Multi-Level Deep Generative Model with Poisson Variational Autoencoders and Reinforcement Learning for Enhanced Intrusion Detection System

by

Riddha Majumder
23341093
Md.Tanzim Qaiyum
24241049
Fathin Ishrak
20301027
Mehrin Amin Neha
21201070

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
October 2025.

© 2025. Brac University
All rights reserved.

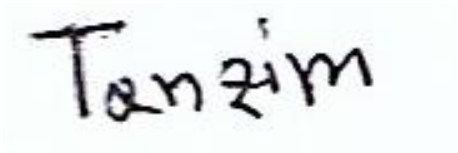
Declaration

It is hereby declared that

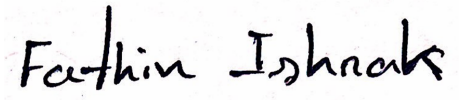
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Riddha Majumder
23341093



Md. Tanzim Qaiyum
24241049



Fathin Ishrak
20301027

Mehrin Amin Neha
21201070

Approval

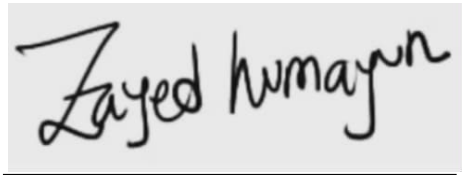
The thesis/project titled "Multi-Level Deep Generative Model with Poisson Variational Autoencoders and Reinforcement Learning for Enhanced Intrusion Detection System." submitted by

1. Riddha Majumder (23341093)
2. Md. Tanzim Qaiyum (24241049)
3. Fathin Ishrak (20301027)
4. Mehrin Amin Neha (21201070)

of Summer, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in October 10, 2025 Year.

Examining Committee:

Supervisor:
(Member)



Zayed Humayun

Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Network intrusion detection systems face two critical challenges: the need to identify specific attack types beyond binary classification and the ability to adapt to evolving threats while maintaining low false positive rates. In response, we propose a hierarchical multi-level Poisson Variational Autoencoder (PVAE) system augmented with reinforcement learning-based ensemble weighting for intrusion detection. At its core, a three-level PVAE chain with normalizing flows captures network traffic patterns at packet, flow and session levels. In our research, we applied class-conditional radial recalibration fitted on validation data to align the latent space separately for each attack class. Rather than training multiple separate models, we create ensemble diversity by instantiating three architectural variants from a single trained checkpoints. A Proximal Policy Optimization (PPO) agent then learns dynamic per-sample weights based on prediction confidence and other patterns. Finally, an XGBoost meta-learner refines the PPO-weighted ensemble outputs using uncertainty diagnostics to produce the final classification. Evaluated on the NF-UQ-NIDS-v2 dataset with 14 traffic classes, our system achieves 89.17% multiclass accuracy with 88.01% weighted F1-score, while maintaining strong binary discrimination with 98.11% AUROC and 99.11% PR-AUC for normal versus attack detection.

Keywords: IDS, Generative model, Poisson Variational Autoencoder, Proximal Policy Optimization

Acknowledgement

All praise is due to Allah, whose blessings allowed us to complete this thesis without facing any significant obstacles. We are profoundly thankful to Zayed Humayun Sir, our advisor, for his thoughtful advice and steady mentorship whenever we needed it. Again, our deepest thanks go to our parents for their constant support and prayers, which finally have brought us so close to graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Modern cyber threats have evolved into sophisticated, large-scale attacks that challenge traditional security mechanisms. Botnets, which combine "robot" and "network," exemplify this evolution. A botnet refers to a network of compromised computers or Internet of Things (IoT) devices infected with malware and controlled by a botmaster or bot herder [12]. These networks function as integrated forces for cyber-attacks, enabling fake websites, Distributed Denial of Service (DDoS) attacks, spamming and malware distribution [17]. The rapid growth of IoT devices, which surpassed twenty billion in 2020 [18], has amplified the threat landscape. The European Union Agency for Network and Information Security ranked botnets as the seventh major global cyber threat in 2018 [16], underscoring the urgency for adaptive defense mechanisms.Traditional signature-based intrusion detection systems (IDS) rely on predefined attack patterns to identify known threats. While effective against familiar attacks, these approaches struggle with novel and zero-day exploits [1]. In response, IDS have evolved toward machine learning (ML)-powered solutions that offer greater flexibility through data-driven models capable of anomaly detection and traffic classification using supervised, unsupervised, or semi-supervised methods [22]. However, the choice of model architecture introduces critical trade-offs. Discriminative models such as decision trees, Random Forest (RF), and XGBoost (XGB) excel at defining class boundaries and often achieve high accuracy [10]. Yet, these models can be overconfident in their predictions, leading to calibration issues where predicted probabilities fail to reflect actual likelihoods [13]. Such miscalibration increases false alarm rates, wasting resources and contributing to alert fatigue among security analysts.

Generative models, including Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) [8] and Variational Autoencoders (VAEs) (Kingma & Welling, 2013) [4], offer complementary advantages by capturing underlying data distributions and quantifying uncertainty. This is particularly valuable for handling noisy and imbalanced network data. Normalizing Flows (NF), introduced by Rezende and Mohamed (2015) [9], extend this capability with tractable density estimation and have been applied across domains including image generation, video synthesis, and reinforcement learning. By integrating generative modeling with Deep Reinforcement Learning (DRL), systems can adaptively learn decision policies that balance de-

tection accuracy with operational costs.Modern network traffic exhibits multi-scale characteristics, ranging from individual packets to aggregated flows and sessions. These distributions are inherently count-based, heavy-tailed, and bursty, reflecting dynamic user behavior and diverse application protocols [23]. Simplistic feature engineering often fails to capture bursty patterns indicative of advanced attacks such as DDoS and reconnaissance, necessitating hierarchical modeling to represent dependencies across scales [3]. The NF-UQ-NIDS-v2 dataset addresses these challenges by providing a comprehensive, multi-class collection of NetFlow records encompassing benign traffic and labeled attacks including DoS, DDoS, reconnaissance, and malware [23]. The dataset's inherent class imbalance and heavy-tailed feature distributions mirror real-world operational conditions. Through careful preprocessing involving sampling, normalization, and multi-level feature grouping, researchers can align models with the statistical properties of network traffic, improving both detection performance and calibration quality.This research proposes a novel intrusion detection system that merges hierarchical generative modeling through multi-level Poisson Variational Autoencoders (PVAE) with Normalizing Flows and Deep Reinforcement Learning (DRL) for adaptive ensemble weighting. By addressing calibration, uncertainty quantification, and multi-scale feature representation, the proposed approach aims to bridge the gap between high detection accuracy and operational reliability in dynamic threat environments.

1.2 Research Motivation

Botnet identification remains one of the most difficult tasks in modern cybersecurity due to the subtle and adaptable nature of assaults. The ability of traditional intrusion detection systems (IDS) to capture the larger behavioral context of malicious activity has been limited by their frequent reliance on packet-level or flow-level information in isolation [7], [5]. To identify both long-term behavioral patterns and micro-level structures, modern intrusion detection necessitates examining traffic at multiple granularities, including packets, flows, and sessions. Session-level aggregate displays higher-order behaviors across time, flow-level features record throughput and directional characteristics, and packet-level features contain fine-grained information such as header values and burst patterns. Previous research has shown that, in comparison to single-level detection techniques, multi-granular analysis offers richer information and dramatically lowers false positives [26],[33]. Even a slight increase in the decrease of false positives can have a significant operational impact on real-world Security Operations Centers (SOCs), where analysts must go through enormous volumes of alerts [16], [32].

By integrating Normalizing Flows (NFs) and Poisson Variational Autoencoders (PVAEs) across all three traffic abstraction levels, our method expands on this understanding. Because generative modeling techniques directly represent the data distribution rather than depending just on discriminative characteristics, they are becoming more and more popular in intrusion detection [9],[25]. Particularly, normalizing flows offer strong density estimation capabilities and adaptable latent representations, which let the system identify sophisticated attack patterns that conventional classifiers are unable to detect [6]. This makes it easier for detection algorithms to differentiate between actual malicious activity and benign traffic fluc-

tuations, which is essential for identifying botnets that are stealthy and constantly changing.

The requirement for calibrated uncertainty in IDS projections is equally important. Even on hostile or out-of-distribution data, a large number of machine learning intrusion detection algorithms provide overconfident predictions [19], [47]. For SOC analysts, this kind of conduct erodes confidence and makes triage harder. The use of post-hoc calibrators and latent recalibration modules in this model guarantees accurate and comprehensible uncertainty estimations. This strategy is in line with previous requests in the literature for uncertainty-aware IDS that can measure prediction accuracy while preserving excellent classification capabilities [46],[20]. When combined, calibrated uncertainty, generative latent learning, and multi-granular modeling offer a basis for developing workable IDS systems that satisfy operational SOC criteria for low false positive tolerance, accuracy, and interpretability.

1.3 Research Problem

Three major obstacles must be overcome in order to identify network attacks: explainability of alarms, low false positive rates (FPR), and accurate multi-class categorization. Multi-scale temporal context is not integrated into traditional intrusion detection systems, which frequently depend on either packet-level or flow-level characteristics. Inadequate generalization of complex botnet activities across various traffic levels results from this particular issue [33], [32]. Furthermore, models are biased toward majority classes in imbalanced datasets, when benign traffic greatly outnumbers malicious flows. This leads to inconsistent performance for minority attack types [29],[36].

The absence of a reliable uncertainty estimate is yet another enduring problem. Many current systems are overconfident in misclassifications because they produce deterministic predictions without calibrated confidence [19],[47]. In high-stakes situations like SOCs, where analysts want actionable insights supported by trustworthy confidence scores, this issue is made worse. Operators are unable to differentiate between known attack types, benign anomalies, and truly unexpected threats in the absence of calibrated uncertainty [46],[20]. Lastly, many deep learning-based IDS still lack enough explainability. To trust and operationalize alerts, analysts need clear justifications for occurrences that are detected, including both feature-based explanations and interpretable rules [31].

In order to fill these gaps, the model uses hierarchical PVAEs and conditional Normalizing Flows to model network traffic at the packet, flow, and session levels; introduces class-conditional recalibrators to align latent spaces and guarantee meaningful uncertainty; employs adaptive sampling techniques to rectify class imbalance; and combines generative features with XGBoost classifiers and rule-based heuristics for hybrid decision-making. With calibrated uncertainty and comprehensible justifications appropriate for practical deployment in SOCs, this approach specifically aims for accurate multi-class attack detection at low FPR.

1.4 Objective

This work aims to develop, deploy and assess an adaptive hierarchical intrusion detection system that combines explainability, uncertainty calibration, and generative modeling for reliable botnet detection. The particular goals are:

1. Design a multi-level generative model that learns network intrusion patterns hierarchically by processing features at three distinct granularities: packet-level details (sizes, flags), flow-level statistics (byte counts, durations), and session-level behaviors (throughput, directional patterns). Each level uses a Poisson Variational Autoencoder (PVAE) to capture the count-based nature of network traffic.

2. Enhance latent space expressiveness by integrating advanced normalizing flows within each PVAE level, enabling the model to capture complex, multimodal distributions of both benign and attack traffic that simple Gaussian assumptions cannot represent

3. Connect hierarchical representations across levels through Bridge Flows. These conditional normalizing flow networks learn to align the latent spaces from packet, flow, and session levels, ensuring consistency and enabling information flow across different feature abstractions.

4. Stabilize training dynamics by applying PFGM-inspired regularization constraints that prevent latent space collapse, control volume changes, and discourage excessive radial expansion, ensuring the model produces reliable and well-calibrated latent representations throughout training.

5. Implement class-conditional latent recalibration using a novel quantile-based radial calibrator that adjusts the latent space geometry separately for each attack class, aligning learned distributions with theoretical expectations and improving discrimination between similar attack types.

6. Create ensemble diversity from a single trained model by instantiating three architectural variants from the same checkpoint: one with full recalibration and flows enabled, one with recalibration disabled, and one with all flows removed. This approach generates complementary decision boundaries without requiring separate training runs.

7. Learn dynamic per-sample ensemble weights using Proximal Policy Optimization (PPO), a reinforcement learning agent that observes prediction confidence, cross-base disagreement, flow densities, and sparsity patterns, then assigns optimal weights to each base learner while optimizing for cost-sensitive rewards that heavily penalize false positives on benign traffic.

8. Refine final predictions through an XGBoost meta-learner that combines the PPO-weighted ensemble outputs with uncertainty diagnostics, providing robust multiclass predictions with strong discrimination across all 14 traffic categories while maintaining low false alarm rates.

9. Ensure interpretability and operational trust by providing gradient-based attribution that traces predictions back to influential input features and ranking latent channel importance, enabling security analysts to understand why the system flagged specific traffic as malicious.

10. Evaluate system performance using multiclass accuracy, per-class F1 scores, macro/micro AUROC, binary PR-AUC for normal-vs-attack discrimination with particular emphasis on cost-sensitive metrics that reflect real-world deployment constraints.

1.5 Methodology in Brief

We proposed a hybrid intrusion detection system that learns patterns first, then decides smartly. Traffic features are grouped into three levels, packet, flow and session and fed to a chained Poisson VAE (PVAE). Each level has normalizing flows and we also apply a simple class-wise recalibration to tidy the latent space. From the same trained PVAE, we create three variants to get diverse views of the data. Therefore, base A keeps normalizing flows and class-wise recalibration on, giving the most expressive model. Base B keeps the flows but turns off recalibration, changing the latent geometry for a different perspective. Again, base C turns off the flows, a simpler and often more stable baseline. A PPO agent learns per-sample weights to blend these generative models. Finally, an XGBoost meta-learner stacks the weighted outputs with uncertainty cues to make the call. We use NF-UQ-NIDS-v2 dataset and we do standardize, impute and scale features before training.

