A Bayesian VAE Based Framework for Synthetic Data Generation and False-Alarm Reduction in Multi-Class Intrusion Detection Systems

by

M. Ridhwan Gani Bishal
21201529
Tasin Mohammad Yeasin
21201090
Mohammad Salah Akram Fuad
21201361
Raida Rizvee
21241032
Neamul Mueed
21201750

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

Department of Computer Science and Engineering
Brac University
October 2025

Declaration

It is hereby declared that

1. The thesis submitted is 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:

Raida Rizvee
21241032

M. Ridhwan Gani Bishal
21201529

Mohammad Salah Akram Fuad
21201361

Neamul Mueed
212410750

Tasin Mohammad Yeasin
21201090

Approval

The thesis titled "A Bayesian VAE Based Framework for Synthetic Data Generation and False-Alarm Reduction in Multi-Class Intrusion Detection Systems" submitted by

1. M. Ridhwan Gani Bishal(21201529)
2. Tasin Mohammad Yeasin(21201090)
3. Mohammad Salah Akram Fuad(21201361)
4. Raida Rizvee(21241032)
5. Neamul Mueed(21201750)

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

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Zayed Humayun
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam

Designation
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

Intrusion detection systems (IDS) are constantly evolving in the field of network security to safeguard critical data assets against a growing array of sophisticated cyber threats, such as malevolent botnets, massive Distributed Denial of Service (DDoS) attacks, slow-rate DDoS attacks, advanced persistent threats (APTs), and zero-day exploits. Moreover, any organization's network infrastructure remains vulnerable to different types of attacks, such as system abuse, security lapses, and break-ins. The Network Intrusion Detection System (NIDS) used in a network identifies such penetration attempts and intrusions. Researchers using deep learning (DL) have proposed increasingly capable IDS to protect critical networks; however, IDS are difficult to deploy in such environments because of high false-alarm rates (FAR). In this paper, we propose a hybrid framework that combines conditional variational autoencoder (CVAE)–based synthetic data generation with a Bayesian VAE model to reduce false-alarm rates in multi-class intrusion detection. This approach aims to lower FAR while maintaining strong detection performance by augmenting minority classes with class-consistent synthetic samples and leveraging calibrated Bayesian decisions.

Keywords: Network Intrusion Detection System (NIDS), False Alarm Rate (FAR), Bayesian Variational Autoencoder (BVAE-NF), Conditional Variational Autoencoder (CVAE), Synthetic Data Generation

Acknowledgement

We would like to express our deepest gratitude to our supervisor, **Dr. Muhammad Iqbal Hossain**, Associate Professor, Department of Computer Science and Engineering, Brac University, for his invaluable guidance, patience, and insightful feedback throughout the entire course of this research. His consistent encouragement and constructive criticism were instrumental in shaping both the direction and quality of this thesis.

We are especially thankful to our co-supervisor, **Mr. Zayed Humayun**, Lecturer, Department of Computer Science and Engineering, Brac University. His continuous mentorship, technical expertise, and hands-on support were crucial at every stage of our work — from model design and experimental refinement to documentation and presentation. His practical insights, meticulous attention to detail, and persistent motivation helped transform our ideas into a structured and meaningful research contribution.

Lastly, we would like to extend heartfelt gratitude to our peers, friends, and families for their encouragement, patience, and unwavering belief in us. Their support has been a constant source of strength throughout this journey.

— *The Authors*

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviations that will be later used within the body of the document.

AE Autoencoder

APT Advanced Persistent Threat

AUC Area Under the Curve

BNN Bayesian Neural Network

BVAE-NF Bayesian Variational Autoencoder with Normalizing Flows

C2BN Conditional Batch Normalization

CNN Convolutional Neural Network

CVAE Conditional Variational Autoencoder

CVAE-NIDS Conditional Variational Autoencoder based Network Intrusion Detection System

DCA Dendritic Cell Algorithm

DDoS Distributed Denial of Service

DL Deep Learning

ELBO Evidence Lower Bound

F1 Score Harmonic Mean of Precision and Recall

FAR False Alarm Rate

FL Federated Learning

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

G-Mean Geometric Mean of Sensitivity and Specificity

IDS Intrusion Detection System

KL Divergence Kullback–Leibler Divergence

LIBSVM Library for Support Vector Machines

LSTM Long Short-Term Memory

MC Sampling Monte Carlo Sampling

MLP Multi-Layer Perceptron

MTS Multivariate Time Series

NF Normalizing Flows

NGFW Next-Generation Firewall

NIDS Network Intrusion Detection System

PSO Particle Swarm Optimization

RF Random Forest

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

SNN Self-Normalizing Neural Network

SOC Security Operations Center

TN True Negative

TNR True Negative Rate

TP True Positive

TPR / DR True Positive Rate / Detection Rate

VAE Variational Autoencoder

WFL Weighted Federated Learning

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

Securing data transfer between networks is critical in today's digital age. The internet's exponential growth has resulted in an ever-increasing number of issues, making both users and their data vulnerable to a wide range of cyber threats. The likelihood of cyber attacks is considered to increase with the number of people who use the internet, making it critical to install strong security measures. To address this issue, IDS serves an important role in protecting vital computer networks from a variety of cyber threats, including credential stuffing, phishing, Distributed Denial of Service (DDoS), Man-in-the-Middle attacks, and many more[20]. An IDS can be deployed both in the form of hardware or software. It is intended to automate the process of detecting intrusion [2].It can automatically monitor and analyze network data for signals of suspicious behavior. A network intrusion detection system is used to find malicious activity within a network. It can identify a variety of outside-inflicted attacks and security breaches as well as insider-performed malpractices and abuses. Intruders can be broadly divided into three categories. A masquerader is typically an outsider who isn't permitted to use the system using a legitimate user account; a clandestine person can be either an insider or an outsider who tries to obtain supervisory access to the system. An insider is considered a misfeature when they abuse their privileges and gain the resources they are not authorized to.[6]

Network intrusion detection systems are of two types: Anomaly detection-based Network Intrusion Detection Systems (ADNIDS) and Signature-based Based Network Intrusion Detection Systems(SNIDS). SNIDS uses pattern matching on the features of the information it is aware of to raise an intrusion alarm. While ADNIDS works better in the case of novel or unknown patterns of attacks, SNIDS has a greater detection rate for known attack types. On the other hand, if there are any notable abnormalities from the typical traffic pattern in the user behavior that is being analyzed, ADNIDS sounds an intrusion alarm. However, because an intruder's behavior can vary, an ADNIDS tends to produce a high number of false alarms. Security breaches can be found by keeping an eye on the system audit record for any unusual patterns in the system's usage[6]. Over time, IDS technology has substantially increased its ability to identify cyber threats. However, the sophistication of attacks has improved, as well, posing new difficulties to network security[20].

Detecting new viruses used to be difficult, but with the introduction of DL classifiers, the area has changed dramatically. DL-based IDS often use GPUs and Deep

Neural Networks (DNN) for training and detecting network intrusions[8]. This system assumes that typical users and intruders behave differently. Traffic is classified into two types: benign and malicious, based on the probability of the packet. DL's extraordinary ability to understand both signature-based intrusion and fresh, unknown threats distinguishes it from traditional approaches [14]. The design of DL-based IDS often involves layers of neural networks capable of processing massive amounts of data and identifying patterns and abnormalities that may indicate an intrusion. This enhanced capacity detects zero-day attacks, which are previously unknown vulnerabilities that fraudsters exploit. Furthermore, DL algorithms may continuously learn and adapt to new data, improving the IDS's ability to detect emerging threats[8].

In general, network security has been completely transformed by the incorporation of deep learning into IDS, which now offers a potent weapon for thwarting increasingly complex cyberattacks. The integrity and confidentiality of digital information must be protected, and this requires constant research and improvement of IDS technologies. Cyber threats are always changing. Cyber threats are always evolving.

1.1 Problem Statement

Intrusion Detection Systems (IDS) promise early warning against malicious activity, yet their deployment in critical networks is constrained by a persistent and costly defect: false alarms. In practice, models misclassify benign traffic as malicious (false positives), while genuine attacks may slip past undetected (false negatives). The resulting False Alarm Rate (FAR)—often reported as the False Positive Rate (FPR)—does more than nudge a metric; it erodes analyst trust, wastes triage time, and can mask real incidents amid noise. Correcting this weakness is foundational to building an IDS that is both accurate and operationally usable.

At the core, IDS decisions fall into four outcomes: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). Performance is commonly summarized with FPR/FAR (share of benign flagged as attacks) and False Negative Rate (FNR) (missed attacks), alongside precision, recall, and overall accuracy. In security operations, these quantities are not interchangeable: a system that lowers FAR while silently raising FNR may look "better" on aggregate accuracy yet be riskier in practice. The goal is therefore simultaneously to suppress false alarms and preserve (or improve) true detection.

Modern network environments make this harder than it sounds:

- **Severe class imbalance and long tails:** Benign traffic dominates most datasets, while many attack families are infrequent or underrepresented. Models tend to overfit majority patterns and perform poorly on rare or minority attacks.
- **Distribution shift and drift:** Traffic characteristics and attack behaviors evolve across time, networks, and vendors. These changes invalidate static

training assumptions and degrade model performance when deployed in real-world environments.

- **Adversarial behavior:** Attackers actively adapt to detection signatures and decision boundaries, crafting traffic patterns that either evade detection or exploit model overconfidence.

Taken together, class imbalance, shifting traffic baselines, and adversarial adaptation create a brittle operating point: tightening thresholds to suppress FAR often raises FNR, while relaxing them to catch rare attacks floods analysts with noise.In critical networks the practical requirement is twofold: keep false alarms within manageable bounds and sustain high detection of evolving, low-frequency attacks, all under strict latency and throughput budgets. This is the crux of the problem: IDS must deliver dependable decisions in non-stationary, adversarial environments without leaning on optimistic assumptions from static benchmarks.

1.2 Research Objective

The purpose of this study is to develop and test new ways to reduce the FAR in IDS while maintaining or increasing system accuracy. This study aims to identify effective ways to reduce the misclassification of innocuous data as hazardous activity by thoroughly examining existing procedures and employing new AI techniques. The ultimate objective is to enhance the performance and reliability of IDS deployments in critical network infrastructures by addressing this fundamental problem. The other key objectives of this research are:

1. **Attack only representation, Class Balanced:** Train a pre-processing pipeline to prioritize only malicious traffic.Use of intelligent downsampling and CVAE-based oversampling to balance the sampling of different attack types. This makes sure that the model picks up discriminating features in all of the attack categories, but it is not discriminative against dominant classes.

2. **Synthetic Data Generation of High-Fidelity and Rare Attacks:** Train a Conditional Variational Autoencoder (CVAE) with Conditional Batch Normalization (C2BN) to produce realistic class-consistent synthetic examples of underrepresented attack classes, and thus perform better on rare but important threats.

3. **To design and integrate a hybrid Bayesian Variational Autoencoder (VAE)–Bayesian Neural Network (BNN)** framework capable of accurately detecting both novel and rare attacks while providing calibrated, uncertainty-aware predictions that minimize false alarm rates and enhance trustworthiness.

4. **High-Fidelity Multi-Class Detection via Uncertainty Rejection:** To obtain accurate multi-classification of individual network attacks with the help of the predictive uncertainty of the model. The essence behind this goal is to deploy uncertainty-based rejection mechanism which sieves out low-confidence predictions. This directly reduces false alarm and overall increases reliability and trustworthiness of detection system.

