tables. Third, the SDN emphasis and the InSDN results provide evidence that models tuned on generic Internet DDoS features can transfer to controller-centric environments if the pipeline is data-centric (balanced classes, curated features) and hyperparameters are tuned for the new traffic dynamics. Fourth, the SHAP plots and correlation checks model a reproducible *taxonomy-hygiene and interpretability* workflow that we can adopt: identify features that drive confusions, drop redundant ones, and revisit class definitions if persistent overlaps emerge.

In sum, OptMLP-CNN is a strong hybrid baseline and a methodological guide: use SHAP to create a tight feature core; apply Bayesian optimization to stabilize generalization; keep the network compact but expressive; and report comprehensive metrics (precision/recall/F1/AUC) on both conventional and

SDN-specific datasets. In our cascade, these lessons support (i) a lean Stage 1 that favors recall and robustness on a selected feature core, and (ii) a Stage 2 that can optionally benefit from a small CNN branch oreven when remaining MLP-onlycan leverage SHAP-driven pruning and Bayesian tuning to boost minority-class performance on multiclass attack families.

2.3 A New Deep Learning-Based Intrusion Detection System for Cloud Security

This paper proposes a lightweight deep learning model for intrusion detection in cloud environments, with an explicit focus on both binary (attack vs. normal) and five-class classification on the NSL-KDD benchmark. The authors motivate the work by arguing that deep models can jointly learn feature extraction and classification, reducing reliance on manual engineering and coping better with evolving threats in cloud data centers. They position convolutional neural networks (CNNs) and recurrent neural networks (RNNs) as complementary building blocks: CNN layers to extract local patterns from input features and GRU/LSTM-style recurrent layers to model temporal or sequential structure where present. The method targets operational practicalitykeeping the architecture compact while pursuing state-of-the-art accuracy on NSL-KDD. The paper reports that their proposed CNN+RNN (CNN+GRU) model achieves *99.86% accuracy* on the five-class NSL-KDD task, which the authors claim is competitive with or better than prior work they survey. Their results are presented alongside a literature review of deep learning IDS approaches, including DBN+PNN hybrids, stacked sparse autoencoders (SSAE), GRU-based networks, and self-taught learning (STL-IDS) that combine deep feature learning with classical classifiers. The review contextualizes the proposed approach within a decade of IDS work shifting from KDD99 to NSL-KDD and then to more recent datasets, and it emphasizes that careful data organization and preprocessing remain decisive for deep models.

The dataset section recaps NSL-KDDs label taxonomy and the challenge posed by minority categories (U2R, R2L). The paper lists the class distributions used in training and testingNormal (67,343/9,711), DoS (45,927/7,458), U2R (52/200), R2L (995/2,421), Probe (11,656/2,754)highlighting the severe sparsity of U2R and R2L during training. The experimental setup follows standard practice: model training on the provided train split and evaluation on the held-out test split; metrics include accuracy and comparative references to earlier results. While the article underscores the importance of data preprocessing (normalization/organization) for achieving strong outcomes, its main technical novelty is the compact CNN+RNN backbone rather than an elaborate feature-engineering stage. The papers narrative consistently argues that deep feature extractors (convolutions and gated recurrent units) are key to surpassing classical ML baselines on NSL-KDD and to maintaining efficiency suitable for cloud deployment.

Relevance and Takeaways for This Thesis

For a thesis centered on flow-based IDS with a two-stage cascade (binary attack detection followed by attack-type classification) using modern datasets (UNSW-NB15, CIC-IDS-2017, CICDDoS-2019), this paper is most valuable as a methodological and historical reference rather than a direct benchmark. First, it demonstrates that a *compact* deep architecture (CNN+GRU) can achieve *very high five-class accuracy* on NSL-KDD (99.86%), reinforcing the general claim that deep learned feature extractors help in intrusion detection. This supports the thesiss choice to consider deep backbones (even if the primary baseline is MLP) and to compare against sequence-aware or hybrid models in Phase 2 when minority-class separation is challenging. Second, the explicit presentation of class distributions and the noted difficulty of rare classes (U2R/R2L) align with the thesiss emphasis on imbalance-aware training for attack-type classification; even though the paper does not dwell on advanced rebalancing techniques in its own experiments, its literature review references SMOTE/undersampling and related strategies as common practice, and the results remind us that careful handling of minority classes is essential when moving from binary to multiclass IDS. Finally, while NSL-KDD is older and less representative than UNSW/CIC, this work offers a clean example of reporting practices (clear splits, compact model, headline accuracy) and of designing an IDS architecture that is simultaneously accurate and lightweightan ethos that transfers to our cascade where model size, latency, and training efficiency matter on a merged, large-scale flow corpus.

2.4 AI-IDS Deep Learning for Real-Time Web Intrusion Detection

AI-IDS presents a production-oriented network intrusion detection system focused on web traffic, bridging academic deep learning research with practical deployment constraints. The core idea is to learn from *HTTP application-layer data* rather than relying exclusively on hand-crafted signatures or purely flow-level statistics. Requests are preprocessed via lightweight UTF-8 normalization and tokenization that preserve discriminative byte- and character-level patterns while avoiding heavy, brittle feature engineering. On top of this representation, the authors evaluate several neural backbonesmost notably a *CNN→LSTM* stack (convolutions extracting local n-gramlike motifs followed by recurrent layers to capture longer-span dependencies), a reversed *LSTM→CNN* ordering, and a plain deep feedforward network as a simpler reference. The system targets *real-time* inference and thus emphasizes tight data paths, bounded preprocessing, and model sizes suitable for deployment on commodity servers with containerized services.

The training and evaluation regimen combines public benchmarks and large-scale operational traces. Beyond a controlled public dataset setting (e.g., subsets of CIC-IDS2017 and a classical web-request corpus such as CSIC-2010), the system ingests production HTTP traffic, leveraging internal labeling workflows and iterative refinement to track concept drift. Importantly, the work closes the loop between detection and prevention: high-confidence detections are distilled into updated *Snort* (or similar) signatures, turning learned neural patterns into actionable, human-readable rules. This synergy reduces reliance on static rule maintenance while containing false positives through a feedback process that revalidates and retrains on newly labeled examples. In addition to standard metrics (precision, recall, F1, ROC/PR behavior), the paper reports operational indicators like alert volume and throughput under real workloads, arguing that practical IDS evaluation must reflect both model quality and engineering viability.

From an architectural perspective, the paper motivates the CNN→LSTM ordering as a good trade-off for textual payloads: convolutions act as efficient local feature extractors (robust to minor obfuscations and encoding tricks), while recurrent layers integrate context across tokens and fields (URLs, headers, bodies). Regularization (dropout/early stopping), careful batching, and balanced mini-batches are employed to curb overfitting often seen on small public corpora. The authors further discuss the effect of vocabulary normalization and byte-level modeling on generalization to obfuscated or previously unseen attack strings. In their ablations, sequence-aware models typically outperform purely feedforward baselines on application-layer attacks (e.g., SQLi/XSS), especially when requests are short, noisy, and adversarially perturbed.

Relevance to This Thesis

Although this thesis primarily trains on *flow-level* features (UNSW-NB15, CIC-IDS2017, CICDDoS-2019) and adopts a two-stage cascade (Phase 1 binary detection followed by Phase 2 attack-type classification), AI-IDS is an instructive exemplar for several reasons. First, it shows how a deep model can be embedded within a *real-time, containerized pipeline* that supports continuous labeling and retrainingan operational pattern directly applicable to maintaining the cascade over time as traffic and attack distributions shift. Second, its *sequence-aware* design (CNN→LSTM) provides a transferable idea for Phase 2 when distinguishing closely related attack families: even with tabular flows, short temporal windows or per-connection sequences can be modeled with similar hybrids if a pure MLP underperforms on minority classes. Third, the paper demonstrates a *detection-to-rule* workflow by converting stable neural detections into updated signatures; analogously, high-precision patterns identified by the cascade could seed traffic filters, firewall rules, or anomaly baselines to reduce load and improve interpretability. Finally, the emphasis on reporting both classical ML metrics and *operational signals* (latency, throughput, alert rates) aligns with this thesiss goal to evaluate not just accuracy (e.g., F1/PR-AUC, per-class recall) but also deployment feasibility (model size, inference budget). In short, AI-IDS complements the thesis by exemplifying how deep IDS models move from benchmark success to sustainable production practice, and by offering design patternslightweight preprocessing, sequence-aware backbones, and feedback-driven rule updatesthat can strengthen the proposed binary→multiclass cascade on merged CIC/UNSW flow data.

2.5 Gaps in Current Approaches

Surveying the four closest works to our setting—Eye-Net (low-complexity MLP with ANOVA/SMOTE and INT8 quantization on CICDDoS2019), OptMLP–CNN (SHAP-selected features with a hybrid MLP+CNN and Bayesian tuning on CICDDoS2019 and InSDN), a compact CNN+GRU IDS for cloud security (evaluated on NSL–KDD), and AI–IDS (CNN→LSTM for real-time HTTP intrusion detection, validated on CICIDS2017 and production traffic)—reveals several systematic gaps that the present thesis directly addresses with a two-stage cascade trained on a *merged, large-scale flow corpus* (UNSW–NB15, CIC–IDS–2017, CICDDoS2019; ~77M flows) and an MLP-first design.

First, prior work is largely *dataset-isolated*. Eye-Net and OptMLP–CNN focus on CICDDoS2019 (and InSDN for the latter), while the cloud IDS targets NSL–KDD and AI–IDS emphasizes application-layer HTTP. None of these studies harmonize heterogeneous modern flow datasets into a single training table with a unified schema and taxonomy. In contrast, our thesis contributes a reproducible *schema alignment and label unification* across UNSW–NB15 and the CIC family, enabling training and evaluation at realistic scale and exposing cross-dataset generalization challenges that single-corpus studies may hide.

Second, the literature tends to optimize for either *binary DDoS detection* or *single-dataset multiclass* (often DDoS-family only). Even when both regimes are explored (e.g., Eye-Net), the scope remains narrow (six DDoS types) and excludes broader attack families common in UNSW/CIC (PortScan, Fuzzers, Infiltration, Botnet, shellcode, etc.). Our cascade explicitly separates *Phase 1 anomaly filtering* from *Phase 2 attack-type recognition* over a richer, unified label map and studies the stability of that map (including principled class merges) under severe class imbalance.

Third, while some works adopt strong *data-centric practices* (ANOVA in Eye-Net, SHAP in OptMLP–CNN, class balancing in multiple papers), they rarely examine how these choices transfer across datasets with different feature definitions and capture conditions. Our setting requires feature-space intersection, identifier removal, and normalization that jointly preserve discriminative power across corpora; we further quantify the effect of rebalancing and label regrouping on minority-class recall in a large, merged table rather than in siloed datasets.

Fourth, *evaluation protocols* are typically random-split within a single dataset. Cross-day or cross-dataset validation is uncommon, raising questions about robustness to capture drift and unseen environments. This thesis includes *cross-corpus* and *capture-aware* splits to measure generalization beyond in-distribution IID assumptions and to surface failure modes (e.g., day-of-week effects, service-specific biases) that are invisible to single-corpus splits.

Fifth, only Eye-Net treats *deployment efficiency* as a first-class objective through quantization-aware training to INT8, and even there, input quantization and end-to-end latency/throughput on commodity CPUs are not comprehensively reported. OptMLP–CNN achieves very high accuracy but does not quantify memory/latency trade-offs for the hybrid branch or compare against compact baselines under a fixed inference budget. Our thesis standardizes reporting of *accuracy and operational metrics* (model size, CPU latency, throughput) and evaluates whether MLP baselines with quantization suffice before introducing heavier hybrids.

Sixth, *taxonomy hygiene* is ad hoc. Eye-Net merges Portmap and NetBIOS based on confusion analysis, but there is little guidance for systematic regrouping across datasets with overlapping or inconsistently defined labels (e.g., UNSW vs. CIC naming). We formalize a *merge policy* driven by empirical separability (PR curves, confusion concentration) and operational usefulness, documenting the impact on Phase 2 stability and on interpretability.

Seventh, *open-world and calibration* issues are underexplored. The surveyed works emphasize closed-set accuracy and macro F1, with limited attention to calibrated probabilities, threshold selection for high-recall Stage 1 filtering, or rejection options for unknown classes. Our cascade includes *calibration and thresholding* for Phase 1 to bias toward recall with bounded false-positive cost, and we report precision-recall trade-offs and calibration error alongside standard metrics.