1.6 Scopes and Challenges

The scope of our work was limited to the NF-UQ-NIDS-v2 dataset, which contains diverse attack types and realistic traffic patterns. Our model was trained and evaluated only on this dataset, so real-time deployment and generalization to unseen network environments remain as future work. Due to hardware constraints and the dataset's large size, we used a sampled subset (1.48M records) rather than the full 75M records to ensure feasible training. Some rare classes such as Infiltration and Analysis had limited samples, making them challenging to classify accurately. Moreover, our proposed architecture is computationally intensive. But it was essential to achieve the desired level of generative modeling accuracy. In our work, our group collectively handled data preprocessing, model design, and experimentation. Our role focused on developing and training the hierarchical PVAE chain, implementing the normalizing flow and radial recalibration mechanisms. After that, we integrated the PPO and XGBoost ensemble for final classification.

Chapter 2

Literature Review

2.1 Review of Existing Research

The growing number of Internet of Things (IoT) devices has resulted in a parallel rise in security concerns, particularly from malicious network traffic. Recent studies have focused on utilizing methods from machine learning to improve botnet detection systems [12]. Using BOT-GAN to detect botnets is one of the novel approaches. BOT-GAN is a generative adversarial network based framework, which is based on deep learning architecture. Yin et al. (2018) [15] have mentioned, the increasing and alarming rate of botnets, which have evolved to overcome traditional intrusion detection methods which mainly relies on packet inspection, means inspecting each and every data being sent over any network. So to improve these IDS's, using BOT-GAN could be an effective approach, as it can significantly enhance the detection accuracy making it a practical solution. However, this approach heavily relies on the diversity of the training data set. If the dataset used for training does not represent the variety of intruder's behavior, then the model may struggle to generalize a practical solution [15].

To prevent the struggle of detection systems with accuracy, Hernadi (2023) have decided to use the modern CICIDS2017 dataset [14] which provides a variety of modern traffic and alert types with the application of two generative models, Variational Autoencoders (VAEs) and Wasserstein Generative Adversarial Networks (WGANs). This method effectively generates synthetic data for attack classes like botnet [38], [39], [44]. Also as the dataset is very latest and has current network traffic patterns, it makes the results more relevant and applicable for real world scenarios. However, this approach provides an improvement in the detection system, it also has some certain limitations. The dataset might be modern, but it doesn't encompass all possible attack scenarios in different network conditions, potentially limiting the results [44]. Similarly, Amokrane et al. (2024)[49] highlighted that selecting the right features from the dataset plays a key role in improving IDS accuracy while reducing unnecessary computation. Singh et al. (2021) also showed that good feature selection greatly improves model scalability and accuracy, especially in resource-limited IoT and mobile intrusion detection systems. Moreover, another novel [47] approach by Randhawa et al. (2023) is "RELEVAGAN" which is created by using generative adversarial networks (GANs) and also with the help of deep reinforcement learning (DRL). In a typical GAN, it consists of two neural networks called generator which

creates adversarial attacks and discriminator which works as a identifier in identifying real attack. Specifically in this model, the DRL attacks the discriminator of the GAN to verify that the adversarial examples are semantic aware. Moreover, the model that is used in this approach, "RELEVAGAN" [47], performs better than Auxiliary Classifier GAN (ACGAN) and Evasion Generative Adversarial Network (EVAGAN) [34] models. However, the training time takes higher in this approach, but still it's a novel model which might provide an optimized output in future research.

In another approach, Shahid et al.(2021) [24] combine an autoencoder with Generative Adversarial Network (GAN) to generate network traffic resembling the actual behavior of Internet of Things (IoT) devices. They have used the malicious datasets from IoTPOT, which is an IoT honeypot designed for malware research. The authors of this research reported that their trained detectors achieved a true positive rate of 95.04 % and a false positive of 2.17 %. The combination of Autoencoders with a GAN to generate flows of IoT network traffic is a novel approach that improves the traditional traffic generation methods. After test and run, the authors have demonstrated that the generated traffic can successfully mimic legitimate behavior showing potential implications for both security and malicious use. However, this study heavily relies on the data generated from a single type of IoT device which is Google Home Mini, which may not represent the diversity of traffic from various IoT devices with different behaviors. So for this reason, the performance of the generative model may not generalize other types of IoT devices or other network environments like other studies [51], [27], potentially limiting its application. Furthermore, this study might not hold against advanced or hybrid detection methods in real world scenarios as this study mainly focuses on some selected algorithms [24]. Another research [28] presents an approach to enhance the security of a machine learning based detection system against evasion attacks, called Bot-shot. It is a proactive learning technique for adversarial evasion, which is similar to other methods used in different papers [40], [47]. They have also used an extended feature set for botnet data generation, addressing data imbalance issues in botnet datasets to improve the performance.

Discussing data imbalance, Ullah et al.(2021) [31] addressed the challenge of anomaly detection due to this data imbalance. This also hampers the traditional methods of detection. To prevent this problem, authors provided a solution using conditional generative adversarial networks (CGAN) to produce synthetic data that keeps balance in the dataset. The authors showed that their proposed model significantly improved the performance by achieving a deviation of 2 % to 4 % in classification tasks. Moreover, the findings by these authors also indicate that use of CGANs not only addresses the data imbalance but also enhances the overall performance of anomaly detection. Similar to the previous papers, the limitation of the data generated by GANs may not fully capture the behaviors of real world anomalies. Moreover, the performance of the proposed models (ocGAN & bcGAN) in highly dynamic scenarios need to be rechecked [31]. Another study [27] uses an adversarial autoencoder model with Bidirectional generative adversarial networks (BiGAN) to detect the unknowns. The have used the IoT-23 dataset is a new dataset based on network traffic from Internet of Things (IoT) devices. Additionally, k-nearest neigh-

bor algorithm (KNN) was used as a classifier here with AAE and BiGAN. Both of the models showed a close similar result and were able to detect DDoS attacks. However, BiGAN was more effective in detecting anomalies. Shinan et al.(2021) [30] structured their survey on botnet detection approaches, focusing on machine learning (ML) techniques in software-defined networks (SDNs) . They investigated around four main areas which are general botnet detection techniques, ML-based detection methods, detection in SDNs and ML applications in SDN environments. Their approach involved a thorough review of 96 relevant articles, where they extracted data on various detection techniques, protocols and performance metrics. However, they just summarized existing research findings and do not provide detailed information on the specific datasets utilized in their analysis. Research by Ibrahim et al.(2021) [26] Proposes an innovative detection structure that uses behavior-based analysis to identify botnets. The framework contains two layers, the first layer filters regular traffic to reduce processing time and the second layer conducts an analysis of suspicious traffic. The authors demonstrate that their approach achieves over 90 % accuracy with a false-negative rate below 2.5 % while highlighting the success of the k-nearest neighbors (KNN) algorithm, which performed better without oversampling.

Another research by Wu et al.(2019) [19] presents deep reinforcement learning (DRL) for adversarial botnet detection. They have shown an advanced approach, where the model simulates black box attacks. The adversarial samples used in this model maintain original botnet functionality, ensuring realistic evasion. However, the study focuses on two models, decision tree and convolutional neural network (CNN) which refuses the generalizability to the architecture. However, Al-Fawa'reh et al.(2023) [43] introduces MalBoT-DRL, a malware botnet detection framework using Deep Reinforcement Learning (DRL) to provide safety to IoT networks. This system addresses challenges like limited generalizability shown in [19] by dynamically adapting to evolving malware patterns. Utilizing two datasets (MedBIoT and N-BaIoT), MalBoT-DRL achieves exceptional detection rates of 99.8 % and 99.4 % while maintaining low computational overhead. Yet, this approach still Struggles with detecting certain malware during early lifecycle phases and performance in large-scale real-world IoT environments is not discussed fully in this approach. Another study by Apruzzese et al.(2020) [21] provides a framework for botnet detection, with the help of DRL. In this paper, they mentioned that this enhancement against the evasion technique can be achieved in 3 stages, preparation, where the DRL is trained to generate attack samples and then attack where such samples are used in evaluating machine learning based detectors and finally hardening, where these generated samples form part of the training set to make the detector more robust. Their approach was also tested in diverse datasets, the CTU [7] dataset, which contains collections of network flows from malicious devices and the BOTNET [5] dataset which also features similar characteristics. These datasets provide advantages as it becomes suitable for various attack scenarios. However, drawbacks include high computation requirements during DRL training.

Vu et al. (2022) [41], proposed another two deep generative models named conditional denoising adversarial autoencoder (CDAAE) and CDAAE-KNN. Furthermore, in this research, the authors conducted the experiment in 6 well known

datasets, which is a very good approach to get an idea of different attacks and also which includes various botnet scenarios. Among those 6 datasets, one was a cloud IDS dataset, two network IDS dataset and three malware datasets. The experimental results show that the proposed generative models improve the detecting accuracy over other traditional methods and prior generative approaches like ACGAN and CVAE. This study suggests the most practical adaptability and effectiveness in real world scenarios. The models used in this study are designed to produce samples that better align with the original data, which addresses the key limitations of traditional methods that often create unrealistic samples. The versatility of the authors are visible in this study as they have evaluated their models on multiple datasets, representing the robustness. By generating this kind of malicious samples, the model helps in improving the detection rates [41].

2.2 Summary of Key Findings

Looking at the studies discussed in this review, we can see that researchers between 2018 and 2025 have made considerable progress in improving intrusion detection systems through feature selection and generative deep learning techniques [49], [42], [39], [47], [24], [14]. The research deals with persistent issues like data imbalance, evasion attacks, and generalization across different network environments, with the goal of cutting down false positives and catching rare attacks more effectively. Most studies use well-known datasets including NF-UQ-NIDS-v2, CICIDS2017, NSL-KDD, UNSW-NB15, BoT-IoT, and IoTPOT, which cover a wide range of scenarios from traditional IT networks to Internet of Things environments with imbalanced classes and high-dimensional features. The methodologies vary across convolutional and recurrent networks (1D-CNN, LSTM), deep generative models (Wasserstein GAN, Deep Reinforcement Learning GAN), tree-based ensembles (Random Forest, XGBoost, ExtraTrees), and hybrid strategies that combine deep learning features with classical classifiers. Feature selection techniques like correlation-based filtering and recursive feature elimination are commonly used to reduce dimensionality and improve modeling speed. The experimental setups extract features at different levels—packet-level (packet size, counts), flow-level (inter-arrival times, byte counts), and session-based (port numbers, TCP flags)—and evaluate performance using metrics like Accuracy, F1-score, AUROC, PR-AUC, False Positive Rate (FPR), and True Positive Rate (TPR), with special attention to low-FPR values needed for real-world deployment. However, several challenges remain persistent: class imbalance causes models to perform poorly on minority attack types like reconnaissance and infiltration, label noise in merged datasets reduces accuracy, and overfitting to specific datasets limits generalization across different network domains, requiring careful data preprocessing and model calibration [39], [14].

The performance trends show that hybrid models combining generative features with tree-based classifiers outperform traditional machine learning by 5–10% in AUROC and F1-score on imbalanced datasets like NSL-KDD and UNSW-NB15, achieving TPR of 95%–98% at FPR around 1%, with feature selection capable of cutting false alarm rates in half [49], [39]. In binary classification, these hybrid systems reach up to 96% accuracy on NF-UQ-NIDS and CICIDS2017 datasets, especially when generative model outputs are well-calibrated to avoid mode collapse [49], [47]. In

multiclass scenarios, common attack types like DoS and DDoS achieve F1-scores above 0.95, while rare classes like malware and reconnaissance often score below 0.90 due to feature overlap and limited training samples [42], [14]. Studies emphasize that accuracy alone is insufficient for evaluating IDS performance, and maintaining low FPR is critical for practical deployment, with FPR/TPR and PR-AUC providing more reliable assessments than accuracy [50], [42], [14]. There is some divergence in how researchers view generative models like VAEs and flow-based models, with performance differences often attributed to preprocessing strategies, calibration methods, and threshold selection approaches, where correlation-based feature removal helps hybrid models but can hurt standalone GANs [39], [47], [24]. Despite these advances, existing studies have limitations including narrow focus on datasets like NSL-KDD, lack of open-source code for implementations like RELEVAGAN or DRL-GAN, missing details on calibration and threshold selection, and limited analysis of evasion attacks, which reduces practical applicability. Our proposed research aims to address these gaps by developing a DRL-augmented hybrid GAN model with recursive feature reduction to achieve FPR below 0.5% and TPR above 95% on NF-UQ-NIDS-v2 dataset, while improving explainability through per-class visualizations for easier real-world deployment. The key takeaway is that hybrid generative-tree models with class-wise threshold adjustment deliver the best IDS performance, but opportunities remain in exploring multi-granular analysis (flow-level and packet-level) and rule-based augmentation, which form the core of our proposed solution.