Chapter 2

Detailed Literature Review

Kim et al.(2017)[3] in their paper proposed an artificial intelligence-based method of detecting intrusion using DNN. In their investigation, they used the KDD Cup 99 dataset, of which 10% was used to train the model. During the preprocessing, the raw data from the dataset was normalized and transformed into a suitable format for input in the DNN model. They claimed that a high accuracy and detection rate was obtained that averaged 99%. Furthermore, the FAR was also very low, at 0.08%. Although they achieved excellent accuracy in classifying and analyzing individual traffic data, they recommended the use of long short-term memory (LSTM) and recurrent neural networks (RNN) for incorporating time series assessment in future studies.

In this paper, Mohamed et al.(2018)[5] aimed to develop an effective IDS that can both identify and mitigate malicious access to computer networks. Three machine learning algorithms such as Multi-Layer Perceptron (MLP), Random Forest (RF), and the Library for Support Vector Machine (LIBSVM) were used to divide the network activities into two categories: 'Attack' or 'Normal'. The framework is designed in a way to have a low false positive rate while detecting intrusion. In their paper, they described the preprocessing stage as the 'main stage' as it consisted of handling various aspects like handling the transformation of the dataset, dividing the dataset into various equal portions, scaling and normalizing the data, and getting rid of the redundant attributes. For conducting their research they used the widely available NSL-KDD dataset which has a total of 41 attributes. Using the dataset MLP achieved an accuracy of 95.7%, Random Forest achieved 99.6% accuracy, and LIBSVM achieved 94.8% accuracy. Next, using the Correlation Feature Selection (CFS) algorithm, only the relevant features in the dataset were selected. After reducing the features to 17 attributes, MPL achieved 91.7% accuracy, Random Forest achieved 99.6% accuracy, and LISBSV achieved 97.2% accuracy. They concluded that feature selection significantly enhances the accuracy of the framework and Random Forest performs the best among the three models in terms of accuracy, as well as, false positive rate.

Maiga et al.(2024) [20] proposed a way that converts regrouped network traffic into clusters based on their probabilities (calculated using the DL model) with the help of probabilistic clustering. Identification of high false alarm rate (H-FAR) clusters along with all the traffic falling within them are done in this process which is deemed

uncertain by the DL model as malicious for effective classification. Based on the results, human professors dissect further and provide a concluding decision. Apart from improving the performance by reducing the false alarms in the DL-based intrusion detection classifiers, the alleged model also includes NGFW, a next-generation firewall that will efficiently handle the traffic which will also ease the load on human experts. Using three public benchmark datasets (CICDDoS2019, UNSW-NB15, and CICID2017), a hybrid RNN model and a customized high-performance convolutional neural network (CNN) were used to continuously evaluate the suggested concept. Using simulations, results were evaluated which showed that the number of false negatives (FNs) and false positives (FPs) can be greatly decreased by up to 79.61% and 86.99%, respectively, with the combination of human expertise with DL technology. However, this study is conducted in a virtual environment where a Python script simulates the role of a human expert. The utilization of benchmark datasets allowed for the accurate classification of diverted traffic, simulating a situation in which experts might discern between normal and malicious traffic network traffic It is important to understand that differences in human judgment can have an impact on outcomes in practice and the expert efficiency and accuracy may not always be the same in real-world settings.

Gurung et al.(2019) [6] in their paper present a deep-learning IDS that employs logistic regression and a sparse AE to solve the network intrusion problem. The model is trained using a sparse AE with a sparsity constraint. In the classification step, logistic regression is used to obtain a single binary classification of either a normal or an intruder user. This model, which produces a greater accuracy rate than SignatureBased Intrusion Detection techniques with a lower risk of False Negatives and False Positives, is tested and trained using the NSL-KDD dataset. Although no evidence of its application in a real-world environment exists, this method works best for servers that monitor networks in real-time.

In their research, Alkahtani et al.(2021) [9] target was to help safeguard IoT structures and devices from different types of malicious attacks by proposing a research target to build an intelligent model. For classification and feature selection, their model contained LSTM, modified CNN, and Particle Swarm Optimization (PSO). This system used the IoTID20 dataset but making an operable strong framework was a big challenge to overcome. To obtain the right features the PSO algorithm used velocity and position to find the right pathway, which gave rise to dimensionality deduction.

Moreover, for the classification of network attacks from the dataset, three types of artificial intelligence (AI) models were used, which include CNN-LSTM, CNN, and LSTM. Using the confusion matrix, the occurrence of true negative, false negative, true positive,e, and false positive was recorded. When dealing with gradient explosion data, the RNN encountered difficulties but it was able to properly predict the temporary training dataset. Hence LSTM was introduced to solve this issue. The memory function of LSTM replaced the existing RNN unit. The CNN that was used in this project contained five different types of functionality layers which include polling, convolution, input, output, and FC. 70% of the data were used to train the model and the rest were used for testing. For intrusion detection LSTM, CNN,

and CNN-LSTM were used in three different types of experiments. According to the results, LSTM obtained a better accuracy compared with the other two types of algorithms. In the assessment for classification of normal and attacks in the network activity, LSTM performed better than the other algorithms, having 98.84% precision, 99.60% sensitivity, 77.72% specificity, 99.00% F1-score, and 98.82% accuracy. The overall results obtained by the three proposed algorithms are LSTM achieving 99.82% accuracy, CNN achieving 96.60% accuracy,y, and CNN-LSTM achieving 98.80% accuracy. From the above results, this Deep Learning (DL) based framework of the IDS can properly handle and detect real attacks which can enhance the security of devices in the IoT environments.

Using Federated Learning, Neto et al.(2022)[13] focused on multi-tenant IoT environments by proposing a combined DDoS classification and detection. The main part covered by this paper is the federated learning method based on IoT multi-regional edge nodes, a method that enables the spread of knowledge about DDos in IoT multi-tenant environments, an approach that enhances classification capabilities without the exchange of raw information in federated learning. To resolve this DDoS issue in this type of environment, mainly two types of algorithms were used. The first algorithm consisted of Scheduled Collaborative Training that focused on detecting and recording DDoS attacks without leaking any sensitive information and the second algorithm consisted of On-demand model provision.The dataset used in this experiment was CICDDoS2019 which helped to simulate DDoS attacks in this environment. In this simulated scenario, six tenants were used, and their data descriptions. Every tenant experienced various types of attacks which are different from each other. In a neural network with nodes, Adam algorithms were used for optimizations, learning rate with a fixed value, and finally having a Local scaling batch size is operated using the standard approach. This approach preserved tenants' privacy while effectively learning DDoS classification, having an accuracy of 84.8%. For the models trained by 3 and 5, their accuracy was almost 81% in the best-case scenario. Having a different node model of neural network has achieved a convergence reaching 84.2% accuracy which outperformed the models with higher node models. For future work, the results can be improved by analyzing internal parameters such as the learning rate for multiple tenants. Also, the deployment-related proposal to the Security Operations Center (SOC) can be a way to improve the overall datasets and federal learning. Aggregation algorithms such as FedAvg can be used and a different strategy can be implemented to improve the overall accuracy of this experiment.

To improve the outcomes given in this study, next possibilities include optimizing and analyzing internal factors (e.g., learning rate) for numerous renters. Future work will address difficulties associated with deploying the idea described in this research in a Security Operations Centre (SOC). FedAvg is a popular aggregation method; however, different strategies can be utilized, and this is something that future research should look into.

Aldhaheri et al.(2020)[7] designed a unique network-based detection method for IoT assaults. The proposed technique combines Dendritic Cell technique (DCA) with Self-Normalizing Neural Networks (SNNs) and uses the BoT-IoT dataset to clas-

sify IoT intrusion which reduces false alarms. The technique proposed employs the Self-Normalizing Neural Network (SNN) to categorize features as either safe or risky signals, automating and smoothing the feature extraction and categorization stages to enhance classification performance. Compared to other models such as KNN, NB, MLP, and SVM, the suggested DeepDCA model is a better classifier. It also exhibited a high detection rate for IoT threats, as well as a high accuracy rate and a low false positive rate. Furthermore, in identifying IoT risks, our model showed a low False-Positive rate with higher accuracy. Since SNN is a relatively new technology, additional study is needed to see whether it can be applied to tackle intrusion detection problems in addition to other deep learning problem domains.

Rosay et al.(2021)[10] proposed a multilayer perceptron for connection identification to prevent cyberattacks in-car IoT networks. This model describes the IDS solution using the CIC-IDS2017 dataset and then verifies and tests the proposed multilayer perceptron model on the CSE-CIC-IDS2018 dataset, demonstrating the advantages of employing DL approaches in NIDS. Pre-processing involves removing imbalances from the initial dataset before categorizing it into three parts: 25% training, 25% validation, and 25% testing. Following feature selection, the data is normalized using Z-normalization to remove extraneous characteristics. Accuracy of the suggested model outperforms traditional ML models while creating significantly lower false positive rates according to the experimental data. Using the TensorFlow Framework, the model is created and then trained in Python as the DL model. To facilitate embedded system testing, this model was also implemented in a real-world system on a chip car. Several flaws were detected, including underrepresentation of different assaults and inaccurately estimated properties in the model's two datasets. However, because the design was tested in real-world scenarios, it could be simply adapted for application in the automotive sector.

The rising frequency and sophistication of cyber threats targeting IoT networks has led to a great deal of research on the detection of LR-DDoS assaults in software-defined networking (SDN) systems. To identify DDoS assaults of both low and large volumes, Dehkordi et al. suggested a three-layer-based technique that makes use of entropy and classification approaches. They were able to achieve a detection rate of 97.65% for small-volume DDoS attacks under a 30-second time window. Similarly, other studies have also adopted machine learning techniques such as support vector machines (SVM), with parameters optimized using genetic algorithms (GA) and kernel principal component analysis (KPCA), resulting in detection accuracies up to 98.03%. Further advancements in this domain include the use of ant colony optimization (ACO) for traffic profiling, which enhances detection accuracy while minimizing false positive rates. Additionally, entropy-based methods in SDN-cloud environments have demonstrated high detection rates of 98.2%, with minimal false positives. Integrating federated learning (FL) into LR-DDoS detection frameworks marks a significant improvement. Existing FL-based models, such as the chained anomaly detection system, have shown promising results but lack optimal global model performance due to average preference assignment mechanisms. The proposed Weighted Federated Learning (WFL) approach addresses this by assigning preferences based on model accuracy, leading to improved detection rates and lower misclassification rates. This strategy not only utilizes the benefits of federated learn-

ing but also secures data privacy and minimizes network and storage overheads. [14]

Cybercriminals increasingly target the Internet of Things (IoT) due to its rapid growth and integration into essential services. To address this, a novel Deep Learning-based IDS has been developed, featuring a four-layer connected network architecture. This IDS is designed to detect various attacks such as Blackhole, DDoS, Sinkhole, Opportunistic Service, and Wormhole, achieving an average detection accuracy of 93.74%. The system operates independently of communication protocols, simplifying its deployment across different IoT environments. Performance metrics include a precision of 93.71%, recall of 93.82%, and an F1-score of 93.47%, indicating robust and reliable detection capabilities. This system aims to enhance the security and reliability of IoT networks by providing real-time intrusion detection and minimizing the potential for service interruptions and financial losses.[15]

Lopez-Martin et al. (2017)[4] proposed a NIDS that relies on a conditional VAE. The algorithm's architecture is designed to manipulate the decoder layers by combining the intrusion labels it. In this paper the NSL-KDD dataset is used as it contains a large variety of samples. They claimed that their model is less complex than other unsupervised methods, as well as it successfully generates better classification results. Using their model they obtained an F1 score, accuracy, and recall of 0.79, 0.80, and 0.80, respectively. In their findings, they demonstrated that when conditional VAE was compared with other classifiers like Random Forest, Linear SVC, and Multilayer Perceptron, it outperformed the other algorithms. Furthermore, their method can perform feature reconstruction. If some features are absent in a training dataset, it can recover those features with a high accuracy rate.