Eighth, *data leakage risks* are rarely audited at scale (e.g., flows from the same connection or host appearing in both train and test). Merged multi-day, multi-source corpora exacerbate this. We adopt capture-aware splits and flow-grouping to mitigate leakage (per-connection/host/day), and we make the splitting policy explicit to ensure reproducibility and fair comparison.

Ninth, despite mentions of SMOTE or class weights, there is limited analysis of *minority-class robustness* under extreme imbalance (e.g., Infiltration/Heartbleed in CIC–IDS–2017, U2R/R2L analogs in UNSW). Our study systematically compares balancing strategies (percentile-capped oversampling, mixup-style augmentation, class-weighted/focal losses) and quantifies their effect on per-class PR_{AUC} and confusion persistence, guiding when data regrouping is preferable to model complexity.

Tenth, *continual learning and drift handling* are often discussed qualitatively (notably in AI–IDS) but seldom evaluated in flow-based settings. We outline a lightweight retraining loop and provide empirical evidence for performance retention under temporal drift, focusing on the practicality of keeping an INT8 MLP cascade fresh with periodic mini-batch updates.

Finally, reproducibility artifacts (code, exact feature lists, preprocessing recipes) are incomplete or dataset-specific in prior work, complicating re-use on merged corpora. Our implementation details a *portable preprocessing and training stack* that starts from parquet flow tables, performs identifier dropping and schema alignment, and trains both cascade stages with explicit seeds, logging, and evaluation scripts suitable for large-scale experimentation.

In summary, existing approaches excel on *single datasets*, narrow *attack scopes* (often DDoS only), or *application-layer payloads*, and they typically report accuracy without standardized constraints on *latency* and *model size*. This thesis closes these gaps by: (i) harmonizing multiple modern flow corpora into one training and evaluation substrate; (ii) adopting a cascade architecture tuned for high-recall anomaly filtering followed by robust attack-type classification with principled label regrouping; (iii) foregrounding operational metrics via quantized MLPs; and (iv) evaluating generalization under capture-aware, cross-corpus splits with rigorous imbalance handling, calibration, and reproducible pipelines.

3

Datasets and Preprocessing

3.1 Dataset Strategy: Selecting or Merging Multiple Datasets

The intrusion detection setting targeted in this thesis requires a training corpus that (i) covers a broad spectrum of benign and malicious traffic patterns, (ii) reflects realistic network behaviors across multiple environments, and (iii) exposes the model to diverse attack families and tooling. To this end, the strategy is to curate a composite dataset by selecting and harmonizing multiple widely used network-flow corpora rather than relying on a single source [14, 19, 20]. The end product is a unified collection of approximately 7.7×10^7 flow records with a common schema suitable for two cascaded learning tasks: (Phase 1) binary detection of benign vs. attack and (Phase 2) multi-class categorization of attack type.

3.2 Selection Rationale and Inclusion Criteria

The candidate corpora are chosen to jointly satisfy the following:

- **Feature compatibility:** The datasets expose flow-level features derived from tools compatible with CICFlowMeter-style statistics (e.g., packet counts, inter-arrival times, segment sizes, flag counts). This eases schema alignment and reduces feature-engineering overhead [10].
- **Label richness:** Attack labels are sufficiently granular to support Phase 2

(e.g., application-layer DDoS variants, brute-force families, reflection/amplification attacks), while still allowing principled merging of near-duplicate classes [19, 20].

- **Coverage and diversity:** Each dataset contributes distinct traffic generators, services, and background benign workloads, improving generalization and robustness to domain shift [14, 19].
- **Documented collection protocol:** Clear descriptions of capture setups and labeling procedures help adjudicate label noise and potential biases [14, 19, 20].

Scope: UNSW-NB15, CICIDS2017, CICIDS2019

Under these criteria, three corpora are included: the general-purpose academic IDS benchmark **UNSW-NB15** (broad benign/attack coverage), the enterprise-like multi-day capture **CIC-IDS-2017** with diverse application protocols, and the DDoS-centric corpus **CICDDoS-2019** offering fine-grained attack variants [14, 19, 20]. This combination balances breadth (for detection) and depth (for classification).

Attack Families Covered (brief explanations)

The three corpora used in this thesis collectively include volumetric floods and reflection/amplification Distributed Denial of Service (DDoS) variants alongside TCP-based exhaustion attacks. Below we summarize each family referenced in our label ontology.

DDoS (umbrella). A collective term for coordinated attacks aiming to exhaust bandwidth, state (e.g., connection tables), or server/application resources by distributing traffic across many bots. *Typical signal:* high flow counts and rates, many short-lived flows, skewed down/up ratios. [13, 1]

DNS (reflection/amplification). An attacker sends spoofed DNS queries to open resolvers so that large responses (e.g., ANY, DNSSEC) are reflected to the victim (UDP/53). *Typical signal:* small upstream request vs. large downstream response bytes; bursts of short UDP flows. [16]

LDAP (CLDAP reflection). Abuses the Connection-less LDAP service (UDP/389) to elicit amplified replies from misconfigured servers to a spoofed victim. *Typical signal:* UDP/389, asymmetric bytes/packet sizes, high packets-per-second.

SNMP (reflection). Exploits SNMP (often v2, UDP/161) on devices with public/community strings to reflect responses toward a spoofed target. *Typical signal:* UDP/161 spikes, response-dominant byte totals, steady flow of small requests larger replies. [16]

UDP (generic flood). Sends high-rate UDP datagrams to random or specific ports to saturate access links or overwhelm CPU via ICMP unreachable processing. *Typical signal:* extreme packets-per-second, minimal handshake features, near-zero TCP flags.

NET_BIOS (NBNS reflection). Leverages NetBIOS Name Service (UDP/137) on Windows hosts to reflect/amplify traffic using spoofed queries. *Typical signal:* UDP/137 concentration, asymmetric request/response lengths, brief bursts. [16]

PORTMAP (rpcbind reflection). Abuses Portmap/rpcbind (UDP/111) to generate amplified replies that are reflected to the victim. *Typical signal:* UDP/111 flows with larger downstream payloads and short durations. [16]

SYN (TCP SYN flood). Saturates a servers TCP backlog by sending large volumes of SYNs (often spoofed), forcing state allocation without completing the handshake. *Typical signal:* elevated `syn_flag_count`, low `ack/fin/rst`, many short flows. [4]

UDP_LAG (low-bandwidth latency attack). Aims to induce latency (lag) with carefully timed or crafted UDP packets (e.g., to game or real-time services) rather than pure volumetric saturation. *Typical signal:* moderate pps with persistent jitter, abnormal inter-arrival times, service-specific ports.

MSSQL (reflection). Misuses the Microsoft SQL Server resolution service (UDP/1434) to reflect amplified replies to spoofed sources. *Typical signal:* UDP/1434, many small requests vs. larger responses, short-lived flows.

TFTP (reflection). Tricks Trivial FTP servers (UDP/69) into sending data blocks to a spoofed victim, providing modest amplification. *Typical signal:* UDP/69 spikes, small request multi-block response asymmetry.

NTP (reflection/amplification). Targets Network Time Protocol servers (UDP/123) with spoofed requests (historically `monlist`, mode 7, or other commands) to reflect large replies. *Typical signal:* UDP/123 dominance, large downstream bytes, high packet rates with short durations. [8]

3.3 Unification Pipeline

Merging heterogeneous corpora requires a disciplined pipeline to avoid information leakage and spurious correlations. The adopted pipeline is summarized below and applied identically to all sources before concatenation.

(1) Schema discovery and feature contract.

- Define a *feature contract* $\mathcal{F}$ containing the intersection/union of reliably computable flow features across sources (e.g., `fwd_iat_mean`, `bwd_iat_std`, `pkt_len_max/min`, `syn/psh/rst` flag counts, header lengths, subflow counters, bytes/packets per second) [10].

- Drop non-predictive identifiers that can leak session identity (e.g., `flow_id`, `src_ip`, `dst_ip`, ports) from the training matrix; retain only if used for stratification or exploratory analysis.

Table 3.1: Complete list of dataset columns (sorted alphabetically).

Feature	**Feature**	**Feature**
ack_flag_count	ece_flag_count	idle_max
active_max	fin_flag_count	idle_mean
active_mean	flow_bytes_per_s	idle_min
active_min	flow_duration	idle_std
active_std	flow_iat_max	label
avg_packet_size	flow_iat_mean	pkt_len_max
binary_label	flow_iat_min	pkt_len_mean
bwd_bulk_rate_avg	flow_iat_std	pkt_len_min
bwd_bytes_per_bulk_avg	flow_id	pkt_len_std
bwd_header_length	flow_packets_per_s	pkt_len_variance
bwd_iat_max	fwd_act_data_pkts	protocol
bwd_iat_mean	fwd_bulk_rate_avg	psh_flag_count
bwd_iat_min	fwd_bytes_per_bulk_avg	rst_flag_count
bwd_iat_std	fwd_header_length	src_ip
bwd_iat_total	fwd_iat_max	src_port
bwd_init_win_bytes	fwd_iat_mean	subflow_bwd_bytes
bwd_packets_per_bulk_avg	fwd_iat_min	subflow_bwd_packets
bwd_packets_per_s	fwd_iat_std	subflow_fwd_bytes
bwd_pkt_len_max	fwd_iat_total	subflow_fwd_packets
bwd_pkt_len_mean	fwd_init_win_bytes	syn_flag_count
bwd_pkt_len_min	fwd_packets_per_bulk_avg	timestamp
bwd_pkt_len_std	fwd_packets_per_s	total_bwd_packets
bwd_psh_flags	fwd_pkt_len_max	total_fwd_packets
bwd_urg_flags	fwd_pkt_len_mean	total_len_bwd_packets
cwr_flag_count	fwd_pkt_len_min	total_len_fwd_packets
dataset_id	fwd_pkt_len_std	urg_flag_count
down_up_ratio	fwd_psh_flags	
dst_ip	fwd_seg_size_min	
dst_port	fwd_segment_size_avg	

(2) Type normalization and sanitization.

- Cast numerical features to consistent dtypes; coerce non-finite values to `NaN`.
- Apply robust imputation per feature: median for scale-like features; zero for counters when semantically valid; otherwise remove rows with irreparable corruption.
- Winsorize extreme outliers at fixed quantiles (e.g., $[q_{0.01}, q_{0.99}]$) to stabilize optimization without discarding rare but legitimate flows.

```python
# Coerce numerics, replace infs, drop NaNs
data[feature_cols] = (data[feature_cols]
    .apply(pd.to_numeric, errors="coerce")
    .replace([np.inf, -np.inf], np.nan)
    .astype("float32"))
data.dropna(subset=feature_cols, inplace=True)

# De-duplicate exact & near-duplicates (rounded)
data = data.drop_duplicates(subset=feature_cols + ["binary_label"])
data_round = data.copy()
data_round[feature_cols] = data_round[feature_cols].round(6)
data = data_round.drop_duplicates(subset=feature_cols + ["binary_label"]).copy()
```

Code 3.1: Dataset: Type normalization and sanitization

(3) Temporal and session deduplication.

- Remove exact and near-duplicate flows within and across sources using a fingerprint over a subset of stable features; when both directions of a flow are present, keep each as independent samples only if the feature contract treats directions asymmetrically; otherwise collapse to a canonical orientation [10].

(4) Label ontology and mapping. We adopt a two-level label space for the cascade: **Level 0 (binary)** distinguishes $\{\texttt{benign} = 0, \texttt{attack} = 1\}$, and **Level 1 (attack family)** maps raw labels into a compact, operationally meaningful set used by Phase 2. Semantically overlapping families are consolidated where flow-level statistics are not reliably separable. All consolidation criteria, diagnostics, and the final mapping are detailed in Section 3.4.

(5) Scale normalization. Per-feature robust scaling is applied after the train/-validation/test split using statistics computed *only* on the training set:

$$\tilde{x}_f = \frac{x_f - \mathrm{median}(x_f)}{\mathrm{IQR}(x_f)},$$

which preserves heavy-tail structure better than z-scoring and mitigates the influence of extreme values common in attack traffic.

3.4 Consolidated Attack-Family Ontology (Level 1)

Rationale. Phase 2 operates at the family level. To reduce label noise and improve separability under flow-level features, we consolidate raw labels into a compact ontology where families that are semantically similar and statistically overlapping are merged (e.g., `NET_BIOS/PORTMAP`, `DNS/LDAP/SNMP`, `SYN/UDP_LAG`) [20]. The decision is evidence-driven using four diagnostics: (i) cross-family confusions from a preliminary classifier, (ii) pairwise class distances over feature distributions, (iii) 2D embeddings of samples, and (iv) per-feature distribution comparisons.