Chapter 3

Proposed Methodology

3.1 Methodology Overview

The approach used in this study uses deep generative models in conjunction with reinforcement learning and ensemble methods to improve network intrusion detection. The system combines a multi-level Poisson Variational Autoencoder (PVAE) chain with normalizing flows and class-conditional radial recalibration for unsupervised feature learning. Rather than using traditional discriminative classifiers, our study consists of three architectural variants of the same trained PVAE chain, each providing different latent geometries through selective toggling. These generative base learners are dynamically weighted using a Proximal Policy Optimization (PPO) agent, with final predictions refined through an XGBoost meta-learner. Our technique is based on the NF-UQ-NIDS-v2 dataset, which includes both benign and malicious traffic with various attack methods including reconnaissance, malware and DoS attacks. During preprocessing, the pipeline handled numeric values carefully, replacing infinities and NaNs with training set median values, then applying Min-Max normalization. For feature engineering, a layered approach was used: Level 1 focused on packet-level details like size distributions and TCP flags, Level 2 captured flow-level metrics such as packet counts and durations and Level 3 covered session-level behavior including throughput rates and directional patterns. Poisson scaling parameters were calculated separately for each level.The modeling stage employed a custom three-level PVAE chain architecture processing different granularities of network features. Each PVAE level uses an encoder with categorical embeddings and advanced normalizing flows including Planar Flows, Affine Coupling layers and Masked Autoregressive Flows (MAF). A class-conditional LRQuantileRadialCalibrator applies per-class radial mappings fitted on validation data to align latent distributions with theoretical expectations. Bridge Flows connect representations across PVAE levels, while Poisson decoders capture count-based network statistics. During training, PFGM-inspired regularization controls latent space properties and proxy classification heads handle multiclass and binary tasks. After training, three generative base learners are instantiated from the same checkpoint: Base A with full recalibration and flows, Base B with recalibration disabled and Base C without normalizing flows. This architectural diversity creates complementary decision surfaces. A small TinyWarp MLP applies post-hoc residual corrections to the latent representations for improved separability. The ensemble stage employs a PPO agent that learns dynamic per-sample weights over the three PVAE variants. The

PPO environment incorporates per-base probabilities, cross-base variance, predictive entropy, flow densities and sparsity metrics. The reward function optimizes for task performance, cost-sensitive penalties, class-aware false negative costs, calibration quality, sparsity preferences, out-of-distribution density penalties and weight entropy regularization. PPO-learned weights are blended with uniform weights for stability. After that, an XGBoost meta-learner refines the ensemble predictions using weighted probabilities, PPO weight allocations and uncertainty diagnostics. Inverse-frequency weighting with extra emphasis on benign samples handles class imbalance. Performance evaluation includes accuracy, log loss, per-class and binary AUROC, precision-recall analysis and FPR at fixed TPR thresholds. Interpretability is achieved through XGBoost-based latent importance ranking and gradient-based probing to identify influential input features. This architecture demonstrates the effectiveness of hierarchical generative models integrated with reinforcement learning for adaptive ensemble weighting, creating a robust system for real-world network security environments with dynamic threat landscapes. As depicted in Figure 3.1.

3.2 Design Specification

The proposed system builds on the concept of Poisson Variational Autoencoders (PVAEs) but extends it significantly by introducing a multi-level hierarchical architecture that processes network traffic at three distinct granularities. This design takes inspiration from the multi-level intrusion detection work of Al-Nashif et al.[2], who showed that analyzing traffic at multiple levels like flow, packet header and payload improves detection rates and reduces false alarms. In our approach, we organize network features into three hierarchical levels: Level 1 captures packet-level details like packet size distributions and TCP flags, Level 2 captures flow-level statistics like byte counts and durations and Level 3 captures session-level behaviors like throughput rates and directional patterns. Each level has its own PVAE that learns a latent representation suited to that granularity and these representations are connected through Bridge Flows, which are conditional normalizing flow networks that align the latent spaces across levels. This multi-level design allows the system to capture patterns at different scales simultaneously, similar to how a security analyst examines both individual packet details and overall session behavior when investigating suspicious activity.

A Variational Autoencoder (VAE) is a probabilistic generative model that learns to encode high-dimensional data into a lower-dimensional latent space [48]. Traditional VAEs assume the data follows Gaussian distributions, but network traffic features like packet counts, byte counts and connection numbers are inherently count-based, not continuous. The Poisson VAE addresses this by replacing Gaussian assumptions with Poisson likelihoods, which naturally model integer-valued counts where the mean approximately equals the variance [48]. This makes PVAEs especially well-suited for network traffic analysis, where many characteristics are discrete counts rather than continuous measurements. As noted in foundational work on Poisson VAEs, using a Poisson-based model avoids modeling problems with negatives or fractional outputs that Gaussian VAEs would produce, since network events are fundamentally counting processes. A difference between pvae and vae is shown in

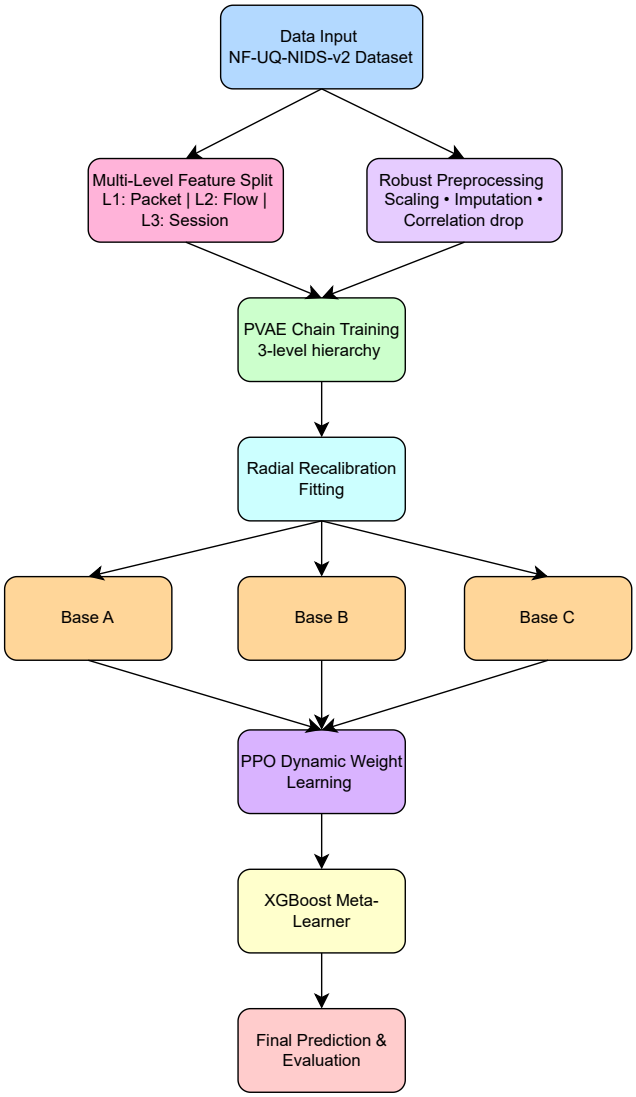


Figure 3.1: Proposed system workflow

Table 3.1. In our implementation, each of the three PVAE levels is augmented with normalizing flows specifically, Planar Flows, Affine Coupling layers and Masked Autoregressive Flows (MAF) which allow the encoder to capture complex, non-linear relationships in the latent space. These flow transformations are invertible and gradually reshape the latent distribution to better fit the data, creating more flexible and detailed representations than standard VAEs. Additionally, we introduce a class-conditional radial recalibrator that adjusts the latent space geometry separately for each attack class after training, using quantile-based piecewise-linear mappings fitted on validation data. This post-hoc calibration aligns the learned latent distributions with theoretical expectations and improves discrimination between similar attack types.

The dataset used in this research is NF-UQ-NIDS-v2, a large-scale network intrusion detection dataset created by merging multiple smaller datasets to provide

VAE	**PVAE**
Input: Continuous features Encoder: Neural network $\rightarrow z \sim \mathcal{N}(\mu, \sigma^2)$	Input: Count-based features Encoder: Neural network $\rightarrow z \sim$ Poisson(λ)
Assumes Gaussian latent space	Assumes Poisson latent space
Output: Real-valued reconstructions	Output: Count-valued reconstructions

Table 3.1: Comparison between VAE and PVAE architectures

flows from various network setups and attack scenarios [35]. It contains 75,987,976 records total, with 33.12% benign flows and 66.88% attacks across multiple categories. Attack labels have been consolidated into parent categories: DoS variants (Hulk, SlowHTTPTest, GoldenEye, Slowloris) are unified as DoS, DDoS variants (LOIC-UDP, HOIC, LOIC-HTTP) as DDoS, brute-force types (FTP, SSH, Web, XSS) as Brute Force and SQL Injection merged into Injection [35]. This consolidation creates meaningful attack categories while maintaining realistic class imbalance. For our experiments, we sampled 500,000 benign records and up to 100,000 per attack type (keeping all available samples for rare classes like Theft with 2,431 samples, Generic with 16,560, and mitm with 7,723), resulting in approximately 1.48 million samples across 14 distinct classes. The preprocessing pipeline organized 40 numeric features into the three hierarchical levels, with correlation-based reduction applied separately per level to remove redundancy while preserving level-specific information.

Rather than training multiple separate models, the system creates ensemble diversity by instantiating three architectural variants from a single trained PVAE chain checkpoint. Base A keeps all normalizing flows and recalibration enabled, Base B disables the class-conditional recalibrator to produce alternative latent geometries and Base C removes all normalizing flow components to provide a simpler VAE perspective. This approach is computationally efficient because it reuses learned weights while generating three complementary decision surfaces that see the data differently. The key innovation comes in how these bases are combined, instead of using fixed ensemble weights or training the bases separately, we employ Proximal Policy Optimization (PPO), a reinforcement learning algorithm to learn dynamic per-sample weights over the three generative variants. This design draws inspiration from recent work by Akshaya Suresh et al.[52], who showed that PPO agents can dynamically reweight base classifiers (Random Forest and CatBoost in their case) based on validation feedback, adapting to which model performs better on different traffic profiles. In our implementation, the PPO agent observes a rich state representation including per-base probability distributions, cross-base variance, predictive entropy, flow log-densities from the Bridge networks and Poisson sparsity metrics (median rates and spike fractions). The reward function is multi-objective, it maximizes task performance gain relative to a baseline model, applies cost-sensitive penalties where benign false positives are weighted 12× higher than attack false negatives (reflecting real-world operational priorities). Moreover, it includes class-aware false negative costs derived from validation recall to rescue weak

classes, encourages calibration quality, prefers sparsity in learned representations. It also penalizes out-of-distribution density on uncertain samples, and regularizes weight entropy to prevent collapse onto a single expert. The PPO agent is trained for 120,000 timesteps and its learned weights are blended 70/30 with uniform weights for stability.

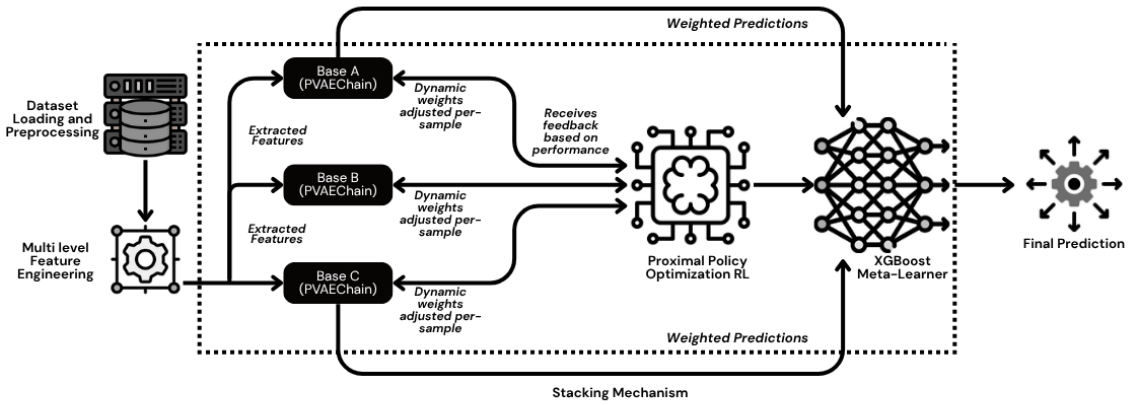


Figure 3.2: Architecture of our proposed model

After the PPO agent produces dynamically weighted predictions from the three generative bases, an XGBoost meta-learner refines the final output. Unlike the previous approach where XGBoost operated directly on PVAE latents, here it functions as a stacking meta-learner that receives an enriched feature space which are the PPO-weighted ensemble probabilities, the PPO weight allocations themselves, cross-base variance and mean density and entropy diagnostics. XGBoost is particularly well-suited for this role because of its ability to handle imbalanced data, capture complex feature interactions through gradient boosting and provide probability scores for threshold tuning [11]. To address class imbalance, inverse-frequency weighting normalized to unit mean is applied during training, with benign samples receiving 1.3× extra weight to further penalize false positives, a practical consideration for deployment where false alarms erode trust in security systems. This final stage allows the system to learn non-linear interactions between the PPO-weighted signals and uncertainty metrics, producing robust predictions that leverage both the generative model's ability to capture complex distributions and XGBoost's discriminative power for classification. The overall architecture demonstrates that hierarchical generative models, when combined with reinforcement learning-based adaptive weighting and gradient-boosted meta-learning, can handle fine-grained multiclass intrusion detection while maintaining interpretability through gradient-based feature attribution and latent importance ranking.