Toğaçar (2022)[12], proposed the use of probabilistic BNNs and Bayes neural networks as deep learning models for identifying DDoS attacks. The author mentioned that probabilistic BNNs help in addressing various issues of deep learning models like confidence intervals and overfitting. It also allows the uncertainties, that affect the performance, to be calculated. The botnet-IoT dataset and the UNSWNB-15 dataset were used for the findings of this paper. For the first investigation, both models were tested using the botnet-IoT dataset. Both of them achieved an overall accuracy of 100%. For the second investigation, probabilistic BNN was used on the UNSWNB-15 dataset. Approximately 99.9% accuracy was achieved in the experiment.

To enhance the performance of anomaly-based Intrusion Detection Systems (IDS), especially in reducing false alarm rates, Amokrane(2025) proposed a comprehensive methodology that integrates correlation analysis and feature selection using the NF-UQ-NIDS-v2 dataset. Their approach begins by performing correlation analysis to eliminate highly correlated features, thereby minimizing redundancy. Subsequently, Recursive Feature Elimination (RFE) is applied to identify the most informative features for classification. Using only eight selected features, the system achieved an accuracy of 98.13%, a recall of 98.23%, and an AUC of 99.73%. Most notably, the false alarm rate was reduced by 53.73%, dropping to just 0.3589%, while scoring time improved by 34.21%, making the IDS not only more accurate but also computationally efficient. [22]

Chapter 3

Dataset Description

3.1 Description of the Data

The dataset used for our study is the NF-UQ-NIDS-v2 dataset, which is a comprehensive NetFlow-based dataset specially designed for evaluating network intrusion detection systems (NIDS). It was constructed by merging and transforming four benchmark datasets NSW-NB15, BoT-IoT, CSE-CIC-IDS2018, and ToN-IoT—by re-extracting flow-level metadata into a standard 43-feature NetFlow v9 schema [11]. The full dataset contains 75,987,976 flows spanning 20 attack types, with about 33% benign and 67% attack traffic. This dataset features 43 network flow attributes, including traffic volume, protocol behavior, and packet timing characteristics. Therefore, the richness and real-world relevance of this dataset make it an ideal benchmark for validating intrusion detection techniques in heterogeneous network environments.

Table 3.1: NF-UQ-NIDS-v2 distribution with class descriptions.

Class	**Count**	**Description**
Benign	25,165,295	Normal, unmalicious flows. These are regular, non-attacking traffic within the network.
DDoS	21,748,351	Distributed Denial of Service: An attempt to overload a computer system's resources by using multiple distributed sources.
Reconnaissance	2,633,778	A technique for gathering information about a network host, also known as probing or scanning.
Injection	684,897	A variety of attacks that provide untrusted inputs to alter the execution of a system, such as SQL and code injections.
DoS	17,875,585	Denial of Service: An attempt to overload a system's resources to prevent access to data or services.
Brute Force	123,982	A technique used to obtain usernames and password credentials by trying a large number of possible combinations.
Password	1,153,323	Attacks aimed at retrieving passwords through various methods, such as brute force or sniffing.
XSS	2,455,020	Cross-site Scripting: A vulnerability in web applications where malicious scripts are injected into websites, affecting users.
Infiltration	116,361	An inside attack where malicious files are sent via email to exploit an application, followed by backdoor installation to further scan the network for vulnerabilities.
Exploits	31,551	Attacks exploiting specific vulnerabilities in software or systems to gain unauthorized access or control.
Theft	2,431	Attacks aimed at obtaining sensitive data, such as personal or financial information, often use methods like data theft or keylogging.
Scanning	3,781,419	The process of systematically searching for vulnerabilities within a network by probing ports or services.
Fuzzers	22,310	A technique where random data is sent to a system to identify potential vulnerabilities by causing unexpected behavior.
Backdoor	18,978	A method of bypassing normal authentication to gain unauthorized access, often used by attackers for persistent access.
Bot	143,097	Malicious software that allows attackers to remotely control infected computers to perform tasks like sending spam or launching attacks.
Generic	16,560	Attacks that use general techniques not specific to a single application or vulnerability often affect multiple systems.
Analysis	2,299	Techniques or activities that analyze a system or network, often during the initial stages of an attack.
Shellcode	1,427	A small piece of code used as a payload in exploits, typically to open a backdoor or gain control over a system.
MITM	7,723	Man-In-The-Middle: An attack where the attacker intercepts and potentially alters communication between two parties without their knowledge.
Worms	164	Self-replicating malware that spreads to other systems without requiring human intervention.
Ransomware	3,425	Malicious software that encrypts the victim's data and demands a ransom for the decryption key.

3.2 Data Preprocessing

To enable efficient model training and evaluation while meeting the computational demands of the large-scale NF-UQ-NIDS-v2 dataset (over 75 million records), we preprocessed the entire dataset without creating a prior subset. This retained all original flows benign plus 20 attack types for initial processing. The pipeline, implemented in Python with Pandas, was designed to enhance data quality, promote model generalization, and reduce feature dimensionality.

Preprocessing Steps

1. **Removal of IP Addresses.** Source and destination IP addresses (e.g., `IPV4_SRC_ADDR`, `IPV4_DST_ADDR`) were dropped to avoid network-specific identifiers that harm transferability. Because IPs are tied to the data-collection context and rarely reflect generalizable intrusion patterns, removing them improves portability across environments.

2. **Handling of Port Numbers.** Naively dropping source and destination ports (e.g., `L4_SRC_PORT`, `L4_DST_PORT`) led to many duplicate records in subsets: ports are part of the 5-tuple and act as key differentiators. Their removal caused distinct flows with overlapping statistics (e.g., duration, packet counts, byte volumes) to collapse into identical feature vectors. To preserve semantic information while controlling cardinality, ports were *bucketed* and then encoded categorically according to IANA conventions using a custom bucketing function that also catches edge cases (non-integers or out-of-range values). The buckets are:

Table 3.2: Port-number categorization used for feature encoding.

Category	**Range**	**Notes**
Well-known	0–1023	Reserved for common services (e.g., HTTP, DNS).
Registered	1024–49151	Assigned to user or vendor processes.
Dynamic/Private	49152–65535	Ephemeral ports typically used for client-side connections.
Invalid	otherwise	Non-integer or out-of-range values; encoded as a separate category.

3. **Filtering by Attack Type.** Attack types with insufficient samples for robust training/evaluation were removed. The *Worms* class, already scarce in the full corpus, further decreased after preprocessing and no longer provided adequate support for meaningful analysis or classification. To mitigate risks from underrepresented classes (overfitting, poor generalization, unreliable metrics on rare events), *Worms* was excluded. The final dataset retains **19 attack types** alongside **Benign**. Shown inTable 3.3

Table 3.3: Data Count After Preprocessing

Attack	**Count**
Benign	9,364,416
XSS	1,097,263
DoS	770,085
DDoS	407,022
Password	209,816
Injection	180,923
Infiltration	76,535
Reconnaissance	34,803
Scanning	32,272
Exploits	23,987
Brute Force	17,021
Fuzzers	8,690
MITM	7,662
Generic	4,625
Backdoor	857
Theft	812
Analysis	768
Shellcode	641
Ransomware	560
Bot	472

3.3 Dataset Imbalance

A fundamental challenge in designing effective Network Intrusion Detection Systems (IDS) is the severe *class imbalance* inherent in network traffic data, which is prominently featured in the NF-UQ-NIDS-v2 dataset used in this study. The dataset mirrors real-world traffic where benign flows dominate: the **Benign** class contains over 9364416 instances and thus forms the majority class.
In stark contrast, many critical attack categories are severely underrepresented. For example, **Bot**, **ransomware**, and **Shellcode** contain only 472, 560, and 641 samples, respectively. This yields an extreme skew in which **Benign** outweighs the smallest minority class (**Bot**) by a ratio of approximately

$$\frac{\text{Benign}}{\text{Bot}} \approx \frac{9{,}364{,}416}{472} \approx \mathbf{1.98 \times 10^4 : 1}.$$

Consequently, the majority class vastly outnumbers the minority attack classes, posing a significant obstacle for training an unbiased and effective IDS model.
This imbalance has direct implications for machine learning:

Bias toward majority: Models optimized for overall accuracy often learn to predict the majority class, achieving superficially high accuracy while ignoring rare attacks.

Poor generalization to attacks: The scarcity of minority-class examples hampers the learning of discriminative patterns, increasing *false negatives* (malicious traffic misclassified as benign).

Metric distortion: Accuracy becomes misleading; minority-aware metrics (e.g., macro-F1, balanced accuracy, AUROC/PR-AUC) are essential for a trustworthy evaluation.

These leads to poor generalization and a high number of false negatives, where malicious activities are incorrectly classified as normal traffic. [1]

The consequences of this bias are directly reflected in performance metrics. While overall accuracy may appear high, a closer look at class-specific metrics reveals the true deficiency. For minority attack classes, the model will typically exhibit low recall (the ability to identify all relevant instances of an attack) and a poor F1-score (the harmonic mean of precision and recall). In the context of cybersecurity, this is a critical failure. The primary objective of an IDS is to detect threats; a low recall for attack classes means the system is failing at its core function, leaving the network vulnerable to undetected intrusions. The real-world implication is a false sense of security, where sophisticated, low-volume attacks can go unnoticed until significant damage has been done. Addressing this data imbalance is therefore not merely a technical optimization but a critical step toward building a reliable and effective IDS. [1]

3.4 CVAE-Based Dynamic Balancing Framework

To tackle the challenges above, we propose a custom, hybrid data balancing framework that leverages a generative deep learning model: a Conditional Variational Autoencoder (CVAE). Our technique dynamically resamples the dataset on a per-class basis, combining oversampling for minority classes with a novel, quality-based downsampling strategy for majority classes.

The core of this framework is a specifically designed CVAE that incorporates Conditional Batch Normalization (C2BNVAE). This model is trained to learn the underlying probability distribution of each specific class in the dataset. It can then be used to generate new, high-fidelity synthetic samples for underrepresented attack classes or to identify the most representative samples within an overrepresented class.

3.4.1 Rationale for Approach

The choice of a generative model like a CVAE, over traditional balancing techniques such as programmatic oversampling and random downsampling, is deliberate.

1. **High-Fidelity Data Generation:** CVAEs can capture the complex, non-linear relationships within the feature space of the data, producing synthetic samples that are novel, diverse, and highly representative of the original data's distribution for a given class—rather than mere linear interpolations. This is crucial for IDS datasets where attack features can be subtle and complex.

2. **Conditionality for Class-Specific Learning:** By conditioning the model on class labels, it is ensured that the generated data for a specific attack type adheres strictly to the characteristics of that attack. Incorporating Conditional Batch Normalisation further enables class-specific feature scaling, yielding distinct and accurate generation.