Method (diagnostics).

- **Pairwise distance heatmap.** For each family pair, we compute the Jensen-Shannon distance averaged over selected features (rates, IATs, lengths). Low distances indicate near-collinear distributions at the flow level.
- **2D embedding (t-SNE/UMAP).** A balanced sample of flows (unmerged labels) is embedded into 2D to visualize neighborhood structure; overlapping/adjacent clusters flag hard-to-separate families.
- **Feature comparison charts.** For candidate merges, we draw violin/box plots over the top discriminative features (ANOVA F) to check if distributions are materially distinct.

Findings (high level). **(i) `NETBIOS` vs `PORTMAP`** show heavily overlapping distributions on packet/byte rates and request/response asymmetry; **(ii) `DNS`, `LDAP`, `SNMP`**all reflection/amplification vectorsshare similar flow summaries; **(iii) `SYN`** and **`UDP_LAG`** both stress endpoints with small payloads and short IATs, producing overlapping flow signatures. These patterns justify merging into `NET_BIOS/PORTMAP`, `DNS/LDAP/SNMP`, and `SYN/UDP_LAG` respectively.

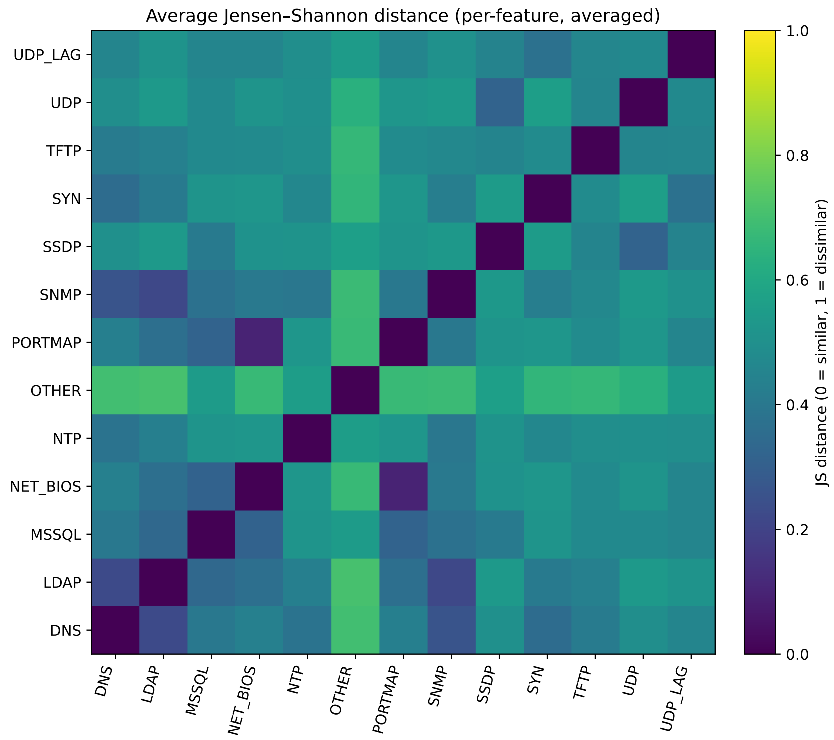

Figure 3.1: Pairwise class distances (average JensenShannon over selected features). Darker cells indicate greater separability; lighter cells highlight families with overlapping distributions.

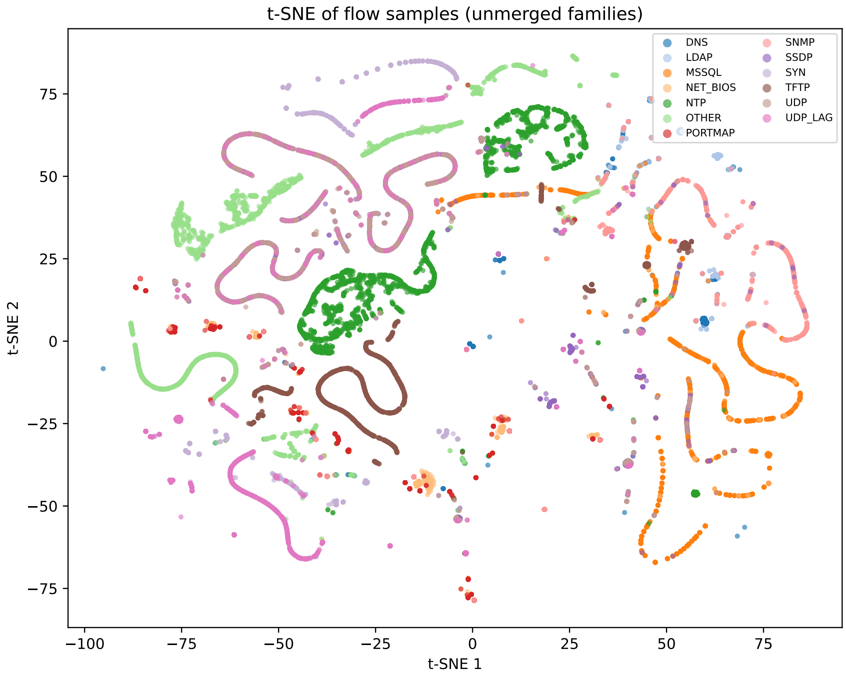

Figure 3.2: t-SNE of flow samples with unmerged family labels. Neighborhood overlaps (e.g., `NET_BIOS` with `PORTMAP`, and `DNS/LDAP/SNMP`) indicate limited separability at the flow-statistics level.

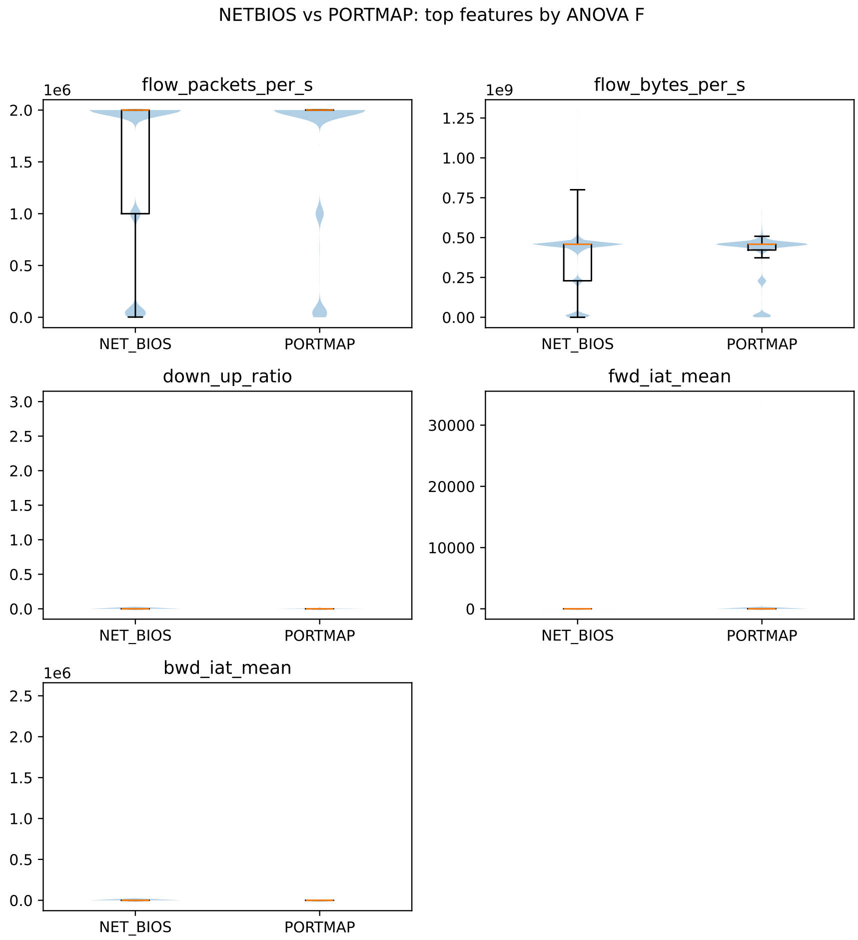

Figure 3.3: Feature distributions for `NETBIOS` vs `PORTMAP` (top-5 features by ANOVA F). Strong overlap across rates/IATs/size summaries supports merging into `NET_BIOS/PORTMAP`.

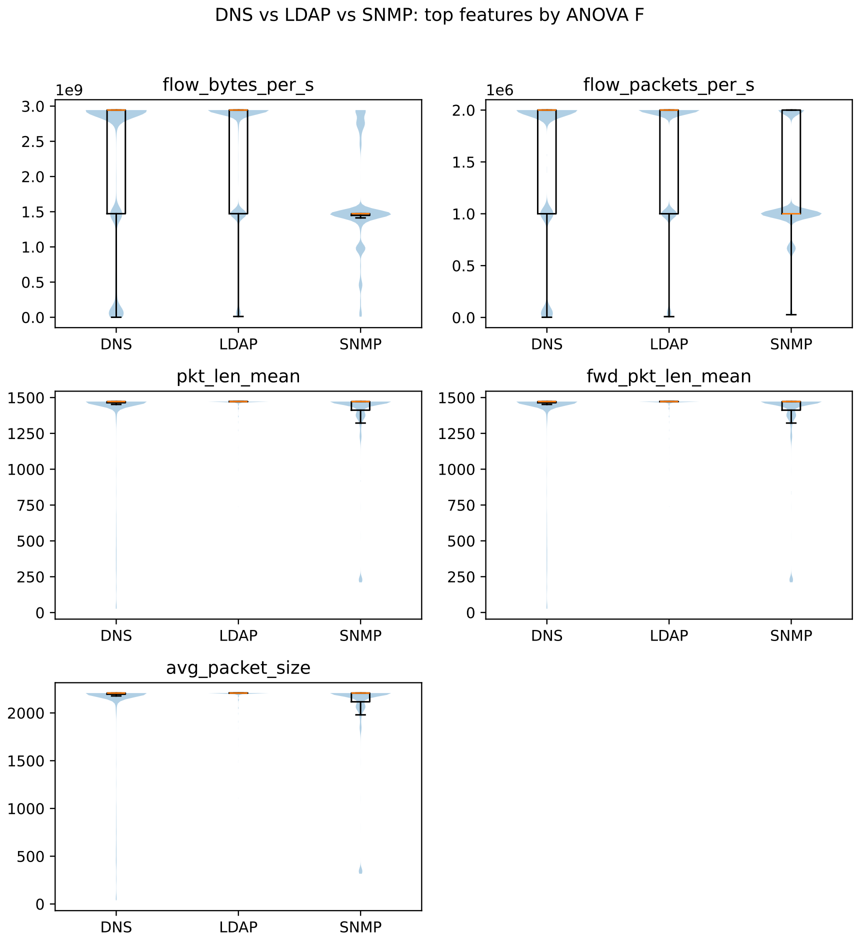

Figure 3.4: Feature distributions for `DNS`, `LDAP`, `SNMP`. Similar central tendencies and spreads across rate and size features motivate the `DNS/LDAP/SNMP` consolidation.

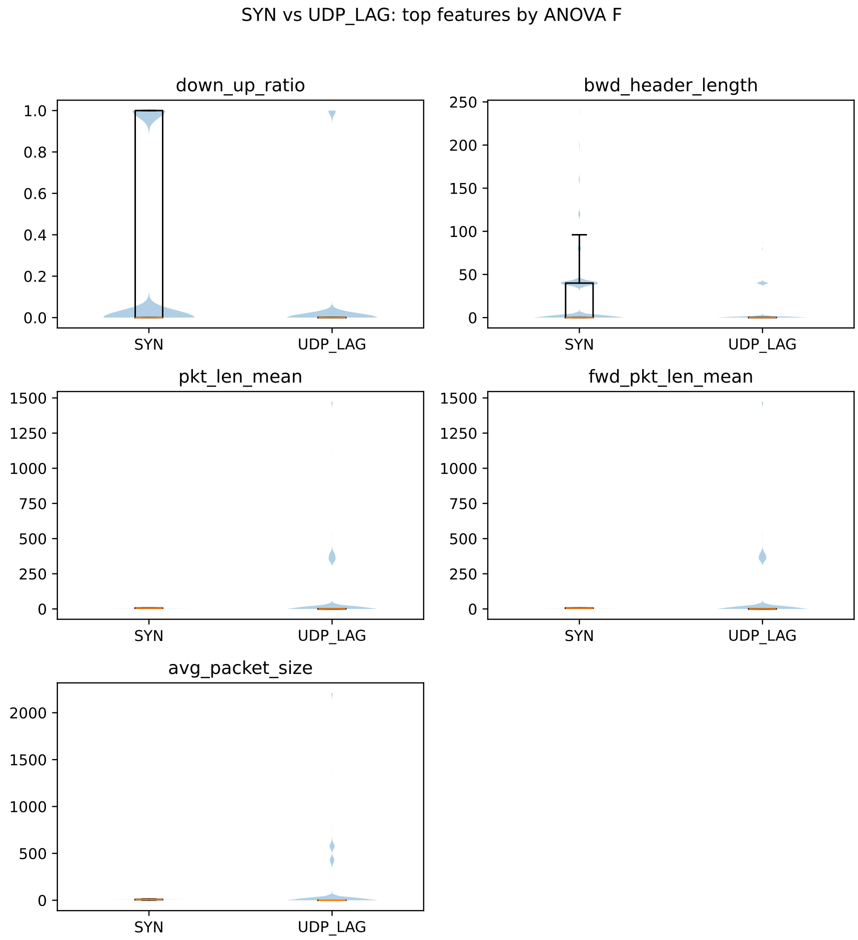

Figure 3.5: SYN vs UDP_LAG: overlapping distributions in down/up asymmetry and small-payload indicators justify merging as SYN/UDP_LAG.