3.3 Data Collection

3.3.1 Dataset Source and Description

The NF-UQ-NIDS-v2 dataset, developed by the University of Queensland, was used in this research. It is a large-scale network intrusion detection dataset containing

both benign traffic and multiple types of attack patterns [35]. It contains 75,987,976 records total, with 25,165,295 (33.12%) benign flows and 50,822,681 (66.88%) attacks across multiple categories. There are 21 total classes with 1 benign and 20 attack types. There are different attack categories, such as DoS, DDoS, Reconnaissance, Bot, Brute Force, Password, Injection, XSS, Malware, Generic, Infiltration, Theft, MITM as depicted in Table 3.2.

Class	**Count**	**Description**
Benign	25,165,295	Regular, legitimate network flows (non-attacking traffic).
DDoS	21,748,351	Distributed Denial of Service: exhausting a system's resources by recruiting many distributed sources.
DoS	17,875,585	Denial of Service: exhausting resources from a single or limited source to block access to services/data.
Scanning	3,781,419	Sequential probing of a network to find open ports or services and check for vulnerabilities.
Reconnaissance	2,633,778	Information gathering about hosts or networks via probing/scanning prior to an attack.
XSS	2,455,020	Cross-Site Scripting: web exploit that runs malicious scripts in a victim's browser.
Password	1,153,323	Attacks to steal passwords (brute force, sniffing and related methods).
Injection	684,897	Supplying untrusted input to change a program's behavior (SQL/code injections).
Bot	143,097	Compromised hosts under remote control (botnet) used to perform malicious actions.
Brute Force	123,982	Repeatedly trying credentials to force entry and obtain sensitive information.
Infiltration	116,361	Insider or foothold-based compromise (via malicious download/backdoor) that exfiltrates data.
MITM	7,723	Man-in-the-Middle: intercepting traffic between parties to read or alter communications.
Ransomware	3,425	Malware that encrypts a victim's data and demands payment for the decryption key.
Fuzzers	22,310	Feeding random or malformed inputs to systems to expose faults and security bugs.
Backdoor	18,978	Hidden access mechanism that bypasses normal authentication/controls.
Generic	16,560	General attack patterns not tied to a single application or vulnerability.
Analysis	2,299	Activities that analyze systems/networks during early investigation phases.
Exploits	31,551	Attacks that use specific software or system vulnerabilities to gain control or access.
Shellcode	1,427	Small payload code used in exploits, often to open a backdoor or gain control.

Continued on next page

Class	Count	Description
Worms	164	Self-replicating malware that spreads across systems without user action.
Theft	2,431	Stealing sensitive data (personal/financial information), often via key-logging or exfiltration.

Table 3.2: NF-UQ-NIDS-v2: Class counts and brief descriptions of attack types.

3.3.2 Data Source and Sampling Strategy

The original dataset is extremely large, with over 25 million benign samples and varying amounts of different attack types ranging from a few thousand to over 20 million samples depending on the attack category. To make the training process manageable while keeping the data balanced, we applied a careful sampling strategy. For benign traffic, we randomly selected 500,000 samples from the 25 million available, which gives us enough data to learn normal behavior patterns without overwhelming the model. For attack traffic, we capped each attack type at 100,000 samples to prevent any single attack category from dominating the dataset. However, some rare attack types like Theft had only 2,431 samples available, Generic had 16,560 samples, and mitm had 7,723 samples, so we kept all available samples for these minority classes. The dataset was read in chunks of 500,000 rows at a time to avoid memory issues, and during this first pass we collected the row indices that met our sampling criteria. In a second pass, we loaded only the selected rows, resulting in a final dataset of approximately 1.48 million samples covering 14 distinct categories: one benign class and 13 attack types including DoS, DDoS, Reconnaissance, Bot, Brute Force, password attacks, injection, xss, malware and others.

3.3.3 Data Cleaning

The cleaning process started by removing rows that had missing values in the Attack column, which contained the attack type labels. Since the dataset combined traffic from different sources, some attack categories had similar meanings and were consolidated to avoid redundancy. For example, reconnaissance-related categories like Reconnaissance, Analysis, Fuzzers and Scanning were grouped together under a single Reconnaissance label. Similarly, malware-related attacks like ransomware, Worms, Shellcode, Backdoor and Exploits were unified under the malware category. This consolidation made the classification task more focused while still preserving meaningful distinctions between attack types. We also identified and removed features that could cause data leakage columns that are specific to individual connections and don't generalize well. Additionally, we dropped the source and destination IP address columns because keeping them would allow the model to memorize specific IPs rather than learning actual attack patterns, which would fail when deployed on new networks. After cleaning, the dataset had 46 original columns, which were split into 40 numeric features, 3 categorical features, and the label columns.

3.3.4 Data Transformation and Feature Engineering

The transformation process handled both categorical and numeric features separately to prepare them for the model. For the three categorical features in the dataset, we used label encoding fitted only on the training set. To handle cases where the validation or test sets might contain categorical values not seen during training, we explicitly mapped unknown categories to a special "out-of-vocabulary" token, preventing errors during inference. For numeric features, we implemented a robust preprocessing pipeline to deal with common issues in network traffic data. First, we replaced infinite values (both positive and negative infinity) with NaN, since some traffic features can produce division-by-zero or overflow. Then we filled all NaN values with the median value from the training set, which is more robust to outliers than using the mean. Next, we applied quantile-based clipping, keeping only values between the 0.1% and 99.9% percentiles calculated on training data. This removes extreme outliers that could destabilize training without throwing away legitimate rare events. After clipping, we used MinMax scaling to normalize all numeric features to the 0-1 range, fitting the scaler only on training data and then applying the same transformation to validation and test sets.

Rather than treating all features the same way during modeling, we organized them into three hierarchical levels based on what aspect of network traffic they represent. Level 1 contains packet-level features like packet size distributions (`NUM_PKTS_64`, `NUM_PKTS_128`, `etc.`), TCP flags, and ICMP types details that describe individual packets. Level 2 contains flow-level features like total incoming and outgoing packets (`IN_PKTS`, `OUT_PKTS`), byte counts (`IN_BYTES`, `OUT_BYTES`), retransmissions, and flow duration—statistics that summarize entire connections. Level 3 contains session-level features like average throughput rates (`AVG_THROUGHPUT`), directional statistics (`SRC_TO_DST_SECOND_BYTES`, `DST_TO_SRC_SECOND_BYTES`), and flow duration in milliseconds—higher-level behaviors that capture how traffic flows over time. This division allows the model to learn representations at different levels of abstraction, similar to how a network analyst might look at both individual packet details and overall session patterns when investigating suspicious activity. An important additional step was computing Poisson scaling parameters for each feature at each level, which are used later by the generative model's decoder to properly reconstruct count-based features like packet counts and byte counts.

3.3.5 Data Reduction

To avoid redundancy and reduce computational cost, we applied correlation-based feature reduction on the numeric features. After preprocessing, we computed the correlation matrix on the training set and identified pairs of features with correlation above 0.95, which indicates they carry nearly identical information. For each highly correlated pair, we kept one feature and dropped the other. This reduced the numeric feature count from 40 to 35, removing redundancy without losing important information. The correlation analysis was done separately for each of the three hierarchical levels (packet, flow, session), resulting in Level 1 having 10 features, Level 2 having 16 features and Level 3 having 12 features after reduction. This reduction not only speeds up training but also helps prevent overfitting, since highly correlated features can cause the model to give too much weight to essentially the

same information.

3.3.6 Summary of Preprocessed Data

After all preprocessing steps, the final dataset was split into training, validation and test sets using stratified sampling to maintain the same proportion of each attack class across all splits. We used approximately a 70-15-15 split, resulting in 1,070,932 samples for training, 188,988 samples for validation, and 222,339 samples for testing. The stratification is critical because the dataset is highly imbalanced, benign traffic makes up about 34% of samples while individual attack types range from less than 1% to around 7% of the data. The validation set was used for early stopping during model training and for fitting the post-training calibration components, while the test set was completely held out and never used during any training or tuning decisions. The training set consisted of 35 numeric features organized into three levels (10 packet-level, 16 flow-level, and 12 session-level features), 1 categorical feature after encoding, and target labels for 14 classes. The validation and test sets maintained the same feature structure. The class distribution remained imbalanced but manageable, with benign samples making up the largest portion at 75,000 test samples, while attack classes ranged from 365 samples (Theft) to 15,000 samples for most major attack types like Bot, DDoS, DoS, and Brute Force. This preprocessing pipeline ensured that the data fed into the model was clean, properly scaled, and organized in a way that supports hierarchical learning at multiple levels of network traffic abstraction, while maintaining realistic class imbalance that reflects real-world network environments.

3.4 Implementation of Selected Design

All the development in this research was conducted using Python 3.11, leveraging PyTorch, NumPy, scikit-learn, XGBoost, stable-baselines3, gymnasium, pandas, and matplotlib libraries. The system was executed on a high-performance desktop equipped with an Intel Core i7-14700K CPU (20 cores, 28 threads), 64 GB RAM, and an NVIDIA RTX 4080 SUPER GPU with 16 GB VRAM, ensuring smooth handling of the complex multi-stage training pipeline and large-scale data operations with GPU acceleration through CUDA-optimized implementations.

We worked with the NF-UQ-NIDS-v2 dataset, which is a large-scale intrusion detection dataset which covers both benign traffic and multiple types of network attacks. During preprocessing, we applied sampling limits to keep things balanced: benign traffic was capped at 500,000 samples and each attack type was limited to 100,000 samples. To make the labels more manageable, we grouped related attacks together for example, reconnaissance-related categories (Reconnaissance, Analysis, Fuzzers, Scanning) were combined and malware-related ones (Ransomware, Worms, Shellcode, Backdoor, Exploits) were unified while still preserving the ability to perform fine-grained classification. The preprocessing pipeline also handled numeric issues robustly where infinities and NaNs were imputed with the training set median, extreme values were clipped using quantiles (keeping only the 0.1%–99.9% range) and features were scaled with MinMax normalization to stabilize convergence during training.We designed a new multi-level feature engineering method that organizes

the features into three layers. The first layer focuses on packet-level details, such as packet size distributions, TCP flags and ICMP types. The second layer is about flow-level statistics, which includes packet and byte counts, durations and retransmissions. Finally, the third layer captures session-level context, such as throughput rates and directional statistics. To avoid redundancy, features with correlations higher than 95% were removed and we calculated Poisson scaling parameters for each level to fine-tune the generative modeling components.

For the initial setup, we used the open-source PoissonVAE framework developed by Vafaii [48] as our starting point, then modified and expanded it extensively to support the hierarchical design.We restructured the base PVAE into a three-level chained architecture, where each level is responsible for processing features at different granularities using its own encoder-decoder pair, while sharing categorical embeddings across levels. Each PVAE level integrates several advanced components like multilayer perceptron encoders that map inputs into 64-dimensional latent spaces, normalizing flows such as Planar Flows for simple transformations, Affine Coupling layers with context conditioning to capture more complex dependencies and Masked Autoregressive Flows (MAF) to handle autoregressive patterns. To improve latent space alignment, we introduced a class-conditional LRQuantileRadialCalibrator, which performs per-class radial mapping via piecewise-linear interpolation fitted on validation data to better match the expected N(0, I) distribution. In addition, we implemented Bridge Flows as conditional Real-NVP networks, allowing information to flow between PVAE levels and linking the representations across the hierarchy.In the decoder design, we used Poisson distributions instead of Gaussian ones, since features such as packet counts and flow statistics are naturally count-based. To avoid the usual training instabilities that come with complex normalizing flow models, we added a PFGM-inspired regularization term, which helps control changes in volume and radial expansion. The learning rates were set differently for different parts of the model 1e-3 for the main parameters and 5e-4 for the bridge flows and we applied gradient clipping with a norm of 5.0 to keep convergence stable. On top of this, we integrated a reinforcement learning approach through a UCB bandit sampler. This component adaptively updated the per-class weights based on validation cross-entropy, allowing the system to handle class imbalance on the fly. Training was carried out for 30 epochs with early stopping, using generative fit metrics (ELBO and NLL) as the criteria. We also included proxy classification heads to support both multiclass and binary prediction tasks.Post-training latent refinement was achieved through a tiny residual MLP warp (TinyWarp) that applies small corrections to the learned representations, trained via a linear probe objective with class-weighted cross-entropy loss and L2 regularization to maintain proximity to the original latent space. To create ensemble diversity without training multiple separate models, we instantiated three architectural variants from the same trained PVAE checkpoint: Base A retains the full model with recalibration and normalizing flows enabled, Base B operates with recalibration disabled to produce alternative latent geometries, and Base C removes all normalizing flow components to provide a pure VAE perspective. This approach generates complementary decision surfaces from a single training run, improving computational efficiency while maintaining ensemble diversity.Instead of assigning static weights to each base learner, we trained a Proximal Policy Optimization (PPO) agent to learn dynamic, per-sample weighting

Algorithm 1 Hierarchical PVAE with RL-based Ensemble: Training and Inference

Require: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, epochs E, PPO timesteps T_{PPO}