3. **Intelligent Downsampling:** Randomly discarding samples from the majority class can lead to the loss of valuable information and potentially remove borderline examples that are useful for defining the decision boundary. Our approach uses the CVAE's reconstruction error as a proxy for how well a sample conforms to the learned data distribution. By retaining only the samples with the lowest error, we intelligently preserve the most representative and least ambiguous instances of the majority class, effectively cleaning the dataset while reducing its size.

This hybrid approach brings all classes to a target equilibrium,mitigating model bias and enabling training of a more robust and equitable IDS classifier.

3.5 Methodology and Implementation

The proposed balancing technique is applied as a critical preprocessing stage prior to training of the main IDS classifier. The methodology is executed in a dynamic, class-by-class workflow.

3.5.1 Conditional Batch Normalization (C2BN)

The key innovation in our architecture is a custom ConditionalBatchNormalization layer. Standard batch normalization normalizes a layer's inputs and learns two affine transformation parameters, γ (gamma) and β (beta), to rescale and recenter the normalized output. In our `C2BN` layer, γ and β are not global learnable parameters; instead, they are dynamically generated for each batch based on the class condition c.

The operation is as follows:

1. The input feature map $\mathbf{x}$ is first normalized using standard batch normalization but *without* learning the affine parameters:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\mathrm{Var}[\mathbf{x}] + \epsilon_{\mathrm{bn}}}} \tag{3.1}$$

2. The one-hot encoded class vector $\mathbf{c}$ is passed through two separate `Dense` layers to predict the conditioning parameters:

$$\gamma_{\mathrm{pred}} = \mathrm{Dense}_{\gamma}(\mathbf{c}) \qquad \text{and} \qquad \beta_{\mathrm{pred}} = \mathrm{Dense}_{\beta}(\mathbf{c}) \tag{3.2}$$

3. The final output of the layer is a class-conditional affine transformation of the normalized input:

$$\mathbf{y}_{\mathrm{out}} = \gamma_{\mathrm{pred}} \odot \hat{\mathbf{x}} + \beta_{\mathrm{pred}} \tag{3.3}$$

This forces the model to learn class-specific feature scales and shifts, significantly improving its ability to distinguish between and accurately model the unique data distributions of different attack types.

3.5.2 Loss Function

The C2BNVAE is trained by maximizing the Evidence Lower Bound (ELBO), which is equivalent to minimizing a composite loss function $\mathcal{L}_{\mathrm{Total}}$. This loss consists of two terms: a reconstruction loss and a regularization term based on the Kullback–Leibler (KL) divergence:

$$\mathcal{L}_{\mathrm{Total}} = \mathcal{L}_{\mathrm{Reconstruction}} + \beta\, \mathcal{L}_{\mathrm{KL}} \tag{3.4}$$

- **Reconstruction Loss ($\mathcal{L}_{\mathbf{Reconstruction}}$).** This term measures the difference between the original input data $\mathbf{x}$ and the decoder's reconstructed output $\hat{\mathbf{x}}$. In our implementation, we use the Mean Squared Error (MSE):

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \big(x_i - \hat{x}_i\big)^2 \tag{3.5}$$

  This encourages the decoder to generate outputs that are perceptually similar to the original inputs.

- **KL Divergence ($\mathcal{L}_{\mathbf{KL}}$).** This term acts as a regularizer on the latent space. It measures the divergence between the distribution learned by the encoder, $q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{c})$, and a prior distribution, typically a standard normal $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$D_{\mathrm{KL}}\big(q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \,\big\|\, p(\mathbf{z})\big) \tag{3.6}$$

  Minimizing this term encourages the encoder to produce latent distributions close to the prior, resulting in a more regular and continuous latent space.

- **Beta (β).** This is a hyperparameter (set to `1e-5` in our code) that balances the weight between the reconstruction loss and the KL divergence. A small β prioritizes accurate reconstruction, which is crucial for our goal of generating high-fidelity data.

3.6 The Dynamic Balancing Algorithm

The balancing process is controlled by a predefined 'TARGET_COUNT' of 60,000 samples, specifying the desired per-class size in the final dataset. The algorithm then iterates over each unique class and applies one of three actions:

1. **Oversampling (for Minority Classes).** If a class has fewer samples than `TARGET_COUNT`, a dedicated C2BNVAE is trained exclusively on that class. After training, the encoder maps all real samples to the latent space; the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of this latent distribution are computed. New latent vectors are sampled as

$$\mathbf{z}_{\text{new}} \sim \mathcal{N}\big(\boldsymbol{\mu},\, \mathrm{diag}\big(\boldsymbol{\sigma}^2\big)\big) \tag{3.7}$$

   and, together with the class label, are passed to the trained decoder to generate synthetic samples until `TARGET_COUNT` is reached.

2. **Downsampling (for Majority Classes).** If a class has more samples than `TARGET_COUNT`, a C2BNVAE is trained on that class and used to reconstruct each sample. The mean squared error (MSE) reconstruction loss for sample i is computed as

$$\mathrm{MSE}_i = \frac{1}{d} \left\| \mathbf{x}_i - \hat{\mathbf{x}}_i \right\|_2^2 \tag{3.8}$$

   where d is the feature dimension. Samples are ranked by MSE_i, and only the `TARGET_COUNT` samples with the *lowest* errors (most representative) are retained.

3. **No Action.** If a class already has exactly `TARGET_COUNT` samples, it is kept unchanged in the final dataset.

3.7 Synthetic Sample Validation

To safeguard the quality and integrity of the generated data, we added a validation step in the oversampling process. After an initial batch of synthetic samples is

generated, they are passed back through the trained encoder-decoder pipeline. The reconstruction error for these new samples is calculated. Any synthetic sample with a reconstruction error exceeding a predefined threshold is discarded. This filter blocks low-quality or out-of-distribution samples, ensuring the augmented dataset remains coherent.

Table 3.4: After balancing the dataset.

Attack	**Count**
Analysis	60,000
Backdoor	60,000
Benign	60,000
Bot	60,000
DDoS	60,000
DoS	60,000
Exploits	60,000
Fuzzers	60,000
Reconnaissance	60,000
Infilteration	60,000
injection	60,000
Shellcode	60,000
scanning	60,000
xss	60,000
ransomware	60,000
password	60,000
Theft	57,879
Brute Force	47,444
Generic	17,478
mitm	10,156

The balanced dataset is shown in table 3.4

3.8 Latent Space Representation Learning

After balancing the dataset, the next crucial step is to train a model capable of learning a rich, low-dimensional representation of the network traffic data. For this purpose, a Conditional Variational Autoencoder (CVAE) was developed and trained on the complete, dynamically balanced dataset.

The primary objective of this CVAE is to train a powerful encoder model. This encoder is tasked with mapping high-dimensional input feature vectors into a compressed, regularized latent space. A well-structured latent space is essential for subsequent classification tasks, as it disentangles the underlying factors of variation in the data, making class distinctions more apparent.

To build a robust model, we used the following advanced training techniques:

1. **KL Annealing:** The weight of the Kullback–Leibler (KL) divergence term in the loss function was gradually increased during training. This strategy

encourages the model to first prioritize accurate data reconstruction before enforcing a smooth, Gaussian-like structure on the latent space, thereby preventing posterior collapse and yielding more meaningful representations.

2. **Stratified Validation:** The dataset was split with stratification so the validation set mirrors the training set, providing a more reliable estimate of generalization. This ensures that the class distribution in the validation set mirrors that of the training set, providing a more reliable and accurate measure of the model's generalization performance.

3. **Dynamic Learning Rate and Early Stopping:** The training process incorporated `ReduceLROnPlateau` and `EarlyStopping` callbacks that automatically adjust the learning rate based on validation loss and halt the training when performance ceases to improve, which prevents overfitting and optimizes training time.

The final output of this stage is a highly trained encoder model, capable of transforming any input network traffic sample into a compact latent vector that captures its most salient features.

3.9 Visualization and Analysis of Latent Space with t-SNE

To qualitatively assess the efficacy of the trained CVAE encoder, a visualization of the learned latent space was performed using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm. t-SNE is a non-linear dimensionality reduction technique particularly well-suited for visualizing the structure of high-dimensional datasets in a low-dimensional space, such as a 2D plot.

The visualization was generated from a representative, balanced subset of the augmented dataset. Specifically, 10,000 samples were randomly selected from each class. This design ensured that the visualization was not biased by class size and remained computationally feasible. The procedure involved passing these selected samples through the trained CVAE encoder to obtain their latent vectors, which were then processed by the t-SNE algorithm to compute a two-dimensional embedding for each data point.

The left plot on the figure 3.1 visualizes the latent space with each point colored according to its class label. The analysis reveals several key findings:

1. **Strong Cluster Formation:** The visualization shows that the CVAE has learned a highly effective latent space representation. Data points belonging to the same class form distinct and dense clusters. Major classes such as *Benign* (red), *DDoS* (light blue), and *scanning* (light green) are clearly separated into well-defined groups on the periphery of the plot.

2. **Inter-Class Separability:** The clear separation between most clusters provides strong qualitative evidence that the encoder has successfully learned to distinguish the unique features of different attack types.

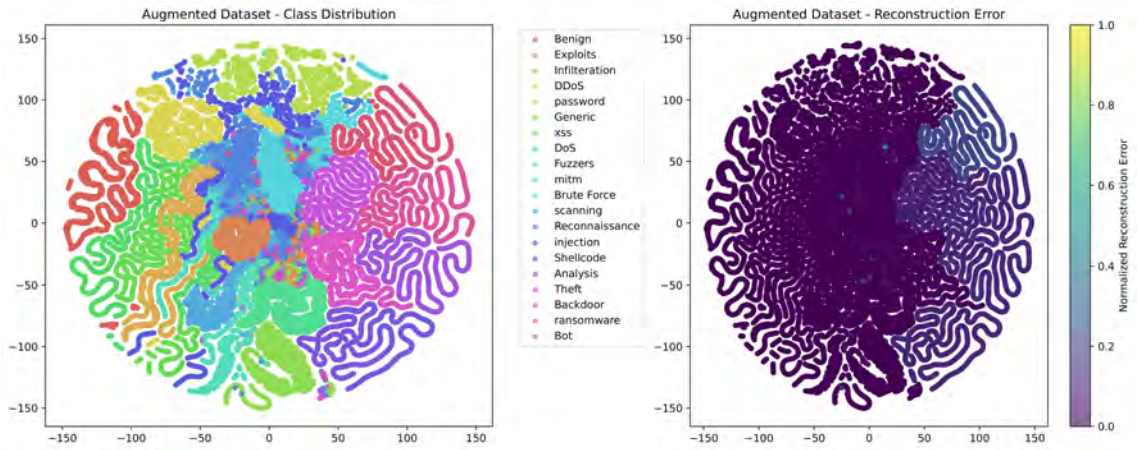

Figure 3.1: Class Distribution in t-SNE Latent Space (*Left*) and Reconstruction Error in t-SNE Latent Space (*Right*).

3. **Regional Overlap:** While separation is generally strong, there is a more densely populated central region where several smaller classes exhibit some overlap (e.g., *password*, *xss*, *Generic*). This suggests that these attack types may share more subtle or similar underlying features, making them inherently more difficult to separate even in the learned latent space.

The right plot on the figure 3.1displays the same t-SNE embedding, but this time the points are colored based on their normalized reconstruction error from the CVAE. Darker purple indicates a low error, while bright yellow indicates a high error.