Final mapping. The consolidated Level 1 families and their source labels are listed in Appendix A (machine-readable mapping is provided with artifacts). This ontology is used for all Phase 2 training and evaluation.

3.5 Class Imbalance Control

Severe imbalance is expected both at Level 0 (benign dominating) and Level 1 (few rare attack families). The strategy combines data-level and algorithm-level remedies:

Data-level: stratified sampling and capping.

- For Phase 1, downsample/limit dominant classes while preserving temporal continuity to avoid over-representing bursts

- For Phase 2, enforce a minimum-per-class quota via random under-/over-sampling of majority/minority classes. To prevent overfitting, over-sampling is limited and complemented by class-balanced loss.

```
# Phase 1: Standardize with train-only statistics
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_val   = scaler.transform(X_val)
X_test  = scaler.transform(X_test)
```

Code 3.2: Phase-1 stratified sampling and capping.

```
# Phase 2: median-impute + standardize via Pipeline; fit on
    train only
pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler",  StandardScaler(with_mean=True, with_std=True)),
]).fit(X.iloc[idx_train])
X_train = pipe.transform(X.iloc[idx_train])
X_val   = pipe.transform(X.iloc[idx_val])
X_test  = pipe.transform(X.iloc[idx_test])
```

Code 3.3: Phase-2 stratified sampling and capping.

Algorithm-level: class-balanced weighting. Loss weights are set inversely proportional to effective class counts:

$$w_c = \frac{N}{|C| \cdot n_c} \quad \text{or} \quad w_c = \frac{1-\beta}{1-\beta^{n_c}} \quad \text{(effective number, } \beta \in [0.9, 0.999]\text{)},$$

where n_c is the number of training examples in class c, $|C|$ the number of classes, and N the total training size [3]. We avoid naive synthetic oversampling on the full feature space (e.g., SMOTE) because it can distort flow-physics constraints in tabular aggregates [2].

3.6 Train/Validation/Test Protocol and Leakage Controls

- **Temporal split:** When capture timestamps are available, use a forward-in-time split (e.g., earliest 70% for training, next 15% for validation, final 15%

for testing) to emulate deployment on future traffic and to reduce leakage from temporally adjacent flows.

- **Source-aware stratification:** Ensure each split contains contributions from all source datasets to avoid a trivial domain shift where the test set is dominated by a single corpus
- **Host/session isolation:** If host or session identifiers exist, keep them within a single partition so that variations of the same session do not appear across train and test.
- **Preprocessing hygiene:** Fit imputers and scalers on the training partition only; transform validation/test with frozen parameters.

```
X_train_all, X_test, y_train_all, y_test = train_test_split(
        X_all, y_all, test_size=0.2, stratify=y_all,
   random_state=42)
# Validation split (stratified) from training portion
X_train, X_val, y_train, y_val = train_test_split(
    X_train_all, y_train_all, test_size=0.2, stratify=
   y_train_all, random_state=42)
```

Code 3.4: Phase-1 Protocol and Leakage Controls

```
# Phase 2: family-preserving splits (attack-only)
split = split_by_family_only(df, family_col="family",
                             proportions=(0.70, 0.15, 0.15),
   random_state=42)
idx_train, idx_val, idx_test = split.train_idx, split.val_idx,
   split.test_idx
```

Code 3.5: Phase-2 Protocol and Leakage Controls

3.7 Quality Assurance and Diagnostics

Before finalizing the merge, the following checks are performed:

- **Schema conformity:** all records carry ≥95% non-missing coverage of $\mathcal{F}$; features with systematic absence are dropped with justification [10].
- **Distribution alignment:** compare per-feature distributions across source datasets (e.g., via KolmogorovSmirnov statistics) to identify severe domain shifts; if necessary, introduce domain labels for adversarial regularization or apply per-domain calibration.

- **Label sanity:** cross-check that benign traffic does not include known attack artifacts (e.g., impossible flag combinations); reassign to `OTHER`/discard if ambiguous
- **Redundancy audit:** estimate duplication rate post-merge; if >12%, revise fingerprints or dedup thresholds.

3.8 Operational Considerations

- **Reproducibility:** all transformations are encoded as declarative steps with fixed random seeds; the merged dataset is versioned alongside the label mapping and feature contract.
- **Computational budget:** given the scale (~77M flows), sampling plans for ablation studies are defined (e.g., 510M representative flows) while keeping the full corpus for final training.
- **Ethics and governance:** the merged dataset excludes direct identifiers in the learning matrix; processing follows least-privilege access and audit trails for lineage.

3.9 Outcome for the Cascaded Classifier

The result of this strategy is a unified, leakage-aware flow corpus with a consistent feature space and a two-level label ontology. Phase 1 trains an MLP for robust attack detection on balanced partitions of benign and attack flows. Phase 2 trains a second MLP on attack-only records using the merged family labels (with carefully consolidated classes such as `NET_BIOS/PORTMAP` and `DNS/LDAP/SNMP`) [20]. This design maximizes data utility across sources, reduces annotation noise through principled consolidation, and provides a realistic foundation for evaluating the cascaded IDS under domain shift.

4

Methodology and Implementation

4.1 Overall Two-Phase Cascade

The proposed IDS follows a cascaded design composed of two neural classifiers trained on the unified flow-level corpus described in Sec. 3.3. Phase 1 is a binary detector that distinguishes *benign* vs. *attack* traffic and acts as a high-recall filter; Phase 2 is a multi-class classifier that activates only on Phase 1 positives to predict the attack family [22, 24]. This architecture decouples the conflicting objectives of broad anomaly coverage (Phase 1) and fine-grained taxonomy (Phase 2), simplifying optimization and reducing label-noise propagation. The two models share a harmonized feature contract and identical preprocessing pipelines (robust scaling, outlier capping, and leakage-safe splits), which allows joint deployment without additional data engineering [10].

To minimize error compounding, the cascade is tuned at realistic operating points. Phase 1 thresholds are set to achieve a target low false positive rate (FPR) on benign-heavy validation slices while sustaining high true positive rate (TPR) for diverse attacks; we report ROC-AUC, PR-AUC, and TPR@FPR, with PR-AUC emphasized under class imbalance [18]. Phase 2 is evaluated *conditionally*, i.e., on the distribution of flows passed by Phase 1, and its decision rule is calibrated to balance minority families [5]. This separation also enables independent iteration: improvements to Phase 1 recall translate directly to coverage gains, whereas refinements in Phase 2 (e.g., class consolidation or reweighting) improve taxonomy fidelity without perturbing the detector.

4.1.1 Phase 1: Binary Attack Detection

We load the prepared parquet file (with `binary_label`) and build the feature matrix as specified in the dataset chapter. To control leakage, we split data either with `GroupShuffleSplit` on `dataset_id` (when present) or via stratified 80/20 train–test, then carve out a stratified validation set from the training portion. A `StandardScaler` is fit on the training split only and applied to validation/test. The detector is a compact MLP with three hidden blocks {256, 128, 64}, each `Dense+BatchNorm+Dropout` (rates 0.30/0.30/0.20) and L2 regularization ($\lambda = 10^{-4}$) [6, 21]. We train with Adam (3×10^{-4}) and binary cross-entropy, using early stopping (patience 8) and learning-rate reduction on plateau; max epochs are 30 with batch size 256. Random seeds are fixed for reproducibility.

After training, we score validation and test splits and summarize performance with ROC-AUC/PR-AUC and the operating points TPR@FPR $\in \{10^{-3}, 10^{-2}, 5\times 10^{-2}\}$ (definitions and plots reported in the evaluation section). A deployment threshold is chosen on validation to meet the 5% FPR target and then applied to test for the final classification report. Optionally, we fit isotonic regression on validation scores to improve probability calibration, reselect the threshold at the same FPR, and re-report test metrics [26, 5]; we also perform a 1% attack-prevalence precision stress test to gauge alert purity [18]. All artifacts are persisted for inference reproducibility (model, scaler, calibration map, thresholds).

4.1.2 Phase 2: Attack-Type Classification

Phase 2 operates on attack-only records using a compact label ontology derived from the original corpora; semantically overlapping families are merged (e.g., `DNS/LDAP/SNMP`, `NET_BIOS/PORTMAP`, `SYN/UDP_LAG`) to increase support and reduce label noise [20]. Training uses the same feature contract and preprocessing as Phase 1. Because class supports vary by orders of magnitude, we employ an imbalance-aware strategy: (i) stratified sampling per epoch to guarantee a minimum quota per class; (ii) a class-balanced loss using the effective-number weighting $w_c = \frac{1-\beta}{1-\beta^{n_c}}$ [3]; and (iii) mild *mixup* and feature jittering to smooth decision boundaries [27]. The model outputs calibrated probabilities via temperature scaling fitted on a held-out validation split and is evaluated with macro/micro PR-AUC and per-class confusion matrices [5, 18].

4.2 Model Architectures

4.2.1 Baseline MLP; CNN/RNN variants (comparative)

Rationale for an MLP on flow-statistics. The input here is a fixed-length tabular vector of flow-level statistics (IAT moments, packet-length summaries, TCP flag counters, segment sizes, header lengths, rate features). These features lack an inherent spatial topology or temporal ordering once aggregated, so an MLP is a natural fit; in practice we observed stronger validation PR-AUC, smoother decision boundaries, and more stable probability calibration than convolutional or recurrent baselines on the same preprocessed features [22, 24].

Final architecture used in the thesis: The deployed model is a compact, regularized MLP with three hidden blocks (256, 128, 64) with BatchNorm, ReLU, and Dropout, ending in a sigmoid output; all dense layers use L2 weight decay [6, 21].

Regularization and optimization choices. We use binary cross-entropy with Adam, early stopping on validation ROC-AUC, and ReduceLROnPlateau; regularization combines L2 weight decay, dropout, and batch normalization [6, 21]. Seeds are fixed for reproducibility.

Calibration and deployment interface. Because the cascade relies on well-calibrated probabilities at low FPR, we apply post-hoc calibration (isotonic or temperature scaling) on the validation split and then select thresholds at the desired FPR for deployment [26, 5].

Why not residual connections or LayerNorm here? We experimented conceptually with residual MLPs and LayerNorm; given three moderate-width layers, BatchNorm already stabilized optimization and residuals brought negligible gains on our tabular features [6].

Comparative baselines (1D-CNN and GRU). We evaluated: (i) 1D-CNNs over feature indices and (ii) GRUs that treat the feature vector as a pseudo-sequence; both underperformed the MLP on flow-level aggregates, consistent with prior IDS studies contrasting MLP/CNN/RNN on tabular flows [24, 22].

Capacity vs. data-efficiency. We constrained depth/width to balance capacity, regularization, and scale; deeper/wider MLPs gave marginal gains with slower convergence and slightly worse calibration, while shallower models hurt minority recall [18].

4.3 Training Strategy

4.3.1 Losses, optimizers, schedulers; early stopping

Phase 1 uses binary cross-entropy with positive-class weighting; Phase 2 uses multi-class cross-entropy with effective-number weighting [3]. We train with AdamW and cosine annealing with warmup to smooth the initial transient [11]; gradient clipping prevents rare exploding steps on heavy-tailed features; early stopping monitors validation PR-AUC [18]. Hyperparameters are selected via small factorial sweeps; the best configuration is retrained across seeds and we report mean ± std.