// TRAINING PHASE
Phase 1: Train Multi-Level PVAE Chain
for epoch $e = 1$ to E **do**
  for mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
    for level $l \in \{1, 2, 3\}$ **do**
      Encode $x^{(l)} \to \mu_l, \sigma_l$; Sample $z_0 \sim \mathcal{N}(\mu_l, \sigma_l^2)$
      Apply normalizing flows: $z_0 \to z_K$ (Planar, Coupling, MAF)
      if $l = 3$ **then**
        Apply class-conditional recalibration: $z_R = \text{Recalib}(z_K, y)$
      else
        $z_R = z_K$
      end if
      Decode: $\hat{\lambda}^{(l)} = \text{Softplus}(\text{Dec}_l(z_R))$
      Compute $\mathcal{L}_{\text{ELBO}}^{(l)}$ (Poisson reconstruction + KL + flow correction)
    end for
    Compute bridge losses $\mathcal{L}_{\text{bridge}}$, regularization $\mathcal{L}_{\text{reg}}$, proxy losses $\mathcal{L}_{\text{proxy}}$
    $\mathcal{L} = \sum_l \mathcal{L}_{\text{ELBO}}^{(l)} + \beta\mathcal{L}_{\text{bridge}} + \mathcal{L}_{\text{reg}} + \alpha\mathcal{L}_{\text{proxy}}$
    Update all parameters via gradient descent
  end for
end for
Phase 2: Fit Radial Recalibrator
Collect validation latents $\{z_K^{(3)}, y\}_{\text{val}}$; Fit Recalib.fit$(\cdot)$
Phase 3: Create Ensemble Bases
Base A: Load trained model (recalib ON, flows ON)
Base B: Load trained model (recalib OFF, flows ON)
Base C: Load trained model (recalib OFF, flows OFF)
Phase 4: Train PPO for Dynamic Weighting
for $t = 1$ to T_{PPO} **do**
  Get base predictions: P_A, P_B, P_C and diagnostics (entropy, density, sparsity)
  Construct state: $s = [P_A, P_B, P_C, \text{var}(P), H, \rho, \xi]$
  PPO policy: $a = \pi(s)$; Weights: $w = \text{StickBreaking}(a)$
  Weighted prediction: $P^{\text{mix}} = \sum_k w_k P_k$
  Compute multi-objective reward: $r = \text{gain} - \text{cost}_{\text{FP}} - \text{cost}_{\text{FN}} + \text{bonuses}$
  Update PPO policy π via clipped surrogate objective
end for
Phase 5: Train XGBoost Meta-Learner
Generate features: $X_{\text{meta}} = [P^{\text{mix}}, w, \text{var}(P), \bar{\rho}, \bar{H}]$
Train: $f_{\text{meta}} = \text{XGBoost}(X_{\text{meta}}, y, \text{weights} = w_{\text{class}})$
// INFERENCE PHASE
Given: New input x with features $(x^{(1)}, x^{(2)}, x^{(3)}, x_{\text{cat}})$
Step 1: Get Base Predictions
for base $k \in \{A, B, C\}$ **do**
  Forward pass through 3-level PVAE: $x \to z_R^{(k)} \to P_k$ (deterministic)
  Extract diagnostics: H_k, ρ_k, ξ_k
end for
Step 2: PPO Weighting
State: $s = [P_A, P_B, P_C, \text{var}(P), H, \rho, \xi]$
Weights: $w = 0.7 \cdot \text{StickBreaking}(\pi(s)) + 0.3 \cdot [1/3, 1/3, 1/3]$
Weighted: $P^{\text{mix}} = \sum_k w_k P_k$
Step 3: XGBoost Refinement
Meta-features: $X_{\text{meta}} = [P^{\text{mix}}, w, \text{var}(P), \bar{\rho}, \bar{H}]$
Final: $P^{\text{final}} = f_{\text{meta}}(X_{\text{meta}})$; Prediction: $\hat{y} = \arg\max P^{\text{final}}$
return Training: Models $\{\theta^*\}, \pi, f_{\text{meta}}$ — Inference: $\hat{y}, P^{\text{final}}$

across the three generative variants. The agent operated inside a custom gymnasium environment with rich state representations including per-base probability distributions, cross-base variance, predictive entropy, flow log-densities from bridge networks, and Poisson sparsity metrics (median rates and spike fractions).

The reward function was designed with multiple objectives: task performance gain relative to a baseline XGBoost model, cost-sensitive penalties with benign false positives weighted significantly higher (FP cost=12.0 vs. FN cost=0.7). The reward also encouraged good calibration (so confidence matched accuracy), preferred simpler and more efficient representations, and penalized uncertain or out-of-distribution samples. Finally, it added a small entropy bonus so that the PPO wouldn't always rely on the same base model and could keep its decisions diverse. The PPO agent was trained for 120,000 timesteps, with learned weights blended 70/30 with uniform weights for stability. For the final prediction layer, we employed an XGBoost meta-learner configured with 550 estimators, maximum depth of 7 and learning rate of 0.055. The meta-learner receives an enriched feature space combining PPO-weighted ensemble probabilities, PPO weight allocations, cross-base variance and mean density and entropy diagnostics. Class imbalance was tackled using inverse-frequency weighting normalized to unit mean, with the benign class further emphasized (1.3x weighting) to penalize false positives more heavily which is a practical consideration for deployment. The full system was optimized for GPU acceleration on CUDA, using persistent data loaders with prefetching and memory pinning to improve transfer efficiency. Batch sizes were set to 2048 for the generative training phase and 8192 for latent encoding. Performance evaluation encompassed comprehensive metrics including multiclass accuracy, log loss, per-class AUROC analysis, binary classification metrics (benign vs. attack with binary AUROC and PR-AUC), precision-recall curves for all classes, and false positive rates at fixed true positive rate thresholds (90%, 95%, 99%). Advanced interpretability techniques were implemented including XGBoost-based feature importance analysis on the top 16 latent dimensions to identify critical channels and gradient-based probing through backpropagation on the L3 encoder to determine which input features most influence important latent dimensions. The design emphasizes both performance and interpretability, making it extensible for deployment in real-world network security environments with dynamic threat landscapes and operational constraints. The whole architecture for our Hierarchical Model is shown in Figure 3.3.

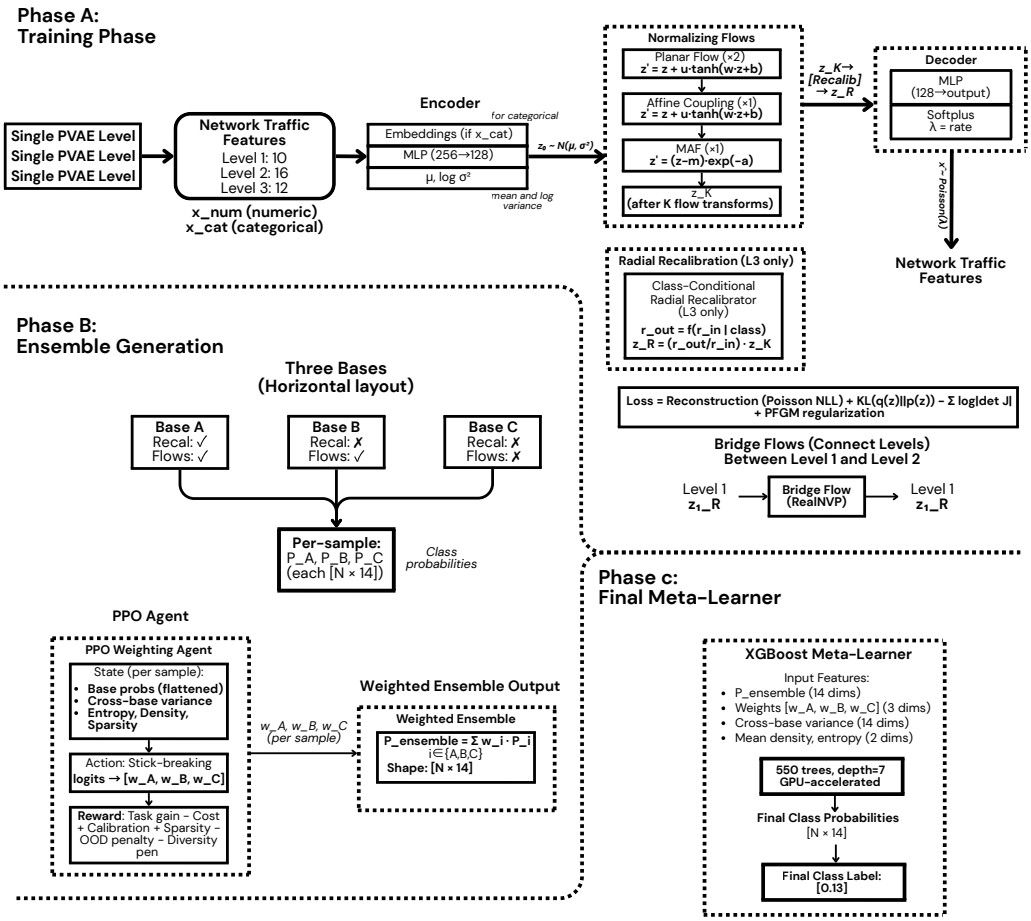


Figure 3.3: Architecture of our Hierarchical Model

Chapter 4

Result Analysis

4.1 Performance Evaluation

The proposed multi-level generative framework in this research is evaluated through a comprehensive multi-stage process that examines both the quality of learned representations and the final classification performance. Unlike traditional binary detection approaches, this work addresses the more challenging task of fine-grained multiclass intrusion detection across 14 distinct traffic categories (1 benign + 13 attack types), while maintaining strong binary discrimination capability between normal and malicious traffic.The evaluation strategy encompasses three complementary dimensions: (1) latent space quality assessment through visualization and geometric analysis of the hierarchical PVAE representations, (2) multiclass classification performance using standard metrics adapted for imbalanced, multi-label scenarios and (3) binary detection capability evaluated through ROC-AUC and precision-recall analysis.

In the first stage of this research, the training process of the pvae remained stable across epochs, as observed in the generative loss curve (Figure 4.1). Both training and validation losses followed a similar trend with no major divergence, indicating that the hierarchical PVAE chain generalized well and learned consistent feature representations before moving into the evaluation phase.

Figure 4.1: Generative loss curve showing training and validation losses over epochs.

Latent Space Quality and Representation Learning :

To assess whether the three-level PVAE chain successfully learned meaningful hierarchical representations, we first examined the top-level latent space (zR) after

class-conditional radial recalibration. The t-SNE visualization of the 64-dimensional latent vectors on 8,000 test samples reveals well-separated manifold structures for most attack classes, with particularly distinct clusters for Bot (class 1), Brute Force (class 2), DDoS (class 3) and DoS (class 4) attacks. This geometric separation indicates that the generative modeling process effectively disentangled attack-specific patterns despite being trained primarily through reconstruction objectives rather than supervised discrimination. Notably, the benign class (class 0) occupies a coherent central region in the embedding space, with attack classes distributed around the periphery in distinct lobes. Some overlap is observed between semantically related attack types such as injection (class 9) and xss (class 13), reflecting genuine similarities in their network-level manifestations. The quality of these learned representations provides strong motivation for the ensemble meta-learning stage, as base classifiers operating on such well-structured latent spaces can more reliably identify decision boundaries.To quantify the importance of individual latent dimensions, we trained an XGBoost classifier directly on the 64-dimensional latent vectors and extracted feature importance scores. The top 16 latent channels exhibited relatively balanced importance (ranging from 0.0163 to 0.0255), suggesting that the PVAE chain distributes information across the latent space rather than concentrating it in a few dominant dimensions. The most important channels (z[1], z[26], z[44]) collectively contribute approximately 7% of the total importance, indicating a distributed representation strategy that enhances robustness to perturbations.

Multiclass Classification Performance :

The final ensemble system, consisting of three PVAE architectural variants dynamically weighted by a PPO agent and refined through an XGBoost meta-learner, achieved an overall accuracy of 89.17% on the 222,339 sample test set spanning 14 classes. The weighted F1-score across all classes was 88.01%, with a macro-averaged F1 of 86.16%, demonstrating reasonably balanced performance despite significant class imbalance (benign samples outnumber minority classes by 30:1 to 200:1). The classification report reveals strong per-class performance for high-volume attack types:

- **Bot (class 1):** Precision 99.96%, Recall 99.91%, F1 99.93%
- **Brute Force (class 2):** Precision 99.65%, Recall 97.81%, F1 98.72%
- **DDoS (class 3):** Precision 95.86%, Recall 95.21%, F1 95.53%
- **DoS (class 4):** Precision 96.01%, Recall 96.17%, F1 96.09%

These results indicate near-perfect detection for bot-related traffic and highly reliable identification of volume-based attacks (DDoS, DoS), which is operationally critical for preventing service disruptions.For minority classes with limited training samples, performance varies:

- **Generic (class 5):** Precision 96.78%, Recall 71.38%, F1 82.16% (2,484 samples)

- **Theft (class 8):** Precision 99.15%, Recall 95.34%, F1 97.21% (365 samples)

The high precision coupled with moderate recall for Generic attacks reflects the system's conservative behavior on rare classes, which is a desirable property in production settings where false alarms are costly.

The most challenging class proved to be Infiltration (class 6), achieving only 27.48% recall despite 90.81% precision, resulting in an F1-score of 42.19%. This indicates that Infiltration attacks, which typically mimic legitimate traffic patterns through stealthy lateral movement, remain difficult to distinguish even with sophisticated generative modeling. The class-conditional radial recalibration and PPO reward design explicitly addressed this through class-aware false negative costs (boosted by 1.35× for class 6), but substantial improvement would likely require additional domain-specific features or longer temporal context. The confusion matrix shows that the majority of misclassifications occur between semantically similar attack types rather than between benign and malicious traffic. For instance, injection attacks (class 9) are occasionally confused with xss attacks (class 13), both of which involve code insertion techniques. The strong diagonal structure of the row-normalized confusion matrix confirms that most classes achieve recall above 70%, with perfect or near-perfect recall for classes 1, 2, 4, and 10. As depicted in Figure 4.3.