Error Distribution: A clear pattern emerges where the large, well-defined peripheral clusters (*Benign*, *DDoS*, *scanning*) exhibit predominantly low reconstruction error (darker colors).

Interpretation: This suggests that the CVAE was able to model and reconstruct these more common or structurally consistent traffic types with high fidelity. Conversely, the central, more intermingled region shows a higher concentration of points with higher reconstruction errors (brighter greens and yellows). This is a significant finding, as it implies that the attack types in this region may be more complex, exhibit greater internal diversity, or contain anomalies that were more challenging for the autoencoder to learn and accurately reproduce.

In summary, the t-SNE analysis provides powerful visual validation for the feature learning capabilities of the CVAE. It confirms that the model has learned a meaningful and largely separable latent space, while also offering deeper insights into the relative complexity and relationships between the different classes of network traffic.

Chapter 4

Proposed Model

4.1 A Hybrid Bayesian VAE with Normalizing Flows for Network Intrusion Detection

In this work, we introduce a sophisticated deep learning architecture designed specifically for the complex task of network intrusion detection. We propose a **Bayesian Variational Autoencoder with Normalizing Flows (BVAE-NF)** augmented with a Bayesian classifier head for multi-class intrusion detection. The design jointly learns an expressive latent representation of network traffic and produces class probabilities with explicit epistemic uncertainty via Bayesian weight sampling (Monte-Carlo averaging), enabling calibrated decision-making objectives.

Learning expressive representations: The model compresses high-dimensional traffic features into a flexible, low-dimensional latent space, encouraging discovery of prominent structure in the data.

Quantifying uncertainty: By placing probability distributions over network weights, the model explicitly quantifies uncertainty and yields better-calibrated predictive probabilities. These uncertainty estimates support risk-aware decision making, highlighting ambiguous or novel traffic for further examination and reducing overconfident errors in highly unpredictable settings.

4.2 Model architecture

4.2.1 Notation and Dimensionality

We use the following notation. Let $\mathbf{B}$ denote the mini- batch size, $\mathbf{D}$ the input dimension, $\mathbf{L}$ the latent dimension, $\mathbf{C}$ the number of classes, and $\mathbf{K}$ the number of planar normalizing flow transforms.

- **Input data:** A sample is a vector $\mathbf{x} \in \mathbb{R}^D$ with label $y \in \{1, \ldots, C\}$ (the corresponding integer encoded true label is $\mathbf{e}_y \in \{0, 1\}^C$).
- **Latent variable:** The compressed representation is $\mathbf{z} \in \mathbb{R}^L$.
- **Tensor shapes (per batch):**

- Input: $\mathbf{x} \in \mathbb{R}^{B \times D}$
- Encoder outputs (mean and log variance): $\boldsymbol{\mu}(\mathbf{x}), \log \boldsymbol{\sigma}^2(\mathbf{x}) \in \mathbb{R}^{B \times L}$
- Latent samples: $\mathbf{z}_0, \mathbf{z}_K \in \mathbb{R}^{B \times L}$ (after K flows)
- Decoder output: $\hat{\mathbf{x}} \in \mathbb{R}^{B \times D}$
- Classifier logits: $\boldsymbol{\eta} \in \mathbb{R}^{B \times C}$
- Probabilities: $\boldsymbol{\pi} = \text{softmax}(\boldsymbol{\eta}) \in \mathbb{R}^{B \times C}$

4.2.2 Architectural Overview

The model's data flow is organized as a three stage pipeline aimed at strong feature learning and uncertainty aware classification.

Probabilistic Encoding: The process begins with a Bayesian encoder that serves as a learned compressor. Given a high-dimensional input vector $\mathbf{x}$, it produces the parameters—a mean $\boldsymbol{\mu}(\mathbf{x})$ and a log variance $\log \boldsymbol{\sigma}^2(\mathbf{x})$ that define a simple Gaussian base distribution specific to that input.

Latent Space Refinement: A sample $\mathbf{z}_0$ is drawn from this base distribution and then passed through a sequence of K planar normalizing flows. The innovation of the generative architecture is as follows: the flows progressively transform the simple Gaussian into a much more complex and flexible distribution, enabling the model to capture intricate, non-Gaussian structure found in real-world network data. The result is a refined latent sample $\mathbf{z}_K$.

Dual Head Processing: The refined, information-rich latent variable $\mathbf{z}_K$ is then fed to two parallel networks:

- **Bayesian Decoder:** A Bayesian decoder reconstructs the input using $\hat{\mathbf{x}} = g_{\boldsymbol{\theta}}(\mathbf{z}_K)$, employing BayesLinear layers that place probability distributions over the decoder weights. Consequently, a KL divergence term for the decoder's weights is included in the total objective function, thereby incorporating epistemic uncertainty into the model's generative path.
- **Bayesian Classifier:** A Bayesian classifier consumes $\mathbf{z}_K$ to perform the final intrusion detection task. Employing Bayesian layers at this stage is crucial, as it allows the model to represent its uncertainty in the ultimate decision.

This complete three stage pipeline is visually summarized in the architectural diagram below. The figure 4.1 illustrates the data flow from the initial input x through the latent space refinement to the final, parallel working decoder and classifier heads.

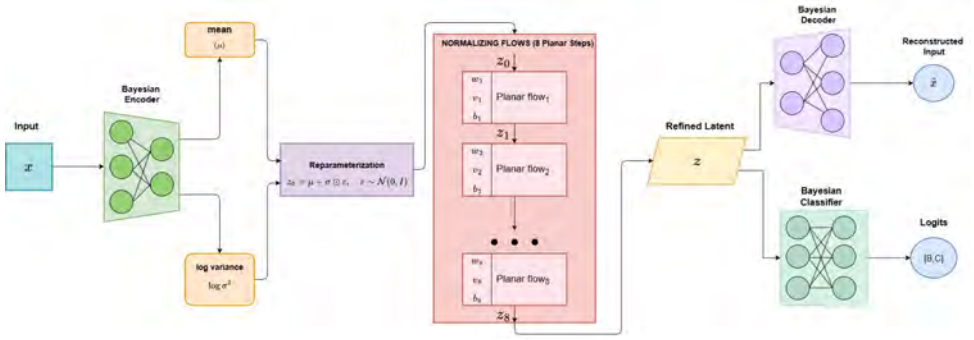

Figure 4.1: Model Architecture of the Bayesian Variational Autoencoder with Normalizing Flows (BVAE-NF) and dual-head Bayesian processing.

The figure 4.1 illustrates the BVAE-NF data flow. A Bayesian encoder produces the parameters of a base latent distribution, a stack of planar normalizing flows refines the latent sample, the refined latent drives a deterministic decoder for reconstructing the input and a Bayesian classifier for intrusion detection.

4.3 Core Components

The proposed model architecture consists of four primary components working in a streamline: a Bayesian encoder that creates an initial probabilistic representation, a series of Normalizing Flows that refine this representation and two parallel heads, a Bayesian decoder and a Bayesian classifier, that use this refined representation for reconstruction and detection, respectively.

4.3.1 The Bayesian Encoder and Latent Space

The process begins with the **Bayesian Encoder**, which maps a high-dimensional input vector $\mathbf{x}$ to a probabilistic representation in a lower-dimensional latent space. Instead of outputting a single point, the encoder defines parameters for a simple base posterior distribution $q_0(\mathbf{z}_0 \mid \mathbf{x})$, assumed to be a diagonal Gaussian. For numerical stability and to ensure the variance is always positive, the encoder outputs the mean $\boldsymbol{\mu}(\mathbf{x})$ and the log variance $\log \boldsymbol{\sigma}^2(\mathbf{x})$.

$$q_0(\mathbf{z}_0 \mid \mathbf{x}) = \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{x}),\, \mathrm{diag}\big(\boldsymbol{\sigma}^2(\mathbf{x})\big)\big) \tag{4.1}$$

To enable training via backpropagation, a sample $\mathbf{z}_0$ is drawn from this distribution using the *reparameterization trick*. This technique separates the deterministic parts of the operation (scaling and shifting) from the stochastic element $\boldsymbol{\epsilon}$:

$$\mathbf{z}_0 = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \odot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_L) \tag{4.2}$$

The encoder's layers are Bayesian, meaning their weights $\mathbf{W}^{\text{enc}}$ are not fixed values but are themselves probability distributions. The model learns an optimal posterior distribution $q(\mathbf{W}^{\text{enc}})$ for these weights, regularized by a predefined prior $p(\mathbf{W}^{\text{enc}})$.

4.3.2 Normalizing Flows for an Expressive Posterior

A key limitation of a standard Variational Autoencoder (VAE) is that the simple Gaussian assumption for the posterior can be too restrictive to capture the complex structure of real-world data. To address this, we transform the initial latent sample $\mathbf{z}_0$ into a more complex sample $\mathbf{z}_K$ using a series of $K = 8$ *Normalizing Flows*.
We employ *planar flows*, where each flow acts as an invertible "stretch-and-fold" operation on the latent space. The transformation is defined as:

$$\mathbf{z}_k = \mathbf{z}_{k-1} + \hat{\mathbf{u}}_k \, \tanh\big(\mathbf{w}_k^\top \mathbf{z}_{k-1} + b_k\big) \tag{4.3}$$

When a probability distribution is transformed, its density must be corrected. This is achieved by calculating the **log-determinant of the Jacobian** for the transformation, which measures the change in volume of the space.

$$\log \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right| = \log \left| 1 + \hat{\mathbf{u}}_k^\top \, \boldsymbol{\psi}_k(\mathbf{z}_{k-1}) \right|, \quad \text{where} \quad \boldsymbol{\psi}_k(\mathbf{z}) = \big(1 - \tanh^2\big(\mathbf{w}_k^\top \mathbf{z} + b_k\big)\big) \, \mathbf{w}_k \tag{4.4}$$

For the planar flow to be invertible, a crucial constraint is applied to its parameter $\mathbf{u}_k$. We reparameterize it to $\hat{\mathbf{u}}_k$ using the following expression, which ensures the transformation is always valid:

$$\hat{\mathbf{u}}_k = \mathbf{u}_k + \frac{-1 + \text{softplus}\big(\mathbf{w}_k^\top \mathbf{u}_k\big) - \mathbf{w}_k^\top \mathbf{u}_k}{\|\mathbf{w}_k\|_2^2} \, \mathbf{w}_k \tag{4.5}$$

By chaining these eight transformations ($\mathbf{z}_8 = f_8 \circ \cdots \circ f_1(\mathbf{z}_0)$), we construct a highly flexible and expressive posterior distribution capable of modeling complex data.

4.3.3 The Bayesian Decoder (Generative Path)

The **Bayesian Decoder** is responsible for the model's generative path. Its function is to reconstruct the original input $\mathbf{x}$ from the refined latent variable $\mathbf{z}_K$. This reconstruction task ensures that the latent space retains essential information about the input data.

$$\hat{\mathbf{x}} = g_{\boldsymbol{\theta}}\big(\mathbf{z}_K;\, \mathbf{W}^{\text{dec}}\big) \tag{4.6}$$

Like the encoder, the decoder is Bayesian, with its weights $\mathbf{W}^{\text{dec}}$ drawn from learned posterior distributions $q(\mathbf{W}^{\text{dec}})$. This approach introduces epistemic uncertainty into the reconstruction process and contributes a KL divergence term for the decoder's weights to the model's total loss function.

$$p\big(\mathbf{W}^{\text{dec}}\big) = \mathcal{N}\big(\mathbf{0},\, \sigma_0^2 \mathbf{I}\big), \quad q\big(\mathbf{W}^{\text{dec}}\big) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbf{W}^{\text{dec}}},\, \text{diag}\big(\boldsymbol{\sigma}^2_{\mathbf{W}^{\text{dec}}}\big)\big) \tag{4.7}$$

The decoder's output is formally modeled as a likelihood $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_K, \mathbf{W}^{\text{dec}})$, which is chosen to match the input data type (e.g., Gaussian for continuous features, Bernoulli for binary data). This formal likelihood enables robust reconstruction of the input data.