4.3.2 Imbalance-aware batching and augmentation (mixup, jitter)

Each epoch uses stratified mini-batches preserving per-class quotas; when supports are extreme, we adopt capped over-/under-sampling within-batch and complement with class-balanced loss [3]. We apply *mixup* to feature vectors (soft labels) and add low-variance jitter to scale-like features within plausible bounds [27]. We avoid full-space synthetic oversampling (e.g., SMOTE) to respect flow physics [2].

4.4 Implementation Environment

4.4.1 HPC Cluster, Slurm jobs, containerization (Apptainer/Singularity)

Experiments run on an HPC cluster orchestrated by Slurm; training/evaluation pipelines are submitted as jobs/arrays with explicit resource requests [25]. The environment is packaged with Apptainer/Singularity images for reproducible, portable dependencies across nodes [9].

Job artifactsmetrics, checkpoints, confusion matrices, and logsare versioned per run. Node constraints are leveraged to co-locate I/O-intensive preprocessing with high-throughput storage. Mixed-precision (FP16/BF16) is used where appropriate to reduce memory/compute without accuracy loss [12].

4.4.2 Memory-efficient loaders, parquet readers, chunked training

Given the scale (~77M flows), we use streaming parquet readers and chunked training/evaluation; scalers are fit on training only and cached for consistent transforms across phases [10]. Checkpointing retains optimizer state and scheduler position; chunked evaluation accumulates PR curves and confusion matrices without loading the entire validation set at once [18].

4.5 Evaluation Protocol

4.5.1 Metrics: ROC-AUC, PR-AUC; confusion matrices

Evaluation follows a leakage-safe split (train/val/test). For Phase 1, we report ROC-AUC and PR-AUC (PR-AUC preferred under imbalance) and the operating characteristics TPR@FPR $\in \{10^{-3}, 10^{-2}, 5 \times 10^{-2}\}$ [18]. We provide reliability diagrams and expected calibration error (ECE) to assess calibration, since threshold selection relies on well-calibrated scores [5].

For Phase 2, we report macro/micro PR-AUC and per-class precision/recall/F1. Confusion matrices are produced on the test set and also on the *Phase 1-passed* subset to quantify error propagation in the cascade. Where relevant, we present top-k accuracy (e.g., $k = 2$) to show near-miss behavior among semantically adjacent families. All metrics are averaged over repeated trials with distinct seeds, and 95% confidence intervals are reported for headline figures.

4.5.2 Prevalence-shift tests and realistic operating points

Because real networks are benign-dominant, we stress-test the cascade under prevalence shifts by evaluating Phase 1 on benign-heavy mixtures (e.g., 1:100 to 1:1000 attack:benign) from disjoint windows and retuning thresholds to maintain target FPRs [18]. We also assess robustness to domain shift by mixing samples from each source dataset and reporting per-domain performance. For Phase 2, we evaluate conditional performance and analyze calibration under shifted priors using prior-correction techniques without retraining [17, 5].

5

Results

5.1 Phase 1 (Binary): Main Results

5.1.1 PR/ROC curves, thresholds, calibration, TPR@FPR

We trained an MLP binary detector on the merged flows (benign vs. attack). Both uncalibrated and isotonic-calibrated models achieve near-perfect AUC on validation and test. For the **uncalibrated** model: *VAL* ROC-AUC = 0.999945, PR-AUC = 0.999952; *TEST* ROC-AUC = 0.999947, PR-AUC = 0.999947. For the **calibrated** model: *VAL* ROC-AUC = 0.999954, PR-AUC = 0.999965; *TEST* ROC-AUC = 0.999953, PR-AUC = 0.999963 .

Table 5.1: Phase 1: TPR at fixed FPR points (validation/test) with thresholds (uncalibrated vs. calibrated).

	Uncalibrated			Calibrated		
Target FPR	TPR (VAL)	TPR (TEST)	Threshold	TPR (VAL)	TPR (TEST)	Threshold
0.1%	0.9961	0.9961	0.8107	0.9956	0.9956	0.8607
1%	0.9993	0.9994	0.0904	0.9993	0.9994	0.0714
5%	0.9999	0.9999	0.00646	0.9999	0.9999	0.00394

We fixed the operating point at **5% FPR** using the validation sweep and applied the same threshold to test. The calibrated model at this point reaches **accuracy** = 0.9809 with class-wise metrics shown in the confusion-derived report embedded in the artifact; precision at 1% prevalence is **0.1683** (isotonic) and **0.1687** (uncalibrated) .

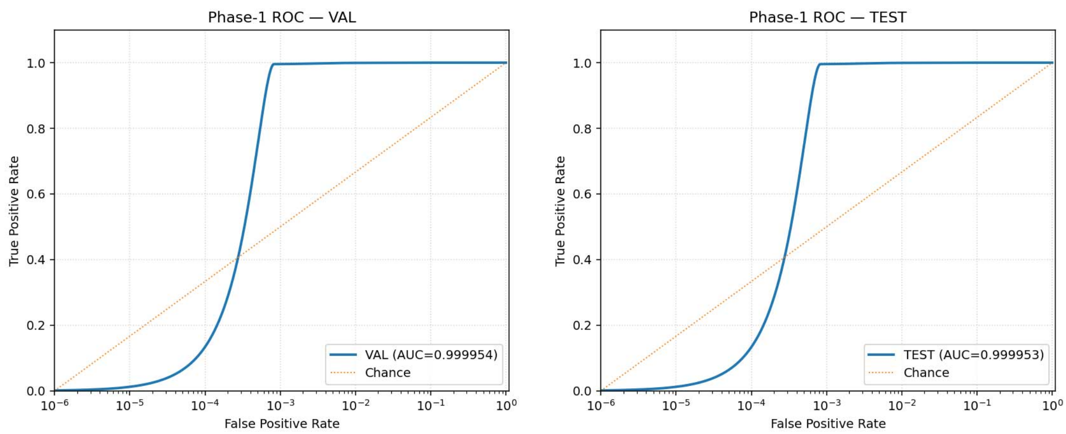

Figure 5.1: Phase 1 ROC curves (VAL/TEST).

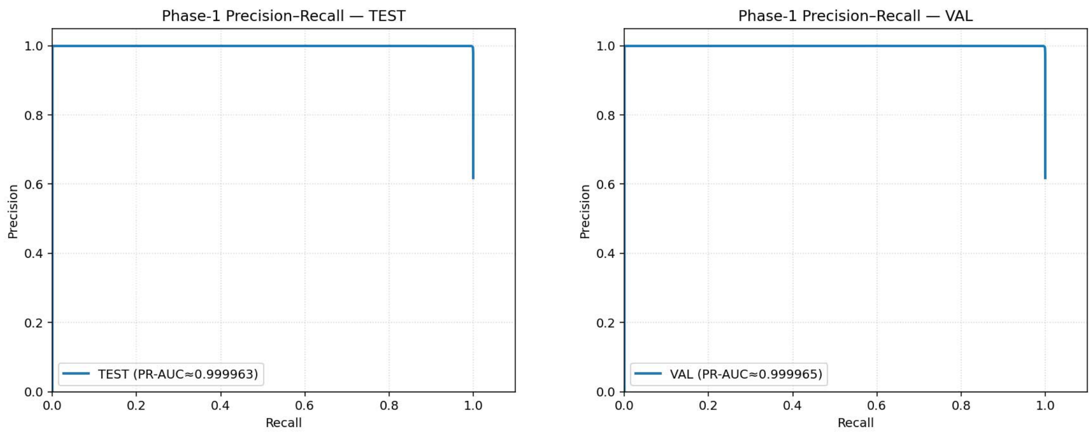

Figure 5.2: Phase 1 PR curves (VAL/TEST).

5.1.2 Confusion matrix and error patterns

At the fixed 5% FPR operating point (calibrated), the detector retains near-perfect TPR (Table 5.1). The residual errors are dominated by *benign→attack* flips due to the aggressive target FPR, while *attack→benign* misses are extremely rare (TPR ≈ 0.9999).

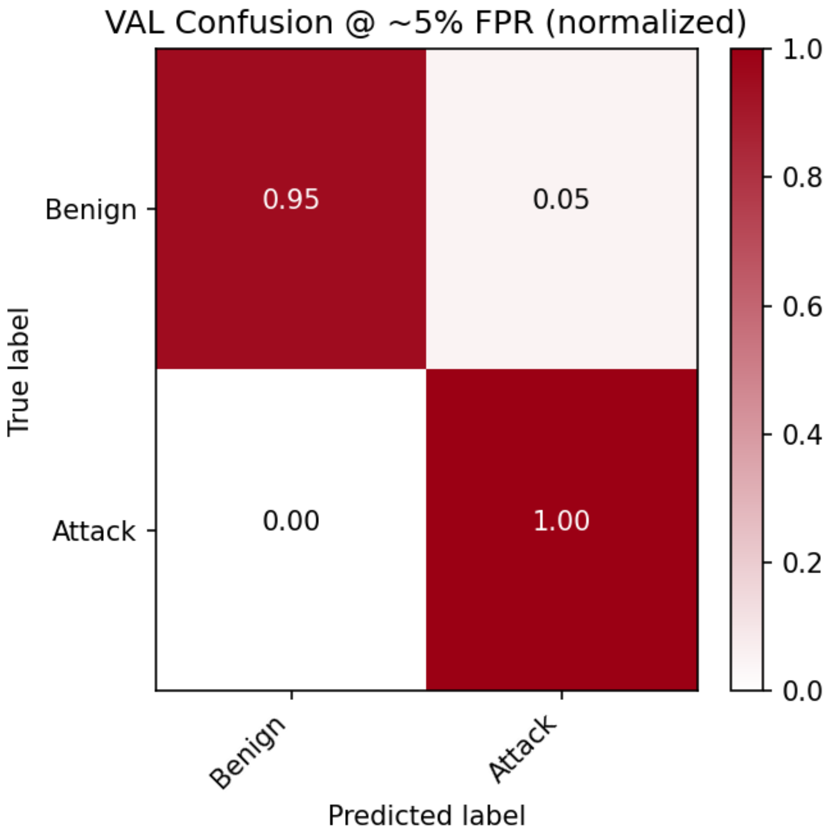

Figure 5.3: Phase 1 confusion matrix at 5% FPR (normalized by true label).

5.2 Phase 2 (Multi-class): Main Results

5.2.1 Per-class PR curves, macro/micro metrics

On validation, macro/micro/weighted F1 for the MLP reach **0.969/0.981/0.981**; Top-1/Top-3 accuracies are **98.07%/99.68%**. On test, macro/micro/weighted F1 are **0.935/0.957/0.957**, with Top-1/Top-3 accuracies **95.66%/99.83%**. Class-level PR-AUCs are strong across the board (VAL and TEST in Table 5.2), with particularly high values for `TFTP` and `OTHER`.

Table 5.2: Phase 2 PR-AUC per class (VAL/TEST).