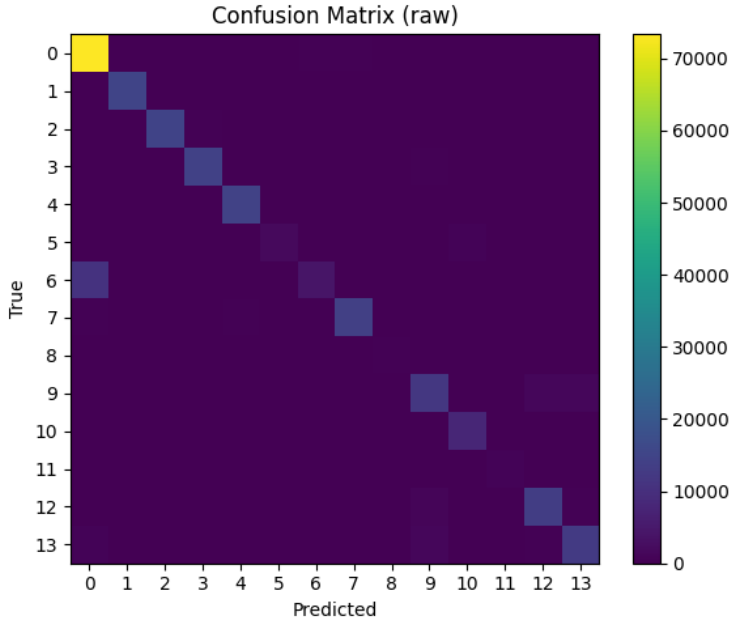


Figure 4.2: Confusion matrix (raw counts) showing the distribution of predictions across 14 classes on the test set.

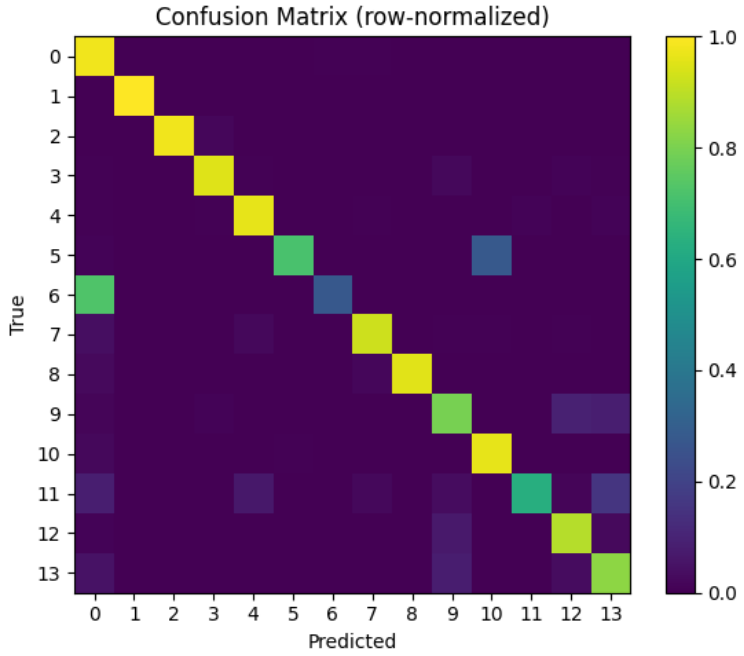


Figure 4.3: Row-normalized confusion matrix illustrating per-class recall rates, with diagonal elements representing correct classification proportions.

To contextualize these multiclass results, we computed additional aggregate metrics:

- **Macro AUROC:** 99.23%, demonstrating excellent class separability
- **Micro PR-AUC:** 95.93%, showing strong precision-recall balance at the sample level

Binary Classification Performance (Normal vs. Attack) :

For operational deployment, the ability to distinguish benign traffic from any form of attack is paramount. We evaluated the system in binary mode by aggregating all attack classes (1–13) into a single Attack label and measuring performance against the Benign class (0) as depicted in Figure 4.4. The binary confusion matrix reveals:

- **True Negatives (TN):** 63,526 benign samples correctly identified
- **False Positives (FP):** 11,474 benign samples incorrectly flagged as attacks
- **False Negatives (FN):** 7,367 attacks missed
- **True Positives (TP):** 139,972 attacks correctly detected

From these counts, we compute key operational metrics: False Positive Rate (FPR):

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{11,474}{11,474 + 63,526} \approx 0.1530 \ (15.30\%)$$

This indicates that approximately 15.3% of legitimate traffic would be flagged for inspection, a significant improvement over threshold-based anomaly detection approaches, though still requiring careful tuning in high-throughput environments. False Negative Rate (FNR):

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} = \frac{7,367}{7,367 + 139,972} \approx 0.0500 \ (5.00\%)$$

This reflects that 5% of attacks are missed, primarily concentrated in the difficult Infiltration class.

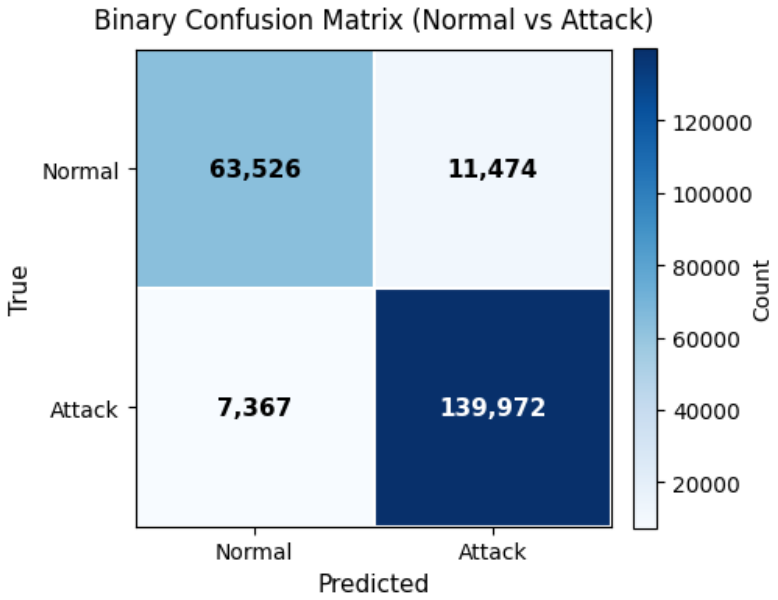


Figure 4.4: Binary confusion matrix for Normal vs. Attack classification

The reward design in the PPO agent explicitly weighted benign false positives 17× higher than attack false negatives (FP cost = 12.0 vs. FN cost = 0.7), for operational contexts where false alarms disrupt business continuity. Specificity (True Negative Rate, TNR):

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{63,526}{63,526 + 11,474} \approx 0.8470 \ (84.70\%)$$

The specificity of 84.70% demonstrates that the majority of benign traffic passes through unimpeded, aligning with the design goal of minimizing operational friction.Detection Rate (True Positive Rate, TPR/Recall):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{139,972}{139,972 + 7,367} \approx 0.9500 \ (95.00\%)$$

The 95% detection rate indicates strong sensitivity to malicious activity across diverse attack types.
The binary ROC curve achieves an AUROC of 0.9811, significantly outperforming traditional anomaly detection methods as depicted in Figure 4.5. The curve exhibits rapid ascent in the low-FPR region, indicating that excellent detection rates ($> 90\%$) can be achieved with minimal false alarms ($< 2\%$).

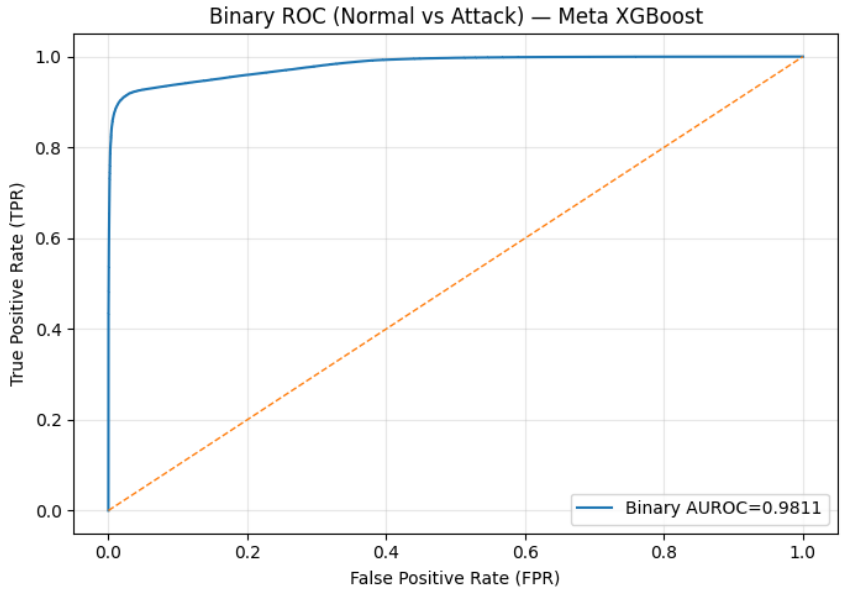


Figure 4.5: Binary ROC curve (Normal vs. Attack) with AUROC = 0.9811.

At operationally relevant thresholds:

- TPR $\approx$ 90%: FPR = 1.57% (threshold = 0.0662)
- TPR $\approx$ 95%: FPR = 15.30% (threshold = 0.1290)
- TPR $\approx$ 99%: FPR = 36.68% (threshold = 0.2114)

These results demonstrate that the system can be tuned for high-security environments (99% detection at the cost of 37% false alarms) or balanced operational settings (95% detection with 15% false alarms), depending on organizational risk tolerance. The precision-recall curve yields a PR-AUC of 0.9911 (Figure 4.6), indicating exceptional balance between precision and recall across the entire threshold spectrum. This metric is particularly meaningful for imbalanced scenarios, as it directly reflects the system's ability to maintain high precision even when maximizing recall, a critical property for intrusion detection systems deployed in asymmetric threat landscapes.

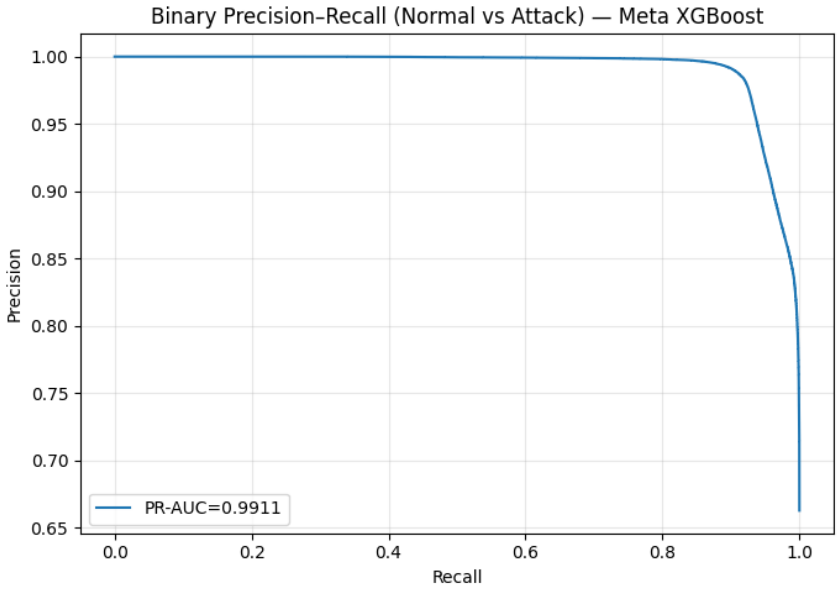


Figure 4.6: Binary Precision–Recall curve with PR-AUC = 0.9911.

Per-Class Diagnostic Analysis :
Examining per-class metrics reveals nuanced insights into system behavior:For the benign class (class 0):

- **Precision:** 84.91%
- **Recall:** 97.80%
- **F1:** 90.90%
- **AUROC:** 98.11%

The high recall (97.80%) on benign traffic confirms that legitimate users experience minimal disruption, while the 84.91% precision indicates that when the system predicts benign, it is correct 85% of the time. For high-stakes attack classes with severe operational impact:

- **Malware (class 10):** Precision 89.87%, Recall 96.11%, F1 Score 92.88%
- **Password Attacks (class 12):** Precision 85.36%, Recall 89.01%, F1 Score 87.14%

These results demonstrate that the system prioritizes detection of credential theft and malware propagation, justifying the class-aware false negative costs in the PPO reward function. The per-class precision-recall curves (Figure 4.7) illustrate the diversity of class difficulty. Classes with high average precision (AP $>$ 99%) include Bot, Brute Force, DDoS, DoS, Generic, and Theft, indicating near-perfect ranking of samples by confidence. Conversely, Infiltration (class 6) exhibits lower AP $\approx$ 62.5%, consistent with its low recall in the classification report.