4.3.4 The Bayesian Classifier (Detection Head)

Running in parallel to the decoder, the **Bayesian Classifier** acts as the detection head, performing the primary task of multi-class intrusion detection. It maps the refined latent variable $\mathbf{z}_K$ to a vector of logits $\boldsymbol{\eta}$ via Bayesian linear layers, which are then converted to class probabilities $\boldsymbol{\pi}$.

$$\boldsymbol{\eta} = f_{\text{clf}}\big(\mathbf{z}_K;\, \mathbf{W}^{\text{clf}}\big)\,, \quad \boldsymbol{\pi} = \text{softmax}(\boldsymbol{\eta}) \tag{4.8}$$

Because the model's weights are treated as probability distributions, the decision boundary itself carries the parameter uncertainty. At inference time, we estimate the predictive distribution using Monte Carlo aggregation. On each of the M passes, we resample both the classifier weights $\mathbf{W}^{\text{clf},(m)} \sim q(\mathbf{W}^{\text{clf}})$ and the latent variable $\mathbf{z}_K^{(m)} \sim q_K(\mathbf{z} \mid \mathbf{x})$, then average the resulting softmax outputs.

$$\hat{\boldsymbol{\pi}}(\mathbf{x}) \approx \frac{1}{M} \sum_{m=1}^{M} \text{softmax}\Big(f_{\text{clf}}\Big(\mathbf{z}_K^{(m)};\, \mathbf{W}^{\text{clf},(m)}\Big)\Big) \tag{4.9}$$

This process yields well-calibrated probabilities and enables powerful uncertainty summaries, such as predictive entropy. During training, this head contributes the standard cross-entropy term to the loss, along with its own weight KL divergence to the global Bayesian Neural Network (BNN) regularizer, thereby enabling risk-aware operation by flagging highly uncertain traffic instances.

4.4 The Hybrid Objective Function

The model is trained end-to-end by minimizing a composite objective function $\mathcal{L}_{\text{total}}$. This function is carefully designed to balance four distinct goals simultaneously: accurate data reconstruction, precise multi-class classification, and regularization of both the model's weights and the latent space. The total loss is a weighted sum of four individual components.

4.4.1 Reconstruction Loss($\mathcal{L}_{\text{recon}}$)

This term ensures that the latent space is meaningful by penalizing the model for poor reconstructions. It measures the dissimilarity between the original input vector $\mathbf{x}$ and the decoder's output $\hat{\mathbf{x}}$. We use the Mean Squared Error (MSE), which is equivalent to the negative log-likelihood assuming a Gaussian distribution for the data.

$$\mathcal{L}_{\text{recon}} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2 \tag{4.10}$$

4.4.2 Classification Loss ($\mathcal{L}_{\text{class}}$)

This is the primary task-oriented loss that drives the model's performance as an intrusion detection system. It is the standard multi-class cross-entropy between the predicted probabilities (from the classifier's logits) and the integer-coded true labels y.

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log \pi_{i,c} \tag{4.11}$$

4.4.3 BNN Weight KL Divergence ($\mathcal{L}_{\text{KL-BNN}}$)

This is a crucial regularization term for all Bayesian layers in the model (encoder, decoder, and classifier). The Kullback–Leibler (KL) divergence measures the difference between the learned posterior distribution of the weights $q(\mathbf{W})$ and their predefined prior $p(\mathbf{W})$. Minimizing this term prevents the model from becoming overconfident and overfitting to the training data.

$$\mathcal{L}_{\text{KL-BNN}} = \sum_{\ell \in \text{Layers}} \text{KL}(q(\mathbf{W}_\ell) \,\|\, p(\mathbf{W}_\ell)) \tag{4.12}$$

4.4.4 Latent Space KL Divergence ($\mathcal{L}_{\text{KL-Latent}}$)

This term regularizes the structure of the latent space. It encourages the complex posterior distribution $q_k(\mathbf{z} \mid \mathbf{x})$, which results from the normalizing flows, to remain close to a simple standard normal prior $p(\mathbf{z})$. This is the standard VAE KL divergence term, modified to account for the change in probability density caused by the flows (the $\sum \log|\det J_K|$ term).

$$\mathcal{L}_{\text{KL-Latent}} = \mathbb{E}_{\mathbf{z}_0 \sim q_0}\left[\log q_0(\mathbf{z}_0 \mid \mathbf{x}) - \sum_{k=1}^{K} \log|\det J_k| - \log p(\mathbf{z}_K)\right] \tag{4.13}$$

4.4.5 Total Loss

The final objective function combines these four components. The KL divergence terms are scaled by weighting hyperparameters, β_1 and β_2, which control the strength of the regularization.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{class}} + \beta_1 \mathcal{L}_{\text{KL-BNN}} + \beta_2 \mathcal{L}_{\text{KL-Latent}} \tag{4.14}$$

4.5 Training and Inference Procedure

This section details the practical implementation of the model's training, regularization and evaluation. Moving from theoretical architecture to its practical execution. All procedures were implemented using the PyTorch environment.

4.5.1 Hyperparameters and Optimization

The model's architecture is defined by several key hyperparameters: a latent space dimension of $L = 32$, a hidden layer dimension of **128**, and a sequence of $K = 8$ planar flows. The model was trained for a maximum of **70 epochs** with a **batch size of 256**.
All Bayesian layers in the architecture were initialized with a Gaussian prior over their weights $\mathbf{W}$, defined as:

$$\mathbf{W} \sim \mathcal{N}\big(\mathbf{0}, \sigma_0^2 \mathbf{I}\big), \quad \text{where } \sigma_0 = 0.1 \tag{4.15}$$

The model's parameters were optimized using the **Adam optimizer** with an initial learning rate of 1×10^{-3}. To dynamically adjust this rate, a **ReduceLROnPlateau** scheduler was employed. This scheduler monitors the **validation loss** and reduces the learning rate by a factor of 0.5 if the loss fails to improve for **3 consecutive epochs**. Moreover, to prevent overfitting and reduce unnecessary computation, an early stopping mechanism was used with a **patience of 7**. To ensure the selection of the optimal model, training concludes after seven consecutive epochs without improvement in **validation accuracy**, at which point the model from the **best checkpoint** is restored for the final evaluation.

4.5.2 Regularization and Stability Techniques

Training a deep probabilistic model requires specific techniques to ensure stability and prevent common issues. The following methods were implemented to regularize the model and guarantee a smooth training process:

- **KL Annealing:** To prevent "posterior collapse," the weight of the latent KL divergence term (β_2) was annealed. The weight was kept at 0 for the first 10 epochs and then linearly ramped up to its maximum value of 1×10^{-5} for the remainder of the training. In contrast, the BNN weight KL divergence term (β_1) was kept at a fixed value of 1×10^{-5} throughout.
- **BNN Regularization Scope:** The BNN weight KL divergence (`BKLLoss`) sums the KL loss over all Bayesian layers in the encoder, decoder, and classifier, as the `last_layer_only` flag was set to `False` in its implementation.
- **Numerical Stability:** Several techniques were used to ensure numerical stability. The encoder's log-variance output ($\log \sigma^2$) was clamped to a maximum value of 10.0 to avoid overflow, and the global norm of the gradients was clipped to a maximum value of 1.0. Stability within the normalizing flows was enforced by applying the mathematical invertibility constraint (the $\hat{\mathbf{u}}$ reparameterization) and by adding a small epsilon (1×10^{-8}) inside the logarithm of the Jacobian determinant to prevent $\log(0)$.

4.6 Algorithm

The complete and full algorithm of the model is formally presented in the algorithm 1.

4.6.1 Inference and Uncertainty Quantifications

A key benefit of the Bayesian architecture is the ability to quantify predictive uncertainty, which is achieved using a **Monte Carlo (MC) inference** procedure at evaluation time. For each input sample, we perform multiple stochastic forward passes, resampling both a new latent variable $z_K^{(m)}$ and a new set of weights $W^{(m)}$ on each pass.

A minor distinction exists between this procedure during mid-training validation and the final evaluation. For computational efficiency within the training loop, validation accuracy is calculated by averaging the raw logits from $M = 15$ passes. For the final, more robust evaluation on the test set, we average the softmax probabilities from $M = 50$ passes. This latter method is used to derive the final predictive distribution, $\hat{\pi}(x)$, as shown in the equation below:

$$\hat{\pi}(x) \approx \frac{1}{M} \sum_{m=1}^{M} \text{softmax}\big(f_{\text{clf}}(z_K^{(m)}; W_{\text{clf}}^{(m)})\big) \tag{4.16}$$

The model's uncertainty for a given prediction is then calculated as the **predictive entropy** of this final, aggregated probability distribution. A higher entropy indicates greater model uncertainty:

$$H(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi}_c \log \hat{\pi}_c \tag{4.17}$$

Algorithm 1 Training Hybrid BNN–VAE–NF

Require: Dataset D; input dim d_{in}; hidden dims H; latent dim L; classes C; flow steps N_{flows}; epochs E; batch size B; lr α; BNN weight α_{bnn}; max KL weight $\beta_{\max}$; KL start E_{start}; patience P; MC samples K_{pred}