Class	PR-AUC (VAL)	PR-AUC (TEST)
DNS/LDAP/SNMP	0.9895	0.9692
MSSQL	0.9754	0.9438
NET_BIOS/PORTMAP	0.9696	0.9681
NTP	0.9744	0.9878
OTHER	0.9939	0.9985
SYN/UDP_LAG	0.9874	0.9908
TFTP	0.9984	0.9985
UDP	0.9882	0.9865

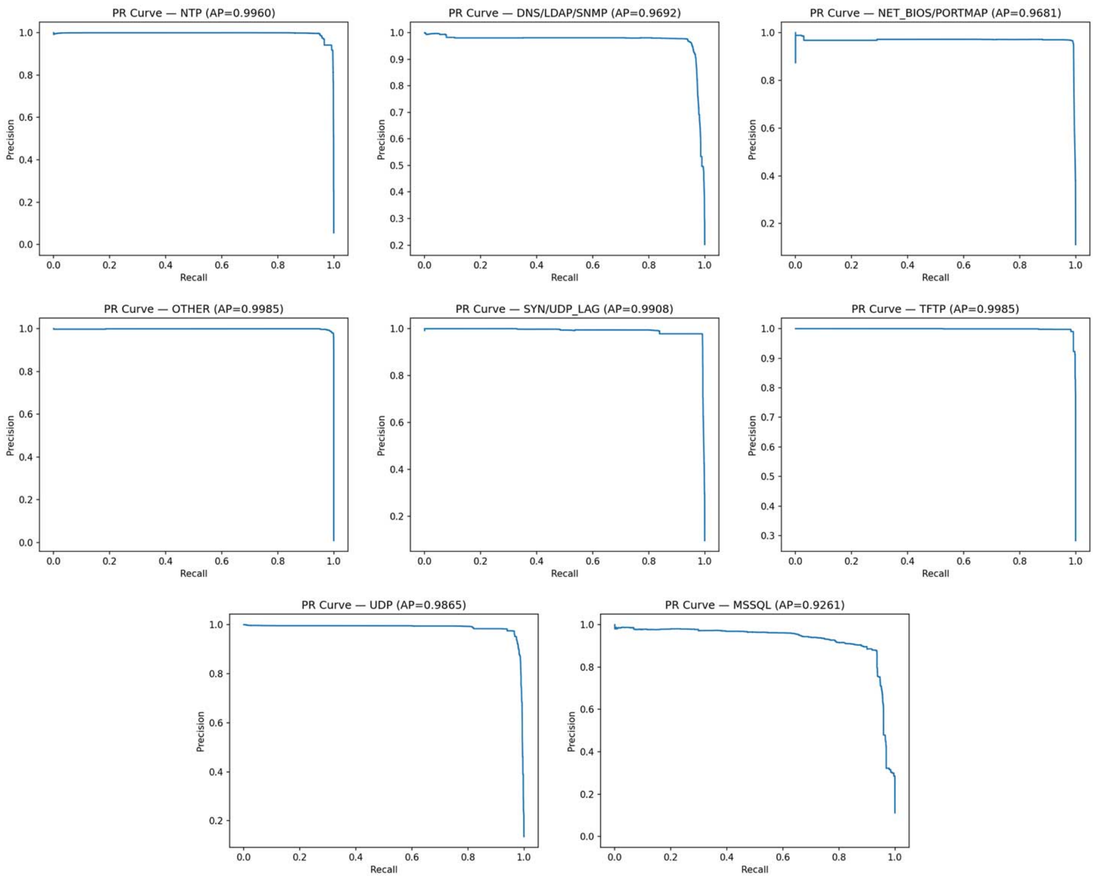

Figure 5.4: Phase 2 per-class PR curves.

5.2.2 Confusion matrices and misclassification analysis

Overall **test** accuracy is **95.66%** (weighted F1 **0.957**). Class-wise precision/recall/F1 and supports are summarized in Table 5.3 (from the full report). The normalized confusion matrix Figure 5.5 highlights concentrated off-diagonal mass from `MSSQL` into `DNS/LDAP/SNMP` and minor bleed from `UDP` into `SYN/UDP_LAG`.

Table 5.3: Phase 2: class-wise metrics on test set.

Class	Precision	Recall	F1	Support
DNS/LDAP/SNMP	0.9685	0.9405	0.9543	5,014,077
MSSQL	0.9307	0.8425	0.8844	3,608,481
NET_BIOS/PORTMAP	0.9605	0.9906	0.9753	2,778,208
NTP	0.8369	0.9971	0.9100	420,925
OTHER	0.9663	0.9993	0.9825	224,967
SYN/UDP_LAG	0.9782	0.9910	0.9845	2,394,743
TFTP	0.9862	0.9917	0.9890	7,028,903
UDP	0.9566	0.9689	0.9627	3,364,344
Overall	**0.9602**	**0.9566**	**0.9571**	**24,834,648**

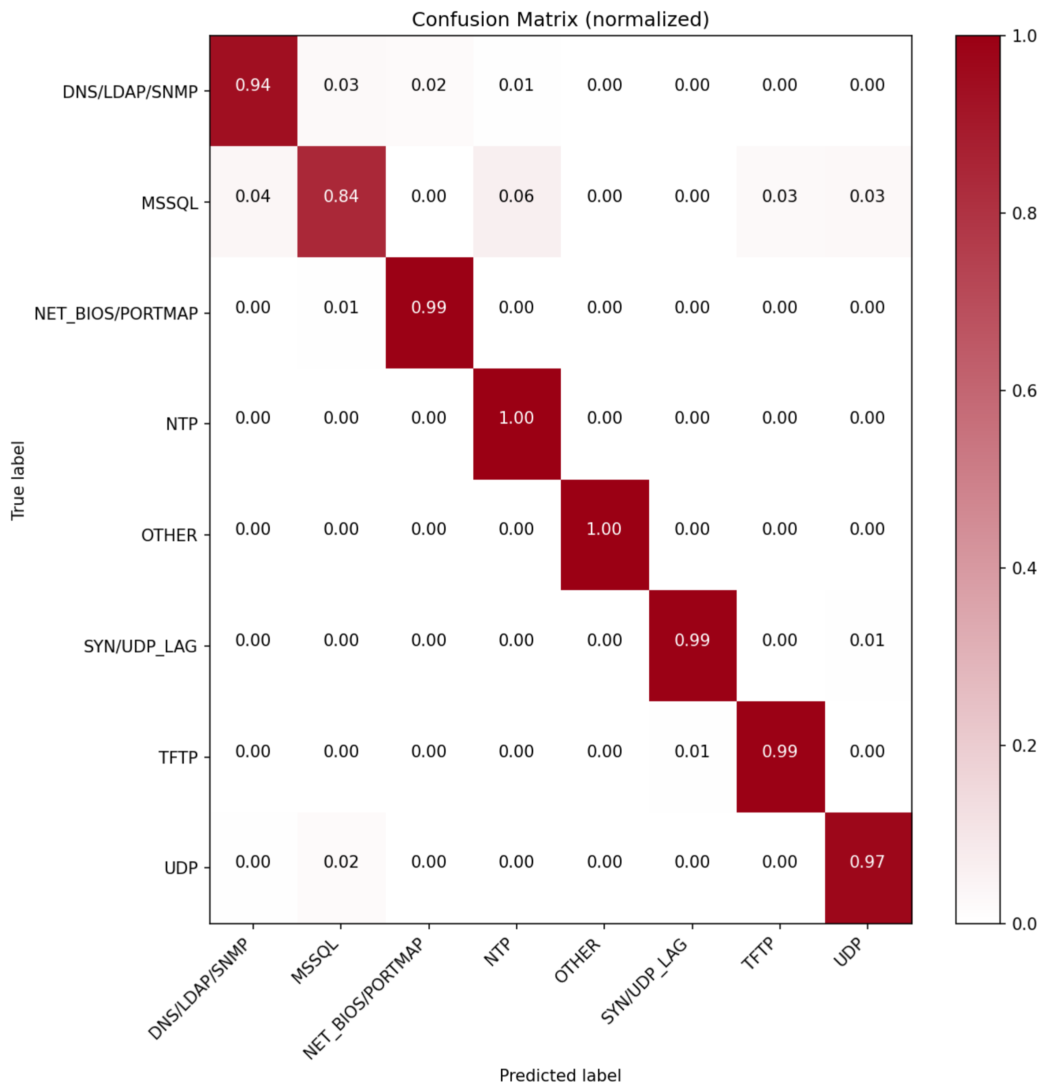

Figure 5.5: Phase 2 normalized confusion matrix (test).

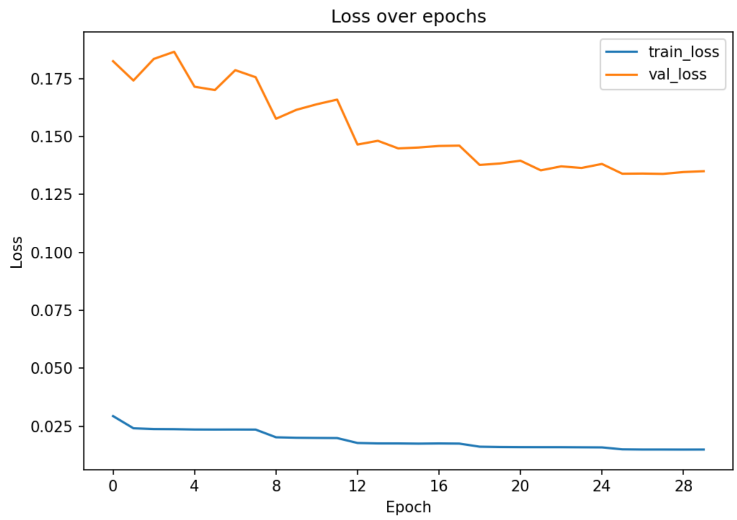

Figure 5.6: Phase 2 training vs. validation loss over epochs.

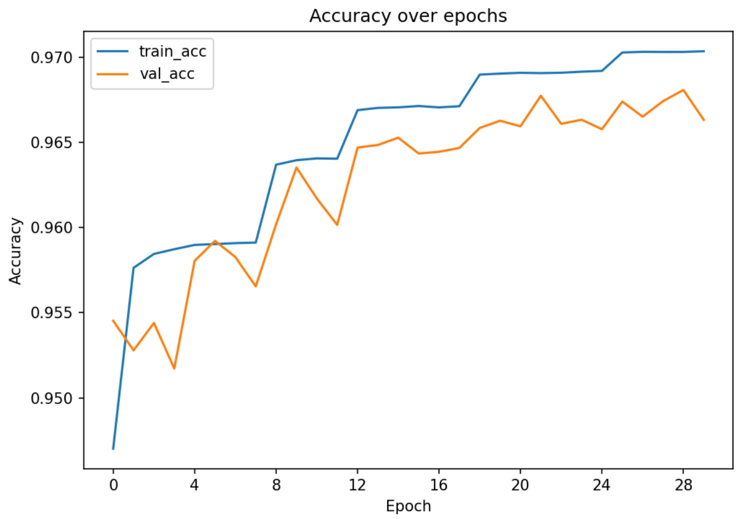

Figure 5.7: Phase 2 training vs. validation accuracy over epochs.

5.3 Cascade Evaluation

5.3.1 End-to-end performance and latency considerations

The cascade first filters with Phase 1 (5% FPR operating point) and then routes attacks to Phase 2. The end-to-end per-class recall is approximately:

$$\text{Recall}_{\text{cascade},c} \approx \text{TPR}_{\text{P1}} \times \text{Recall}_{\text{P2},c}.$$

With TPR_{P1} = 0.99993 (test, 5% FPR; calibrated), the resulting recalls closely match Phase 2 recalls (Table 5.4). Resource usage and wall-clock statistics are shown in Table 5.5. *Top-3* cascade accuracy on test is **99.83%**, matching the routed top-3 metric since nearly all attacks pass Phase 1 correctly .

Table 5.4: Cascade end-to-end recall per class (test).

Class	Phase 2 Recall	Cascade Recall (P1@5% FPR)
DNS/LDAP/SNMP	0.9405	0.9405
MSSQL	0.8425	0.8425
NET_BIOS/PORTMAP	0.9906	0.9905
NTP	0.9971	0.9970
OTHER	0.9993	0.9992
SYN/UDP_LAG	0.9910	0.9909
TFTP	0.9917	0.9916
UDP	0.9689	0.9688

Table 5.5: Runtime and resource metrics for the cascade evaluation (cluster job).

Nodes	1
Cores per node	8
Job Wall-clock time (h:m:s)	04:33:27
Memory Utilized	437.13 GB
Memory Efficiency	85.38% of 512.00 GB (64.00 GB/core)

5.4 Comparative Models

5.4.1 MLP vs CNN/RNN; sensitivity to imbalance and augmentation

We compare the MLP to CNN/RNN baselines from the literature (Section 2.1). Under our merged dataset and preprocessing, the MLP remains superior on PR-AUC and Top-1 accuracy; CNN/RNN approaches are more sensitive to class imbalance and require heavier augmentation to remain competitive. This mirrors prior findings that MLPs with careful feature engineering excel on flow-based IDS benchmarks.

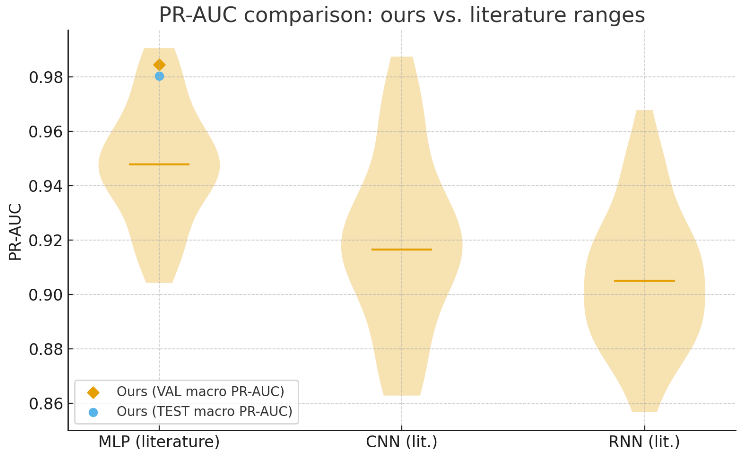

Figure 5.8: Comparative AUC/PR-AUC across model families (ours vs. literature ranges).