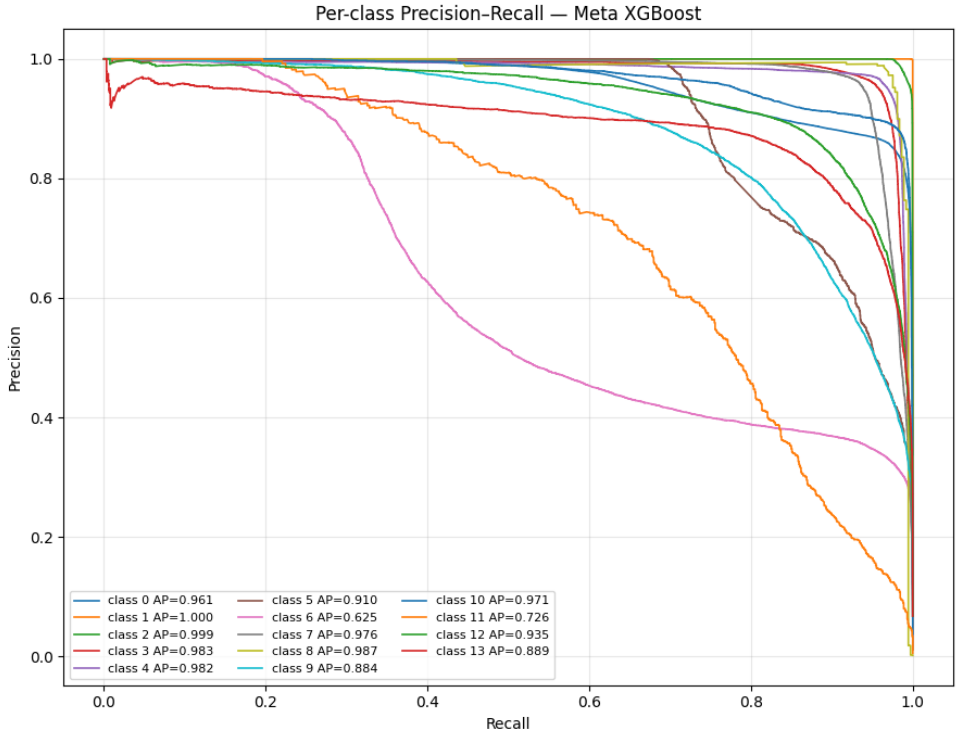


Figure 4.7: Per-class Precision–Recall curves for all 14 classes.

4.2 Analysis of Design Solutions

The proposed architecture uses a multi-stage design that brings together three main components: a hierarchical PVAE chain to learn network traffic patterns at multiple levels, a PPO-based reinforcement learning agent to adjust ensemble weights on the fly and an XGBoost meta-learner for the final classification step. This setup was intentionally built to strike a balance between several competing goals, achieving strong detection performance without triggering too many false alarms, handling 14 different types of traffic instead of just binary distinctions and producing results that are interpretable enough for security analysts to understand and act on confidently. The system creates ensemble diversity efficiently by instantiating three architectural variants from a single trained checkpoint rather than training separate models, which saves computational resources. The PPO agent learns to weight these variants on a per-sample basis by looking at prediction confidence, cross-base disagreement, and uncertainty signals, with a reward function that heavily penalizes false positives on benign traffic. The XGBoost meta-learner then refines these weighted predictions using additional uncertainty diagnostics. In terms of performance, the system reaches an overall accuracy of 89.17% across 14 traffic classes, with per-class AUROC scores ranging from 95.39% up to 100%. It also shows strong binary classification, achieving 98.11% AUROC and 99.11% PR-AUC. At a 95% detection rate, the false positive rate sits at 15.3%, which is generally acceptable for most real-world deployments. The model performs exceptionally well on high-volume attack types such as Bot (99.93% F1), DDoS (95.53% F1), and DoS (96.09% F1). Even for less frequent categories like Theft, it maintains impressive precision at 99.15%, despite having limited training data. However, stealthy attacks like Infiltration are still hard to de-

tect, with recall falling to 27.48%, showing that these cases need more improvement.

The main strengths of this approach lie in its ability to classify threats in detail rather than just labeling traffic as normal or malicious. Its cost-sensitive design aligns well with real-world operational needs and the interpretability features make it possible to trace predictions back to key network factors such as throughput imbalances and protocol identifiers. The use of gradient-based attribution also helps analysts understand why an alert was triggered, which builds confidence in the system's decisions. Although a 15.3% false positive rate is fairly reasonable, high-traffic networks may still need manual checks during peak periods. The system depends on labeled training data for its supervised components, which can be tricky for organizations that don't have enough recorded attack data to start with. In terms of practicality and cost, it makes the most sense for medium to large organizations that already have access to GPUs and dedicated security teams, since they're better equipped to handle the setup and get real value from it. Smaller organizations or edge deployments would probably be better off using simpler binary models, even if that means giving up some of the deeper insights that multiclass classification can provide. Overall, the system does a solid job of balancing performance and interpretability, offering reliable accuracy along with the kind of diagnostic details that security teams actually find useful during real-world threat investigations.

4.3 Statistical Analysis

A comprehensive set of statistical metrics and visualization techniques were used to explain the performance and reliability of the proposed multi-level generative ensemble system. The primary evaluation metric for multiclass classification was the Receiver Operating Characteristic and Area Under the Curve (ROC-AUC), which measures the model's ability to discriminate between the 14 traffic classes across all possible decision thresholds. The system achieved per-class AUROC values ranging from 95.39% to 100%, with 10 out of 14 classes exceeding 99%, demonstrating excellent separability for most attack types. For binary classification (normal vs. attack), the system reached an AUROC of 98.11% and a Precision-Recall AUC of 99.11%, indicating strong discrimination capability that matches or exceeds many binary-only detection systems. Beyond ROC analysis, we evaluated classification performance using accuracy (89.17%), macro-averaged F1-score (86.16%), all of which confirmed robust performance despite significant class imbalance in the dataset. To understand the quality of the learned representations, t-distributed Stochastic Neighbor Embedding (t-SNE) was applied to visualize the 64-dimensional top-level latent space (zR) after class-conditional radial recalibration. The t-SNE projection of 8,000 test samples revealed well-defined manifold structures for most classes, with high-volume attack types like Bot (class 1), Brute Force (class 2), DDoS (class 3) and DoS (class 4) forming distinct clusters, while the benign class (class 0) occupied multiple regions in the embedding space as depicted in Figure 4.8. Some overlap was observed between semantically similar attack types such as injection (class 9) and xss (class 13), which is expected given their shared characteristics at the network level. The geometric separation visible in the t-SNE plot validates that the hierarchical PVAE chain successfully disentangled attack-specific patterns into a structured latent space, providing strong motivation for the ensemble meta-learning approach.

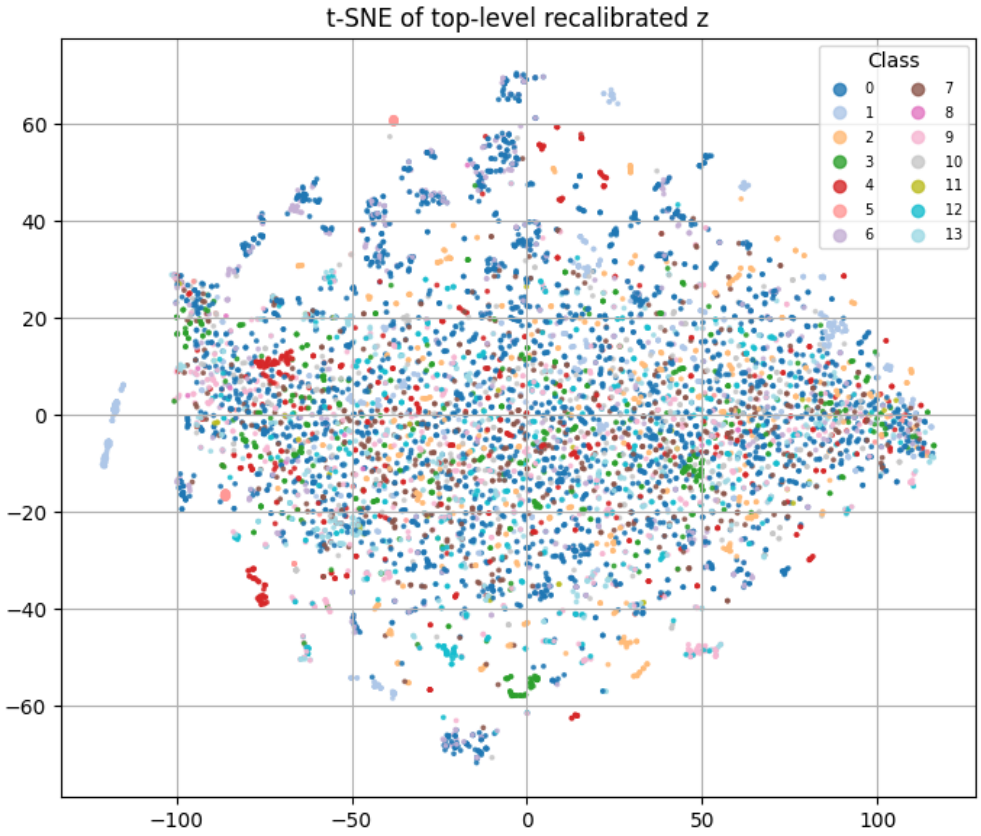


Figure 4.8: t-SNE visualization of top-level recalibrated latent space (z_R) showing the distribution of 8,000 test samples across 14 classes.

Additionally, we computed cost-sensitive metrics at operationally relevant thresholds, finding that at 95% True Positive Rate (detection rate), the system maintained a False Positive Rate of 15.3%, which translates to approximately 1 in 6 benign samples being flagged for review, a reasonable trade-off for security operations centers that prioritize catching attacks over minimizing false alarms.

4.4 Comparisons and Relationships

To contextualize the performance of our proposed system, we compare it against two groups of related work: studies that employ similar hybrid deep learning and reinforcement learning architectures across various datasets and studies that specifically target the NF-UQ-NIDS-v2 dataset used in this research.

The first group, summarized in Table 4.1, includes recent approaches that combine hybrid models or deep networks with DRL agents for intrusion detection. For instance, the Adaptive Defense system (2025) uses a DRL agent with stacked LSTM and achieves 95-99% accuracy on binary classification tasks across multiple datasets including NF-BoT-IoT and NF-UNSW-NB15. Similarly, the DRL-GAN approach (2023) combines CTGAN and CopulaGAN with a DRL model based on PPO2, achieving 89.5% accuracy on the NSL-KDD dataset. The Multi-agent Reinforcement Learning system (2024) reports 99% accuracy and 92% AUC using a hybrid

DQN-MARL architecture on CIC-IDS-2017. While these works demonstrate the potential of combining deep learning with reinforcement learning, most focus on binary classification (normal vs. attack) rather than fine-grained multiclass detection. Our approach differs by using PPO to dynamically weight an ensemble of generative base learners (three PVAE variants) rather than training the RL agent to directly classify traffic or generate synthetic samples. This architectural choice allows us to leverage the uncertainty quantification and interpretability benefits of generative models while using RL to handle the ensemble weighting problem in a cost-sensitive manner.

Name	**Year**	**Datasets**	**Model/ Hybrid**	**Metrics**	**Strengths/ Weaknesses**
Adaptive Defense: Zero-Day Attack Detection in NIDS With Deep RL	2025	NF-BoT-IoT (pri.), NF-ToN-IoT, NF-UNSW-NB15, NF-UQ-NIDS	DRL agent with stacked LSTM	95–99% acc., 99% prec., rec., F1	**Strengths:** adaptability to unseen attacks, multi-dataset robustness. **Limitations:** high computational req., no explicit feature selection. Limited to binary classification
Deep Reinforcement Learning for IDS: Survey	2023	NSL-KDD (U2R/R2L), UNSW-NB15, CICIDS-2017, CIC-DDoS2019, BoTNeT-IoT-101	DRL with CNN, SVM, or PPO2	90–99% acc., FPR <5%	**Strengths:** taxonomy, performance summaries for DRL in IoT. **Limitations:** 2018-2022 only, no original experiments, limited deployment
Survey: Deep RL Network IDS	2024	NSL-KDD, UNSW-NB15, CIC-IDS-2017/18, LITNET-2020, Bot-IoT	DQN, actor-critic, AE-DQN	81–99% Acc., Prec., F1	**Strengths:** adaptive to evolving threats. **Limitations:** high comp. costs, poor imbalanced data handling, limited real-world use

Continued on next page

Name	Year	Datasets	Model/ Hybrid	Metrics	Strengths/ Weaknesses
Multi-agent RL Network IDS	2024	CIC-IDS-2017	hybrid DQN-MARL	99% acc., 92% AUC	**Strengths:** adaptable to new attacks, handles imbalance. **Limitations:** comp. overhead, lower AUC on rare classes
DRL-GAN: Hybrid Binary/Multi Network IDS	2023	NSL-KDD	CTGAN, CopulaGAN with DRL (PPO2)	89.5% acc.	**Strengths:** rare attack detection. **Limitations:** NSL-KDD outdated features, redundancies

Table 4.1: Comparison of reinforcement learning-based hybrid intrusion detection approaches from existing research.