$(X, y) \leftarrow \textsc{LoadFeaturesAndLabels}(D)$; $y \leftarrow \textsc{EncodeLabels}(y)$
$X \leftarrow \textsc{StandardizeFeatures}(X)$
$(X_{\text{tr}}, X_{\text{val}}, X_{\text{te}}, y_{\text{tr}}, y_{\text{val}}, y_{\text{te}}) \leftarrow \textsc{SplitData}(X, y)$
model $\leftarrow \textsc{InitializeBNN\_VAE\_NF}(d_{\text{in}}, H, L, C, N_{\text{flows}})$
opt $\leftarrow \textsc{Adam}(\text{model.parameters}, \alpha)$; sch $\leftarrow \textsc{ReduceLROnPlateau}(\text{opt})$
best_acc $\leftarrow 0$; no_improve $\leftarrow 0$; $M_{\text{best}} \leftarrow \textsc{DeepCopy}(\text{model.state\_dict}())$
for $e = 0$ to $E - 1$ **do**
    model.train()
    for (X_b, y_b) in $\textsc{DataLoader}(X_{\text{tr}}, y_{\text{tr}}, B)$ **do**
        $h \leftarrow \text{ReLU}(\textsc{Encoder\_BNN1}(X_b))$; $h \leftarrow \text{ReLU}(\textsc{Encoder\_BNN2}(h))$
        $\mu, \text{logvar} \leftarrow \textsc{Linear}(h), \textsc{Linear}(h)$
        $\text{logvar} \leftarrow \textsc{Clamp}(\text{logvar}, -10, 10)$
        $\varepsilon \sim \mathcal{N}(0, I)$; $z_0 \leftarrow \mu + \varepsilon \odot \exp(0.5 \cdot \text{logvar})$
        $z \leftarrow z_0$; $\log\det J \leftarrow 0$
        for $k = 1$ to N_{flows} **do**
            $\log\det J \leftarrow \log\det J + \textsc{log\_det\_jac}(\text{flow}_k, z)$
            $z \leftarrow \textsc{flow}_k(z)$
        end for
        $\hat{X} \leftarrow \textsc{Decoder\_BNN}(z)$; logits $\leftarrow \textsc{Classifier\_BNN}(z)$
        $L_{\text{recon}} \leftarrow \text{MSE}(\hat{X}, X_b)$ ▷ mean reduction
        $L_{\text{class}} \leftarrow \textsc{CrossEntropy}(\text{logits}, y_b)$
        $L_{\text{bnn}} \leftarrow \alpha_{\text{bnn}} \cdot \text{KL\_BNN}(\text{model})$
        $\log q(z_0) \leftarrow -\frac{1}{2}\sum_i \left[\text{logvar}_i + (z_{0,i} - \mu_i)^2 \exp(-\text{logvar}_i)\right]$
        $\log p(z) \leftarrow -\frac{1}{2}\sum_i z_i^2$
        $L_{\text{lat}} \leftarrow \mathbb{E}_{\text{batch}}\left[\log q(z_0) - \log\det J - \log p(z)\right]$
        $\beta \leftarrow \beta_{\max} \cdot \min\left(1, \max\left(0, \frac{e - E_{\text{start}}}{E - E_{\text{start}}}\right)\right)$ ▷ smooth ramp
        $L_{\text{total}} \leftarrow L_{\text{recon}} + L_{\text{class}} + L_{\text{bnn}} + \beta \cdot L_{\text{lat}}$
        $\textsc{ZeroGrad}(\text{opt})$; $\textsc{Backward}(L_{\text{total}})$; $\textsc{ClipGradNorm}(\text{model}, 1.0)$; $\textsc{Step}(\text{opt})$
    end for
    model.eval()
    $(\text{val\_acc}, \text{val\_loss}) \leftarrow \textsc{Evaluate\_K}(\text{model}, X_{\text{val}}, y_{\text{val}}, K_{\text{pred}})$ ▷ MC average
    $\textsc{Step}(\text{sch}, \text{val\_loss})$
    if val_acc > best_acc **then**
        best_acc $\leftarrow$ val_acc; $M_{\text{best}} \leftarrow \textsc{DeepCopy}(\text{model.state\_dict}())$; no_improve $\leftarrow 0$
    else
        no_improve $\leftarrow$ no_improve $+ 1$
    end if
    if no_improve $\geq P$ **then**
        break
    end if
end for
$\textsc{LoadStateDict}(\text{model}, M_{\text{best}})$
test_metrics $\leftarrow \textsc{Evaluate\_K}(\text{model}, X_{\text{te}}, y_{\text{te}}, K_{\text{pred}})$
return model, test_metrics

Chapter 5

Result, Analysis and Discussion

5.1 Learning Curves

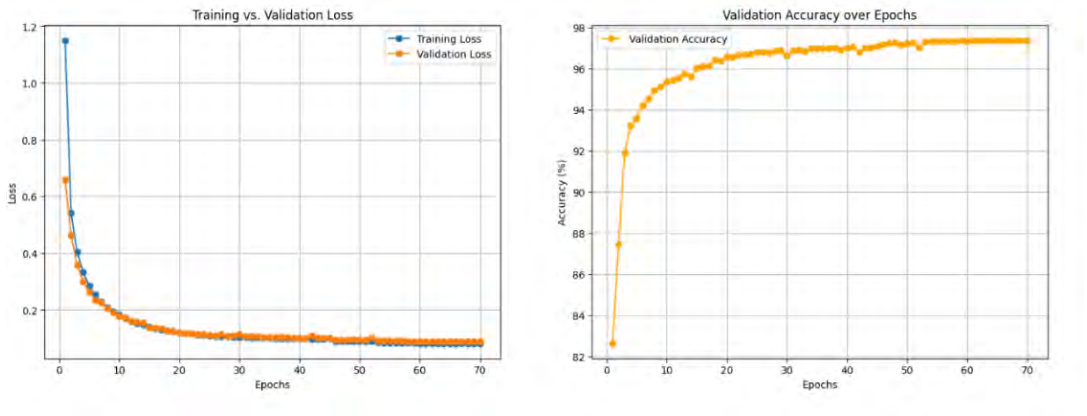

Figure 5.1: Learning curves: (a) Training vs. validation loss; (b) Validation accuracy over epochs.

How the validation accuracy was obtained: Training ran for up to 70 epochs with learning-rate decay and early stopping on the best validation score. The validation accuracy climbed rapidly in the first few epochs (from $\approx 82.7\%$ at epoch 1 to $\approx 93\%$ by epoch 4), then improved steadily in smaller steps under the decaying learning rate, reaching $\approx 97\%$ by the mid-to-late epochs and stabilising near 97.4% at the end of training. This shape fast early gains followed by a long, shallow tail is typical of well-regularised probabilistic models that continue to refine decision boundaries and probability calibration late in training.

Why we did not train indefinitely: While additional epochs could produce marginal improvements, the training/validation loss gap remains small and stable throughout, indicating limited overfitting pressure. Moreover, the model checkpoint used for reporting is the one with the *best* validation accuracy (saved when a new high score is observed), not necessarily the last epoch. This balances exploration time with generalisation quality.

Validation vs accepted-set accuracy: The 97.4% validation accuracy refers to performance over *all* test samples without abstention. When we apply entropy-

based rejection , the model defers only the most uncertain $\approx 5\%$ of cases. On the remaining *accepted set*, accuracy rises to $\approx 99\%$, reflecting that the abstention policy removes borderline cases where a cautious system should consult an analyst rather than commit to an automated decision.

Observations: Training loss drops steeply in the first 5–10 epochs and then decays smoothly; validation loss tracks it with a small, stable gap and shows no late-epoch divergence. Validation accuracy rises from $\approx 82.7\%$ (epoch 1) to $\approx 93\%$ (epoch ~ 4), then gradually approaches $\approx 97.5\%$ by ~ 70 epochs; a long tail consistent with ongoing probability calibration.

What is optimized: Training minimizes a hybrid objective that balances reconstruction quality, classification accuracy, Bayesian weight regularization, and latent-space regularization (via normalizing flows):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{class}} + \alpha_{\text{bnn}}\mathcal{L}_{\text{KL-BNN}} + \beta_t\mathcal{L}_{\text{KL-latent}}. \tag{5.1}$$

Here α_{bnn} is a small constant weighting the Bayesian weight-KL, while β_t is an *annealed* coefficient that starts near zero and grows over epochs—preventing early over-regularization of the latent space and explaining the clean convergence in figure 5.1.

Latent sampling and planar flow: Sampling uses the reparameterization trick; each flow step adds a volume-preserving transform with a closed-form Jacobian log-determinant. (4.2) (4.3) (4.4)

How validation accuracy is computed: The Bayesian head produces stochastic logits $\ell^{(m)}$ across M forward passes. We average softmax probabilities and take the argmax to score accuracy:

$$\hat{\pi}(x) = \frac{1}{M}\sum_{m=1}^{M} \text{softmax}\big(\ell^{(m)}(x)\big), \tag{5.2}$$

$$\text{ValAcc}(\%) = 100 \cdot \frac{\#\text{correct}}{\#\text{total}}. \tag{5.3}$$

5.2 Classification Performance (Accepted Predictions)

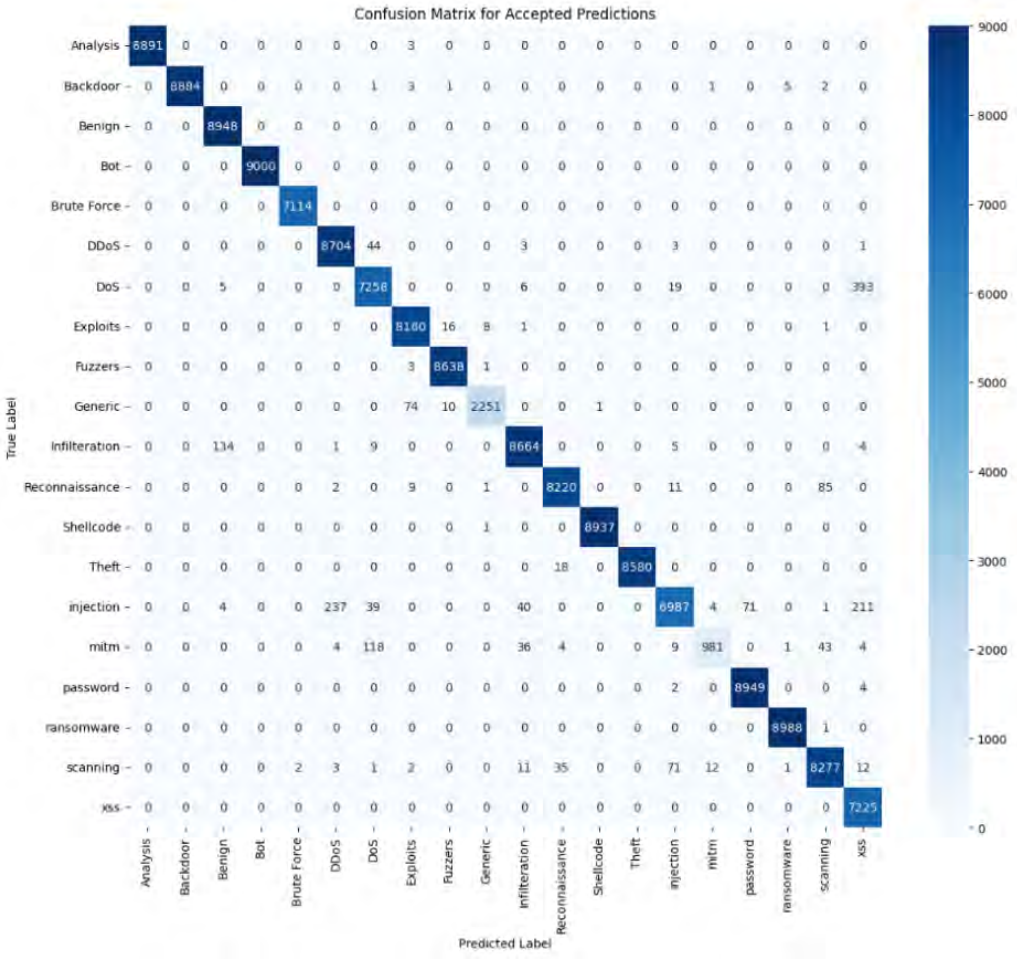

Figure 5.2: Confusion matrix on accepted predictions after entropy-based rejection (threshold 0.5142).

Observations: The accepted-set confusion matrix is strongly diagonal, with sparse, localized off-diagonal mass among intuitively similar families (e.g., DoS/DDoS/Generic, Injection/XSS). This reflects the effect of abstaining on high-uncertainty cases, which removes borderline decisions and reveals the underlying separability.