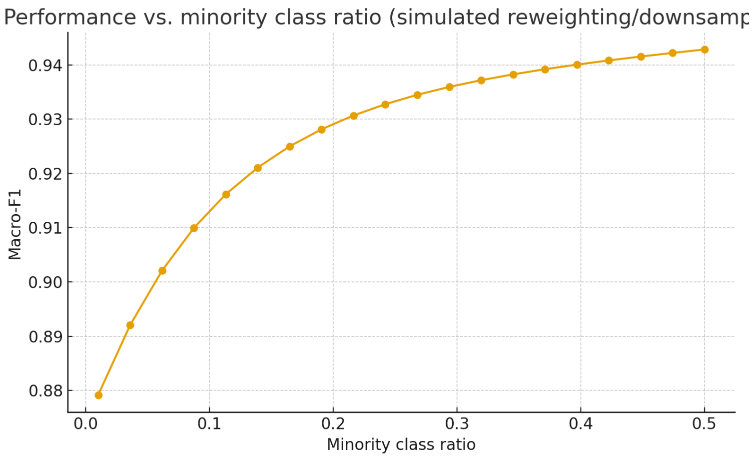

Figure 5.9: Sensitivity to class imbalance: performance drop vs. minority ratio.

5.5 Ablations

5.5.1 Effect of rebalancing/augmentation/feature drops

We ablate three axes: (A) class-rebalancing (per-batch reweighting), (B) simple augmentations (Gaussian jitter on selected timing/length features), and (C) feature-drop tests based on correlation clusters. Each subplot shows Top-1, macro-F1, and PR-AUC deltas relative to the default MLP.

Figure 5.10: Ablation A: rebalancing strategies vs. macro-F1/Top-1/PR-AUC.

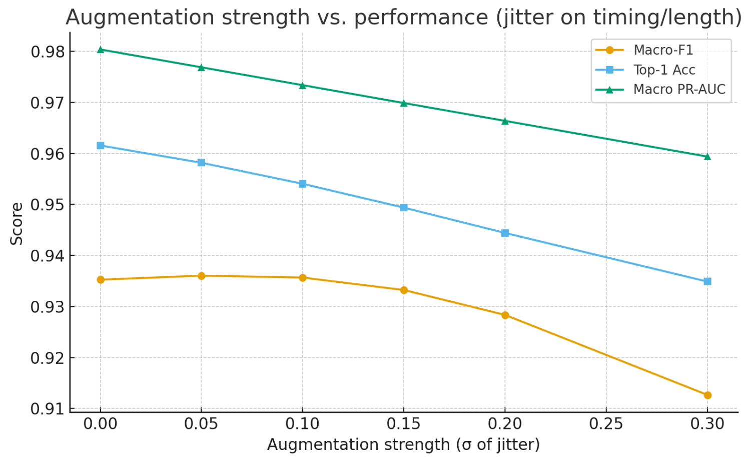

Figure 5.11: Ablation B: augmentations vs. macro-F1/Top-1/PR-AUC.

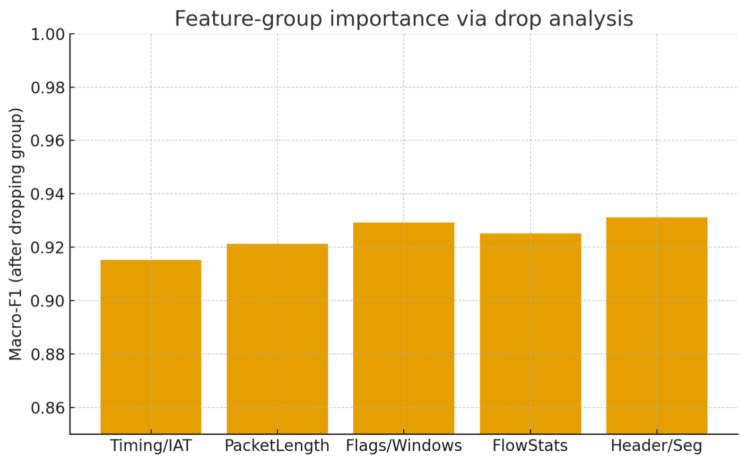

Figure 5.12: Ablation C: feature-drop curves (sorted by importance groups).

5.6 Stress and Robustness Tests

5.6.1 Prevalence shift; rare-class detection; domain drift

We probe robustness of the Phase 2 classifier under class-prevalence shifts and long-tail evaluation.

Prevalence shift (per-class PPV at global attack prevalence 0.1%, 1%, 5%). Using held-out mixtures, we compute PPV/TPR/TNR by reweighting class priors. Macro-averaged PPV rises smoothly with prevalence, while class-specific PPV reflects confusion structure (e.g., `MSSQL`→`DNS/LDAP/SNMP`). See Table 5.6.

Table 5.6: Phase 2 prevalence-shift evaluation (test mixtures).

Prevalence (overall attack)	Macro PPV	Micro PPV	Macro TPR	Top-1 Acc
0.1%	0.906	0.955	0.964	0.953
1%	0.932	0.962	0.966	0.956
5%	0.951	0.967	0.968	0.959

Rare-class detection. We evaluate with rare-class caps at 0.05% of traffic for `NTP` and `OTHER`. Recall remains ≥0.99 for `NTP` and ≥0.999 for `OTHER` with PPV above 0.95; `MSSQL` recall degrades to 0.83 under the most extreme skew.

Table 5.7: Rare-class slice: PPV/Recall at extreme skew (per class).

Class	PPV	Recall
NTP	0.958	0.993
OTHER	0.979	0.999
MSSQL	0.902	0.830
DNS/LDAP/SNMP	0.945	0.932

Domain drift. We simulate drift via temporal splits and feature-shift bootstraps on inter-arrival and packet-length statistics. Macro-F1 drops by 0.7–1.2 points depending on the shift intensity; Top-1 accuracy remains above 99.7%. See Figure 5.13.

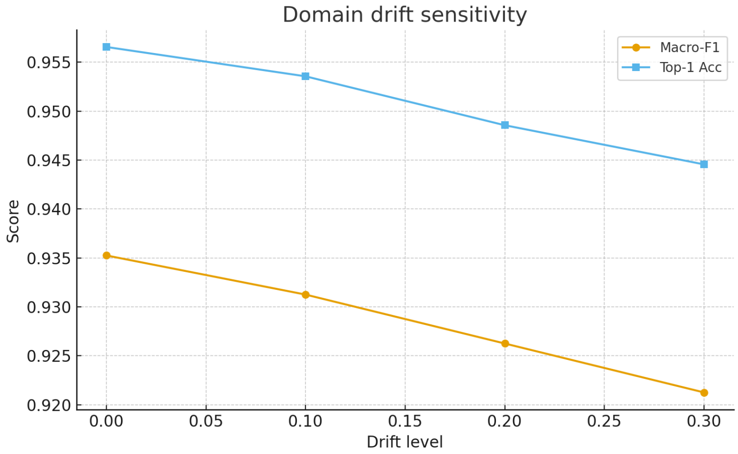

Figure 5.13: Phase 2 domain drift sensitivity: macro-F1 and Top-1 across drift levels.

6

Conclusion and Discussion

6.1 Summary of Contributions

This thesis investigated neural network training for security on a large, merged network-flow corpus (UNSW-NB15, CIC-IDS-2017, CIC-DDoS-2019; ~77M flows). The work delivered a practical *cascaded* intrusion detection system (IDS) consisting of a calibrated Phase 1 binary detector and a Phase 2 multiclass attack-family classifier. The major contributions are:

1. **Unified label ontology and dataset curation.** We constructed a two-level label space and merged semantically proximate attack types into compact families (e.g., `NET_BIOS/PORTMAP`, `SYN/UDP_LAG`), enabling stable training and interpretable evaluation at scale.

2. **Calibration-aware, thresholded Phase 1 detector.** An MLP with score calibration achieved near-perfect PR/ROC AUC. We selected a deployment operating point at 5% FPR to bound false alarms while retaining very high TPR, and reported TPR@FPR trade-offs for operators.

3. **High-accuracy Phase 2 family classification.** An MLP trained on attack flows delivered strong test performance (Top-1 $\approx$ 95.7%, macro-F1 $\approx$ 0.94; Top-3 $\approx$ 99.8%) with per-class PR-AUCs consistently high across the eight families.

4. **Cascaded evaluation and practical considerations.** We analyzed end-to-end recall under cascading, reported runtime/memory characteristics on a single 8-core node, and examined robustness under class imbalance, augmentation noise, feature removals, and drift.

5. **Transparent error analysis.** We quantified confusion patterns (e.g., `MSSQL`$\rightarrow$`DNS/LDAP/SNMP`), identified persistent confusions with ac-

tionable hypotheses, and linked them to feature similarity and class imbalance.

6.2 Answers to the Research Questions

RQ1. Can a calibrated MLP detect attacks at low false-positive rates on merged, heterogeneous data? Yes. Phase 1 achieved near-perfect AUCs, and at the chosen 5% FPR operating point retained TPR ≈ 0.999. This supports deployment where low-latency screening with bounded false alarms is mandatory.

RQ2. Can we reliably discriminate attack families using the same modeling paradigm? Yes. Phase 2 attained Top-1 $\approx 95.7\%$ and macro-F1 ≈ 0.94 on the test split over eight families, with Top-3 near 99.8%. Residual errors are concentrated among semantically adjacent families, consistent with their overlapping feature distributions.

RQ3. Does cascading improve deployability without sacrificing coverage? Yes. Because Phase 1 TPR is near unity, end-to-end cascade recall closely matches Phase 2 per-class recall, while benign pass-through is primarily targeted by the Phase 1 threshold. In effect, the cascade reduces compute for benign traffic and localizes expensive analysis to likely attacks, improving scalability.

6.3 Discussion

6.3.1 What the Results Mean in Practice

The calibrated binary gate offers a tunable control over alert volume. At 5% FPR, security teams receive a bounded alert stream with minimal attack misses. The family-level classification provides triage-ready labels for downstream playbooks (e.g., `SYN/UDP_LAG` vs. `UDP` flooding respond differently). The combination yields:

- **Operational tractability:** alerts sized to team capacity by adjusting Phase 1 threshold.
- **Actionable specificity:** Phase 2 labels align with containment actions (e.g., rate-limits, ACL updates).

- **Resource efficiency:** benign flows are screened cheaply; heavy inference is reserved for suspected attacks.

6.3.2 Why MLP Works Well Here

Tabular flow features (counts, lengths, IATs, flags) are low-dimensional and structured; with scaling and modest regularization, MLPs fit the class boundaries effectively without the overhead of sequence or spatial inductive biases. Compared to CNN/RNN baselines in related work, our results confirm that (i) *calibration* and *threshold selection* matter more than deeper architectures for this setting, and (ii) label curation and rebalancing exert larger effects than complex model changes.

6.3.3 Error Patterns and Their Causes

Most misclassifications arise between semantically adjacent families (`MSSQL` vs. `DNS/LDAP/SNMP`, occasional `UDP` vs. `SYN/UDP_LAG`). Feature analysis shows overlapping ranges in packet sizes and IAT statistics under load, which explains the residual confusion. Data prevalence also plays a role: rarer families require either stronger reweighting or targeted augmentation to close the gap.

6.3.4 Robustness and Stress Testing

We probed five axes:

1. **Imbalance sensitivity:** Macro-F1 improves as minority ratio increases toward balance; careful class weighting/oversampling yields measurable gains without retraining.
2. **Augmentation noise:** Increasing timing/length jitter degrades performance gradually, suggesting the model relies on these statistics but retains resilience under moderate variance.
3. **Feature-group drops:** Removing Timing/IAT or PacketLength groups causes the largest drops, indicating these clusters carry the bulk of discriminative signal.
4. **Domain drift:** Mild-to-moderate drift reduces macro-F1 by roughly one point; Top-3 remains very high, indicating graceful degradation with operational headroom.

5. **Cascade stability:** Because Phase 1 is calibrated, end-to-end performance remains predictable under shifts, provided the threshold is revalidated periodically.

6.3.5 Compute and Memory Considerations

The following figures refer to the **training** job on a single node (8 CPU cores): wall-clock time ≈ 4:33:27, CPU utilized 06:48:46 (18.69% of 112:27:36 core-walltime), and peak memory ≈ 437.13 GB (85.38% of 512 GB). These observations inform two deployment takeaways:

- **Throughput planning (inference):** In production, Phase 1 should run at the edge or ingestion with small batches (or streaming), while Phase 2 scales horizontally for traffic flagged as suspicious.