When comparing with studies that use the same NF-UQ-NIDS-v2 dataset, the performance trade-offs become clearer which is shown in Table 4.2. Binary classification approaches on this dataset achieve higher raw accuracy: the Ensemble feature selection using six classifiers system [45] reaches 95% accuracy with 95.1% Precision and 95% Recall, ExtraTrees with feature selection [49] achieves 98% accuracy with 99% AUC, and the Multi-Model Federated Learning with RNN, LSTM and GRU [37] hits 90% accuracy. Our system, tackling the significantly harder multiclass problem with 14 distinct categories, achieves 89.17% accuracy with 88.01% weighted F1-score. However, when evaluated in binary mode (normal vs. attack), our system is competitive with 98.11% AUROC and 99.11% PR-AUC, demonstrating that the lower multiclass accuracy is not due to poor model quality but rather the inherent difficulty of fine-grained classification. Our system provides richer diagnostic outputs including uncertainty estimates, per-sample ensemble weights, and gradient-based feature attributions that the simpler MLP approach cannot offer. The key advantage of our system lies in its ability to provide actionable threat intelligence: knowing that traffic is specifically a Bot attack versus a DDoS attack enables targeted response strategies, whereas binary systems only indicate attack detected without classification details. Our multi-stage pipeline (PVAE $\rightarrow$ PPO $\rightarrow$ XGBoost) requires more resources than the simpler binary classifiers, but for organizations that need detailed threat classification and can afford GPU infrastructure, the additional insights justify the cost. Furthermore, our system's interpretability features (latent channel importance ranking, gradient-based input attribution) address a critical gap in existing work, as most studies in both tables provide limited explanation of why specific traffic was flagged as malicious. In summary, while binary-focused approaches achieve higher raw accuracy on simpler tasks, our multiclass generative ensemble with RL-based weighting offers a more complete solution for real-world security operations centers that require both accurate detection and

actionable intelligence with transparency.

	Task	Accuracy	Precision	Recall	F1	AUC
Our Work Multi-level PVAE + PPO + XGBoost Meta	Multiclass	89.17%	89.51%	89.17%	88.01%	98.11% (Binary) 99.11% (PR-AUC)
Multi-Model Federated Learning with RNN, LSTM and GRU [37]	Binary	90.33%	—	—	—	—
Ensemble feature selection using six classifiers [45]	MultiClass	95%	95.1%	95%	-	—
ExtraTrees + Feature Selection [49]	Binary	98%	—	98%	—	99%

Table 4.2: Comparison of detection performance on NF-UQ-NIDS-v2 dataset.

4.5 Limitations

Our approach works well for detailed intrusion detection, but it has faced some crucial limitations. First, the NF-UQ-NIDS-v2 dataset is huge, almost 12 million rows. We had to sample and process it in chunks, so we couldn't use every record. That likely means we missed some rare attack types. Second, our research is the first to do 14-class classification on this dataset, so there were no past results or benchmarks to guide us. We had to make design choices without a clear yardstick. Third, the classes are very uneven. Some attacks have 100,000+ examples, but others are tiny: Theft has 2,431, Generic 16,560 and MITM 7,723. This hurt performance on the small classes like on Infiltration, for example, reaching only 27.48% recall even after using cost-sensitive learning and class-aware weights. Finally, our multi-stage pipeline, which includes training the PVAE chain, fitting the recalibrator, training the PPO agent, then XGBoost needs strong GPUs and takes hours. That makes it hard for smaller institutions without high-end hardware to adopt accordingly.

Chapter 5

Conclusion

5.1 Summary of Findings

This research tackles one of the hardest problems in network security: detecting not just whether traffic is malicious, but specifically what type of attack it represents, while keeping false alarms low enough for real-world use. Traditional intrusion detection systems struggle with this because they either focus on binary detection and miss important details, or they try multiclass classification but end up flagging too much normal traffic as suspicious. Our approach takes a different path by using generative models that learn what network traffic looks like at different levels of detail, from individual packets to entire sessions and then combines multiple perspectives using reinforcement learning to decide which view to trust for each sample. The results show this strategy works: the system correctly identifies 14 different types of attacks with nearly 90% accuracy while maintaining strong ability to separate normal traffic from malicious activity overall. What makes this especially promising is that the system doesn't just give predictions it also explains which network features led to each decision and provides uncertainty estimates that tell analysts when to be cautious about a detection.

5.2 Contributions to the Field

The main contribution of this work is showing that generative models can be used effectively for intrusion detection beyond simple anomaly detection. Rather than just flagging unusual traffic, the system provides detailed classification of specific attack types while also quantifying uncertainty through entropy, density, and sparsity metrics. The class-conditional radial recalibration technique introduced here helps align the learned latent space with theoretical expectations separately for each attack class, improving discrimination between similar threats. The use of reinforcement learning to dynamically weight ensemble members on a per-sample basis is also novel in this context, allowing the system to adapt its behavior depending on whether it is confident or uncertain about a particular sample. The cost-sensitive design, which explicitly penalizes false positives on benign traffic much more heavily than false negatives on attacks, makes the system more practical for real-world deployment where false alarms disrupt operations and erode trust in security systems.

5.3 Future Work

In the future, we aim to extend this research in many directions that can make the system more robust and accessible. Moreover, the next part of our work will focus on incorporating temporal modeling to better catch stealthy attacks that unfold gradually over time. We also plan to explore model compression techniques that can reduce the computational requirements without sacrificing accuracy, which will make this technology available to smaller organizations that don't have dedicated GPU infrastructure. Again, another direction we're pursuing is adapting the system for federated learning scenarios, where multiple organizations can benefit from shared learning without exposing their sensitive network traffic to each other. On the interpretability side, we believe this research can be extended to provide not just explanations of why traffic was flagged, but also actionable guidance on what specific changes would make suspicious traffic look benign helping analysts understand the boundary between normal and malicious behavior. Our main goal will focus on that, this work demonstrates a path toward building intrusion detection systems that learn and adapt continuously. This will adjust their detection strategies as they encounter new types of attacks rather than requiring manual retraining and redeployment. Finally these techniques we plan to develop in our research will be able to serve as building blocks for future security systems that combine the pattern recognition power of deep learning with the adaptive decision-making capabilities of reinforcement learning.

Bibliography

[1] K. Scarfone, P. Mell, et al., "Guide to intrusion detection and prevention systems (idps)," *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.

[2] Y. Al-Nashif, A. A. Kumar, S. Hariri, Y. Luo, F. Szidarovsky, and G. Qu, "Multi-level intrusion detection system (ml-ids)," in *2008 International Conference on Autonomic Computing*, IEEE, 2008, pp. 131–140.

[3] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267–280.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[5] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*, IEEE, 2014, pp. 247–255.

[6] W. Enck et al., "Taintdroid: An information-flow tracking system for real-time privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, pp. 1–29, 2014.

[7] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[8] I. J. Goodfellow et al., "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[9] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[12] S. Miller and C. Busby-Earle, "The role of machine learning in botnet detection," in *2016 11th international conference for internet technology and secured transactions (icitst)*, IEEE, 2016, pp. 359–364.

[13] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*, PMLR, 2017, pp. 1321–1330.

[14] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, et al., "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[15] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2018, pp. 228–234. DOI: 10.1109/ICAIBD.2018.8396200.

[16] E. U. A. for Cybersecurity, *ENISA threat landscape report 2018 – 15 top cyberthreats and trends*. European Network and Information Security Agency, 2019. DOI: doi/10.2824/622757.

[17] D. Sharma and A. Sharma, "Botnet and botnet detection techniques," *Academic Journal of Information Security*, vol. 1, no. 1, pp. 6–10, 2019.

[18] P. Wainwright and H. Kettani, "An analysis of botnet models," in *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, 2019, pp. 116–121.

[19] D. Wu, B. Fang, J. Wang, Q. Liu, and X. Cui, "Evading machine learning botnet detection models via deep reinforcement learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761337.

[20] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," *Advances in neural information processing systems*, vol. 33, pp. 14 927–14 937, 2020.

[21] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, and M. Colajanni, "Deep reinforcement adversarial learning against botnet evasion attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 1975–1987, 2020.

[22] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102 419, 2020.

[23] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," in *International Conference on Big Data Technologies and Applications*, Springer, 2020, pp. 117–135.

[24] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar, "Generative deep learning for internet of things network traffic generation," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2020, pp. 70–79. DOI: 10.1109/PRDC50213.2020.00018.

[25] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7327–7347, 2021.

[26] W. N. H. Ibrahim et al., "Multilayer framework for botnet detection using machine learning algorithms," *IEEE Access*, vol. 9, pp. 48 753–48 768, 2021.

[27] A. Murray and D. B. Rawat, "Towards botnet hazard analysis with generative adversarial networks for threat detection," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, SPIE, vol. 11746, 2021, pp. 201–212.

[28] R. H. Randhawa, N. Aslam, M. Alauthman, H. Rafiq, and F. Comeau, "Security hardening of botnet detectors using generative adversarial networks," *IEEE Access*, vol. 9, pp. 78 276–78 292, 2021. DOI: 10.1109/ACCESS.2021.3083421.

[29] K. Shinan, K. Alsubhi, A. Alzahrani, and M. U. Ashraf, "Machine learning-based botnet detection in software-defined network: A systematic review," *Symmetry*, vol. 13, no. 5, p. 866, 2021.

[30] K. Shinan, K. Alsubhi, A. Alzahrani, and M. U. Ashraf, "Machine learning-based botnet detection in software-defined network: A systematic review," *Symmetry*, vol. 13, no. 5, p. 866, 2021.

[31] I. Ullah and Q. H. Mahmoud, "A framework for anomaly detection in iot networks using conditional generative adversarial networks," *IEEE Access*, vol. 9, pp. 165 907–165 931, 2021. DOI: 10.1109/ACCESS.2021.3132127.

[32] Y. Xing, H. Shu, H. Zhao, D. Li, and L. Guo, "Survey on botnet detection techniques: Classification, methods, and evaluation," *Mathematical Problems in Engineering*, vol. 2021, no. 1, p. 6 640 499, 2021.

[33] N. S. Akash, S. Rouf, S. Jahan, A. Chowdhury, and J. Uddin, "Botnet detection in iot devices using random forest classifier with independent component analysis," *Journal of Information and Communication Technology*, vol. 21, no. 2, pp. 201–232, 2022.

[34] R. H. Randhawa, N. Aslam, M. Alauthman, and H. Rafiq, "Evasion generative adversarial network for low data regimes," *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 5, pp. 1076–1088, 2022.

[35] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile networks and applications*, pp. 1–14, 2022. [Online]. Available: https://doi.org/10.1007/s11036-021-01843-0.

[36] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile networks and applications*, vol. 27, no. 1, pp. 357–370, 2022.

[37] A. Al-Ameer, A. Asraa, and W. S. Bhaya, "Intelligent intrusion detection based on multi-model federated learning for software defined network.," *International Journal of Safety & Security Engineering*, vol. 13, no. 6, 2023.

[38] S. Gul, S. Arshad, S. M. Umar Saeed, A. Akram, B. Saeed, and M. A. Azam, "Improving botnet detection with a generative adversarial network-based technique," in *2023 20th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2023, pp. 564–569. DOI: 10.1109/IBCAST59916.2023.10713040.

[39] V. Kumar and D. Sinha, "Synthetic attack data generation model applying generative adversarial network for intrusion detection," *Computers Security*, vol. 125, p. 103 054, 2023, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2022.103054. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822004461.

[40] N. Peppes, T. Alexakis, K. Demestichas, and E. Adamopoulou, "A comparison study of generative adversarial network architectures for malicious cyber-attack data generation," *Applied Sciences*, vol. 13, no. 12, p. 7106, 2023.

[41] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Deep generative learning models for cloud intrusion detection systems," *IEEE Transactions on Cybernetics*, vol. 53, no. 1, pp. 565–577, 2023. DOI: 10.1109/TCYB.2022.3163811.

[42] K. Darnal, "A comparative analysis of machine learning algorithms in network-based intrusion detection systems for detecting advanced persistent threats to enhance cybersecurity," 2024.

[43] M. Al-Fawa'reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang, "Malbot-drl: Malware botnet detection using deep reinforcement learning in iot networks," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 9610–9629, 2024. DOI: 10.1109/JIOT.2023.3324053.

[44] V. Hernadi, *Improving network intrusion detection of minority classes using generative models*, 2024.

[45] J. Krupski, M. Iwanowski, and W. Graniszewski, "Extraction of minimal set of traffic features using ensemble of classifiers and rank aggregation for network intrusion detection systems," *Applied Sciences*, vol. 14, no. 16, p. 6995, 2024.

[46] H. H. Nguyen, C. N. Nguyen, X. T. Dao, Q. T. Duong, D. P. T. Kim, and M.-T. Pham, "Variational autoencoder for anomaly detection: A comparative study," *arXiv preprint arXiv:2408.13561*, 2024.

[47] R. H. Randhawa, N. Aslam, M. Alauthman, M. Khalid, and H. Rafiq, "Deep reinforcement learning based evasion generative adversarial network for botnet detection," *Future Generation Computer Systems*, vol. 150, pp. 294–302, 2024, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2023.09.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23003369.

[48] H. Vafaii, D. Galor, and J. Yates, "Poisson variational autoencoder," *Advances in Neural Information Processing Systems*, vol. 37, pp. 44 871–44 906, 2024.

[49] S.-B. Amokrane, D. Bujaković, B. Pavlović, M. Andrić, and T. Adli, "Enhancing intrusion detection system performance through feature selection," *Acta Polytechnica Hungarica*, vol. 22, no. 1, 2025.

[50] S.-B. Amokrane, D. Bujaković, B. Pavlović, M. Andrić, and T. Adli, "Enhancing intrusion detection system performance through feature selection," *Acta Polytechnica Hungarica*, vol. 22, no. 1, 2025.

[51] S. Feizi and H. Ghaffari, "Botnet detection and information leakage mitigation with differential privacy under generative adversarial networks," *Cluster Computing*, vol. 28, no. 2, p. 89, 2025.

[52] A. Suresh and A. Cyril Jose, "Adaptive network intrusion detection using reinforcement learning with proximal policy optimization," *ACM Transactions on Privacy and Security*, vol. 28, no. 4, pp. 1–24, 2025.