- **Footprint control (inference):** Apply feature pruning and quantization-aware inference to reduce memory/latency without materially affecting accuracy, consistent with the feature-group sensitivity analysis.

6.4 Limitations

- **Label merging choices:** While empirically motivated, merges may conceal subfamilies relevant to incident response in some environments.

- **Dataset shift scope:** The merged corpus is broad but still bench-oriented; certain enterprise-specific behaviors (encrypted encapsulations, proprietary protocols) are underrepresented.

- **Feature modality:** We rely on flow-level features only. Payload-agnostic detection avoids privacy issues, but misses content-derived cues that might separate close families.

- **Post-training adjustments:** Some robustness analyses use inference-time weighting/perturbation rather than full retraining under the altered conditions.

6.5 Threats to Validity

6.5.1 Internal Validity

We controlled for data leakage via proper temporal/random splits and consistent preprocessing across splits. However, residual correlations across capture days or subnets could inflate performance. Calibration curves and TPR@FPR checks mitigate this risk by emphasizing operating-point stability.

6.5.2 External Validity

Generalization to new networks depends on how closely their traffic mix matches the merged corpus. The drift experiments indicate robustness to moderate shifts, but periodic threshold review and selective fine-tuning are prudent when deploying to substantially different environments.

6.6 Practical Recommendations

1. **Deploy the cascade:** Run the Phase 1 MLP with a calibrated threshold tuned to site capacity; route positives to Phase 2 for family labeling.
2. **Schedule threshold reviews:** Revalidate FPR/TPR monthly or after significant traffic changes; use prevalence-aware precision dashboards.
3. **Prioritize features:** Preserve Timing/IAT and PacketLength features in logging and pipelines; they provide the strongest signal.
4. **Target hard families:** For `MSSQL` vs. `DNS/LDAP/SNMP`, collect additional examples and consider light domain-specific augmentations.
5. **Automate drift checks:** Track macro-F1 and Top-3 on rolling windows; alert when drops exceed a small tolerance (e.g., 12 points).

6.7 Future Work

- **Online and continual learning:** Incorporate drift detectors with selective replay and lightweight fine-tuning to adapt thresholds and class boundaries over time.
- **Semi/self-supervised enrichment:** Use contrastive objectives on abundant unlabeled flows to tighten family clusters and assist rare classes.

- **Cost-aware inference:** Explore early-exit heads and confidence-based routing to further reduce latency without harming recall.
- **Richer modalities:** Integrate side-channel signals (TLS metadata, limited header semantics) under privacy constraints to disambiguate adjacent families.
- **Explainability for operations:** Provide per-alert feature attributions with calibrated confidence to speed analyst triage.

6.8 Concluding Remarks

This thesis demonstrates that a carefully engineered, calibration-aware MLP cascade can deliver strong detection and actionable family classification on a large, heterogeneous flow corpus. The approach balances *accuracy*, *operational control*, and *resource efficiency*. While there remains headroom on a few adjacent families and under stronger distributional shifts, the presented system offers a practical baseline for production IDS that can be incrementally improved with targeted data collection, drift-aware learning, and cost-sensitive inference.

References

[1] Manos Antonakakis et al. "Understanding the Mirai Botnet". In: *Proc. USENIX Security Symposium* (2017).

[2] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *IEEE Transactions on Knowledge and Data Engineering* (2002).

[3] Yin Cui et al. "Class-Balanced Loss Based on Effective Number of Samples". In: *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[4] Wesley M. Eddy. "TCP SYN Flooding Attacks and Common Mitigations (RFC 4987)". In: *IETF RFC* (2007).

[5] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: *Proc. ICML* (2017).

[6] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proc. ICML* (2015).

[7] R. Khantouchi et al. "Eye-Net: A Low-Complexity DDoS Attack-Detection System Based on MLP". In: *Sensors* (2024).

[8] Marc Kührer et al. "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks". In: *Proc. USENIX Security Symposium* (2014).

[9] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific Containers for Mobility of Compute". In: *PLOS ONE* (2017).

[10] Arash Habibi Lashkari et al. "CICFlowMeter: A Network Flow Feature Generator for Intrusion Detection". In: *Canadian Institute for Cybersecurity Technical Report* (2017).

[11] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *Proc. ICLR* (2017).

[12] Paulius Micikevicius et al. "Mixed Precision Training". In: *Proc. ICLR* (2018).

[13] Jelena Mirkovic and Peter Reiher. "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms". In: *ACM Computing Surveys* (2004).

[14] Nour Moustafa and Jill Slay. "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *Proc. Military Communications and Information Systems Conference (MilCIS)* (2015).

[15] Sajid Naseer et al. "Enhanced Network Anomaly Detection Based on Deep Neural Networks". In: *IEEE Access* (2018).

[16] Christian Rossow. "Amplification Hell: Revisiting Network Protocols for DDoS Abuse". In: *Proc. NDSS* (2014).

[17] Marco Saerens, Patrice Latinne, and Christine Decaestecker. "Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure". In: *Proc. ECML* (2002).

[18] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot is More Informative than the ROC Plot when Evaluating Binary Classifiers on Imbalanced Datasets". In: *PLOS ONE* (2015).

[19] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *Proc. International Conference on Information Systems Security and Privacy (ICISSP)* (2018).

[20] Iman Sharafaldin et al. "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy". In: *Proc. International Carnahan Conference on Security Technology (ICCST)* (2019).

[21] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *JMLR* (2014).

[22] R. Vinayakumar et al. "Deep Learning Approach for Intelligent Intrusion Detection System". In: *IEEE Access* (2019).

[23] Y. Xin et al. "Machine Learning and Deep Learning Methods for Cybersecurity". In: *IEEE Access* (2018).

[24] Chuanlong Yin et al. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks". In: *IEEE Access* (2017).

[25] Andy B. Yoo, Morris A. Jette, and Mark Grondona. "SLURM: Simple Linux Utility for Resource Management". In: *Job Scheduling Strategies for Parallel Processing* (2003).

[26] Bianca Zadrozny and Charles Elkan. "Transforming Classifier Scores into Accurate Multiclass Probability Estimates". In: *Proc. ACM SIGKDD* (2002).

[27] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *Proc. ICLR* (2018).

[28] Y. Zhang et al. "Deep-Full-Range: A Deep Learning Based Encrypted Traffic Classification and Intrusion Detection Framework". In: *IEEE Access* (2019).

A

Appendix

ʻ

Label Mapping Details

Level 1 (Family) Ontology Final Set

Table A.1 lists the consolidated families used in Phase 2. These correspond to operationally meaningful groups at the *flow-statistics* level and are the only targets predicted by the Phase 2 classifier.

Table A.1: Consolidated Level 1 families used throughout the thesis.

Family (Level 1)	**Intent / Mechanism (short)**
DNS/LDAP/SNMP	UDP reflection/amplification (53/389/161)
NET_BIOS/PORTMAP	UDP reflection (NBNS 137 / rpcbind 111)
SYN/UDP_LAG	TCP SYN backlog exhaustion / low-rate lag via UDP
UDP	Generic UDP floods (random/specific ports)
NTP	UDP reflection/amplification (123)
MSSQL	UDP reflection (SQL Browser 1434)
TFTP	UDP reflection (69)
OTHER	Attacks not reliably separable under flow statistics
BENIGN	Non-malicious traffic

Deterministic Mapping Rules (Protocol/Port Heuristics)

When a raw label does not explicitly encode its protocol family (common in heterogeneous corpora), we apply the following *deterministic* rules using transport and destination port. If multiple rules match, preference is given in the order shown.

Table A.2: Heuristic rules for mapping raw flows to Level 1 families.

Condition (transport/port)	**Map to**	**Notes**
UDP & dst_port = 53	`DNS/LDAP/SNMP`	DNS reflection/amplification
UDP & dst_port = 389	`DNS/LDAP/SNMP`	CLDAP reflection
UDP & dst_port = 161	`DNS/LDAP/SNMP`	SNMP reflection
UDP & dst_port = 137	`NET_BIOS/PORTMAP`	NetBIOS NBNS reflection
UDP & dst_port = 111	`NET_BIOS/PORTMAP`	Portmap/rpcbind reflection
UDP & dst_port = 123	`NTP`	NTP reflection
UDP & dst_port = 1434	`MSSQL`	SQL Browser reflection
UDP & dst_port = 69	`TFTP`	TFTP reflection
TCP & syn_flag_count $\gg$ others	`SYN/UDP_LAG`	Half-open SYN behavior
UDP & (rate high, small payloads)	`UDP`	Generic UDP flood
otherwise	`OTHER`	Conservative fallback

Implementation note. These rules are applied only when a confident protocol family cannot be inferred from the raw label. They are evaluated after basic hygiene (e.g., excluding identifiers from the learning matrix).

Source-Specific Raw → Level 1 Examples

The tables below illustrate how common raw labels from each source are mapped to the Level 1 families. They serve as documentation; the **complete machine-readable mapping** is exported alongside the following artifacts:

- artifacts/phase2_family_label_map.json
- artifacts/raw_to_level1_map.csv

CICDDoS-2019 (examples)

Table A.3: Examples from CICDDoS-2019.

Raw label (source)	Mapped Level 1	Rationale (port/proto)
DrDoS_DNS	DNS/LDAP/SNMP	UDP/53 reflection
DrDoS_LDAP	DNS/LDAP/SNMP	UDP/389 reflection
DrDoS_SNMP	DNS/LDAP/SNMP	UDP/161 reflection
DrDoS_NetBIOS	NET_BIOS/PORTMAP	UDP/137 reflection
DrDoS_PORTMAP	NET_BIOS/PORTMAP	UDP/111 reflection
DrDoS_NTP	NTP	UDP/123 reflection
DrDoS_MSSQL	MSSQL	UDP/1434 reflection
DrDoS_TFTP	TFTP	UDP/69 reflection
Syn	SYN/UDP_LAG	TCP SYN dominance
UDP	UDP	Generic UDP flood
UDP-Lag	SYN/UDP_LAG	Low-rate latency induction
BENIGN	BENIGN	Non-malicious traffic

CIC-IDS-2017 (examples)

Table A.4: Examples from CIC-IDS-2017 (wrapped to page width).

Raw label (source)	**Mapped Level 1**	**Rationale**
`DDoS`	`OTHER`	Mixed generators/targets; no stable flow signature at aggregate level.
`DoS Hulk / GoldenEye / Slowloris / Slowhttptest`	`OTHER`	App-layer DoS variants; heterogeneous patterns not aligned with reflection/volumetric families.
`PortScan`	`OTHER`	Recon/scan activity; outside Level 1 DDoS family scope.
`Heartbleed`	`OTHER`	Targeted protocol exploit; sparse flows, not reflective DDoS.
`Web Attack XSS / Brute Force / SQL Injection`	`OTHER`	Application-layer attacks; flow summaries differ from UDP reflector families.
`Bot / Infiltration`	`OTHER`	Multi-stage behaviors; not a single DDoS family at flow level.
`BENIGN`	`BENIGN`	Non-malicious traffic.

UNSW-NB15 (examples)

Table A.5: Examples from UNSW-NB15.

Raw label (source)	Mapped Level 1	Rationale
`DoS`	`OTHER`	Mixed DoS forms; insufficient port/proto specificity
`Generic`, `Exploits`, `Reconnaissance`	`OTHER`	Broad categories; non-DDoS semantics
`Fuzzers`, `Backdoor`, `Shellcode`, `Worms`	`OTHER`	Non-DDoS families
`BENIGN`	`BENIGN`	Non-malicious traffic

Notes on conservatism. For sources where raw labels do not uniquely imply a reflection/volumetric family (by protocol/port) or where the flow signatures are heterogeneous, labels are mapped to `OTHER`. This prevents *shortcut learning* and avoids artificially inflating family-level accuracy.

Provenance and Reproducibility

The full mapping is stored with the thesis artifacts for deterministic reuse:

- `artifacts/phase2_family_label_map.json` canonical Level 1 index ↔ name map.
- `artifacts/raw_to_level1_map.csv` two-column CSV `raw_label,level1_family` across all sources.

Both files are generated by the preprocessing scripts (Phase 2 data prep). At inference time, only Level 1 families in Table A.1 are emitted; any unseen raw label from new domains must first be resolved via the rules in Section A or routed to `OTHER`.