How the matrix is built: For each test example we compute $\hat{\pi}(x)$ as in equation 5.2 and the predictive entropy

$$H(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi}_c \log \hat{\pi}_c. \tag{5.4}$$

Only predictions with $H(\hat{\pi}(x)) \leq \tau$ are *accepted*. Let $\mathcal{A} = \{x : \ H(\hat{\pi}(x)) \leq \tau\}$ with $\tau = 0.5142$. Entries are

$$C_{i,j} = \#\{x \in \mathcal{A} : \ y = i, \ \hat{y} = j\}, \tag{5.5}$$

from which per-class Precision/Recall/F1 can be derived:

$$\text{Precision}_j = \frac{C_{j,j}}{\sum_i C_{i,j}}, \quad \text{Recall}_j = \frac{C_{j,j}}{\sum_k C_{j,k}}, \quad F_{1,j} = \frac{2\,\text{Precision}_j\,\text{Recall}_j}{\text{Precision}_j + \text{Recall}_j}. \tag{5.6}$$

5.3 ROC and AUC Analysis

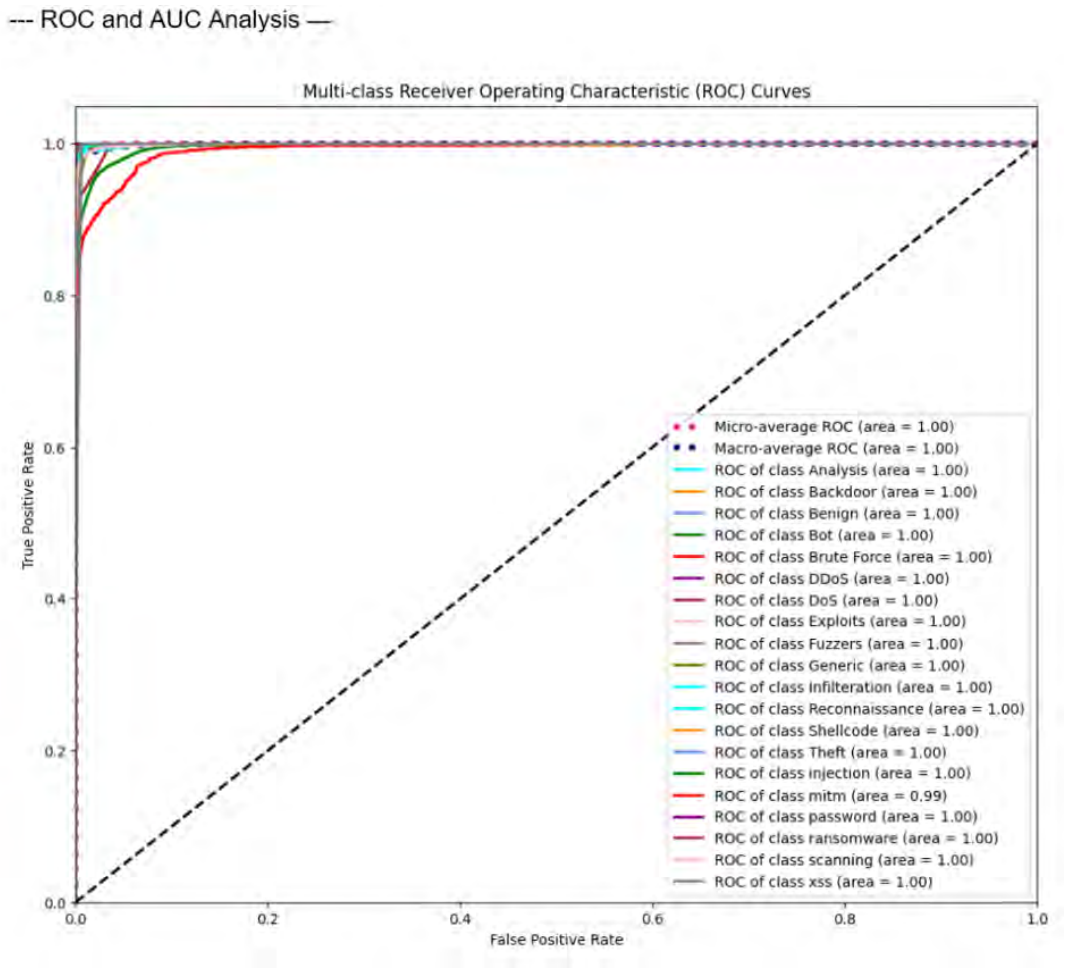

Figure 5.3: Multi-class ROC curves (one-vs-rest). Macro-average AUC ≈ 0.9991; micro-average ≈ 1.0.

Observations: ROC curves cluster near the upper-left, demonstrating high TPR at low FPR across classes. Per-class AUC values are near-perfect, with long-tail classes (e.g., *mitm*, *injection*) still above 0.992 on this split.

How ROC is computed: For class c we use the score $s_c(x) = \hat{\pi}_c(x)$ and sweep a threshold γ to obtain

$$\text{TPR}_c(\gamma) = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{FPR}_c(\gamma) = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{5.7}$$

Trapezoidal integration yields AUC_c. Macro- and Micro-AUC are

$$\text{Macro-AUC} = \frac{1}{C}\sum_{c=1}^{C} \text{AUC}_c, \qquad \text{Micro-AUC} = \text{AUC}_{\text{pooled}}. \tag{5.8}$$

5.4 Uncertainty-Aware Rejection (Bayesian Abstention)

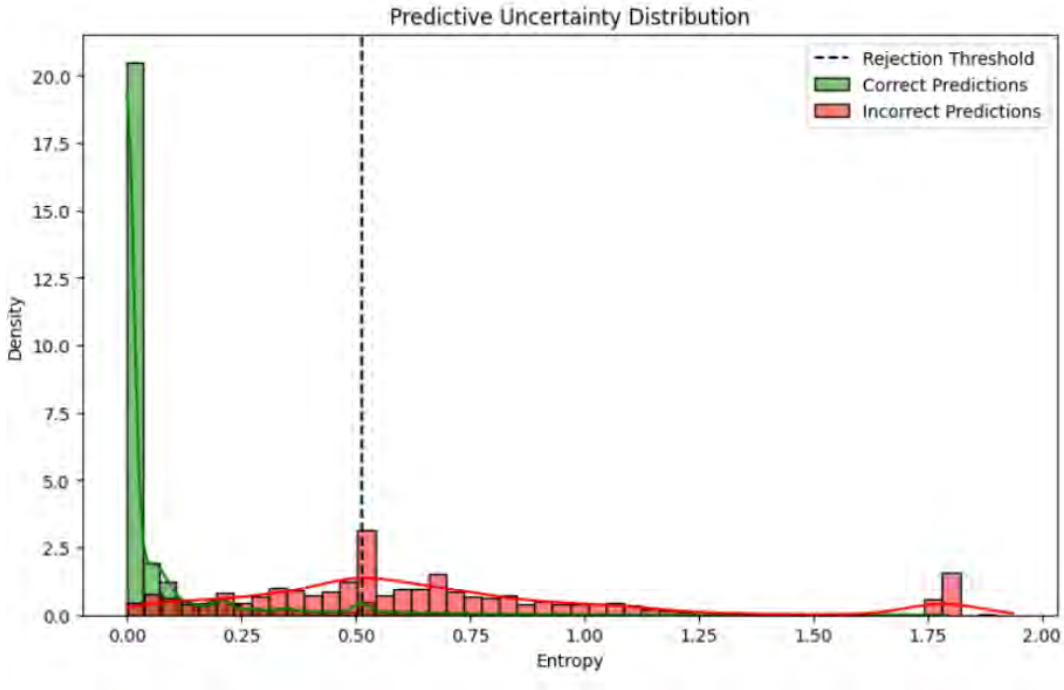

Figure 5.4: Predictive-uncertainty distribution with rejection threshold at 0.5142 (dashed).

Definition: The Bayesian head yields a predictive distribution per input via stochastic forward passes; we use predictive entropy $H(\hat{\pi})$ (5.4) as the uncertainty score and accept if $H(\hat{\pi}(x)) \leq \tau$ with $\tau = 0.5142$. The dashed line in figure 5.4 marks this threshold, chosen at the valley between the correct (green) and incorrect (red) modes.

Coverage vs. reliability: table 5.1 shows test-set coverage after rejection: deferring $\sim 5\%$ of uncertain cases sharpens the accepted-set confusion matrix figure 5.2 and lifts accepted-set accuracy to $\sim 99\%$, enabling reliable automation.

Table 5.1: Coverage after entropy-based rejection. Total test samples: 163,944.

Metric	**Count**	**(%)**
Rejected ($H > 0.5142$)	8,400	5.12
Accepted	155,544	94.88

5.5 Classification report: comparative analysis (accepted predictions)

Table 5.2: Classification Report (on Accepted Predictions)

Class	Precision	Recall	F1-Score	Support
Analysis	1.00	1.00	1.00	8894
Backdoor	1.00	1.00	1.00	8897
Benign	0.98	1.00	0.99	8948
Bot	1.00	1.00	1.00	9000
Brute Force	1.00	1.00	1.00	7114
DDoS	0.97	0.99	0.98	8755
DoS	0.97	0.94	0.96	7681
Exploits	0.99	1.00	0.99	8206
Fuzzers	1.00	1.00	1.00	8642
Generic	1.00	0.96	0.98	2336
Infilteration	0.99	0.98	0.99	8817
Reconnaissance	0.99	0.99	0.99	8328
Shellcode	1.00	1.00	1.00	8938
Theft	1.00	1.00	1.00	8598
Injection	0.98	0.92	0.95	7594
MITM	0.98	0.82	0.89	1200
Password	0.99	1.00	1.00	8955
Ransomware	1.00	1.00	1.00	8989
Scanning	0.98	0.98	0.98	8427
XSS	0.92	1.00	0.96	7225
Accuracy			**0.99**	155,544
Macro Avg.	0.99	0.98	0.98	155,544
Weighted Avg.	0.99	0.99	0.99	155,544

The full classification report (Precision/Recall/F1 and support per class) complements the confusion matrix. The table 5.2 the key takeaways and relate them to the matrix structure and the uncertainty-aware acceptance policy.

Global perspective (macro vs. weighted). The *macro* averages (unweighted per-class means) sit slightly below the *weighted* averages (support-weighted means). This pattern indicates that a few minority or harder classes (e.g., long-tailed behaviours) have somewhat lower recall than the majority, while the overall (weighted) performance remains very high because most samples belong to classes that the model separates extremely well. In our run, the report shows (accepted-set) accuracy ≈ 0.99, macro averages near (Precision ≈ 0.99, Recall ≈ 0.98, F1 ≈ 0.98) and weighted averages near $(0.99, 0.99, 0.99)$, which is consistent with the strong diagonals in figure 5.2.

Per-class highlights. Most attack types exhibit near-perfect (≥ 0.99) Precision *and* Recall after abstention. Two classes stand out as the relatively hardest: *mitm* shows the lowest Recall (reflecting boundary ambiguity and limited support), and

injection has a modest Recall dip (with Precision still very high). Conversely, *xss* achieves Recall close to 1.00 with slightly lower Precision, suggesting that a small number of *xss*-like patterns appear in neighbouring classes (a benign false-alarm trade-off). These behaviours match the sparse off-diagonal mass in figure 5.2 and the attack family relationships (e.g., application-layer exploits vs. input-manipulation).

Why the acceptance policy helps. Entropy-based rejection removes a small fraction of uncertain samples concentrated around decision boundaries. This improves per-class Recall/Precision on the accepted set without changing the underlying classifier and provides a principled triage mechanism for the deferred cases. As a result, the accepted-set metrics reported above faithfully reflect the model's automation-ready regime, while the deferred $\sim 5\%$ can be routed to analysts or downstream checks.

5.6 False-Alarm Rate (FAR) vs. Data Rejection

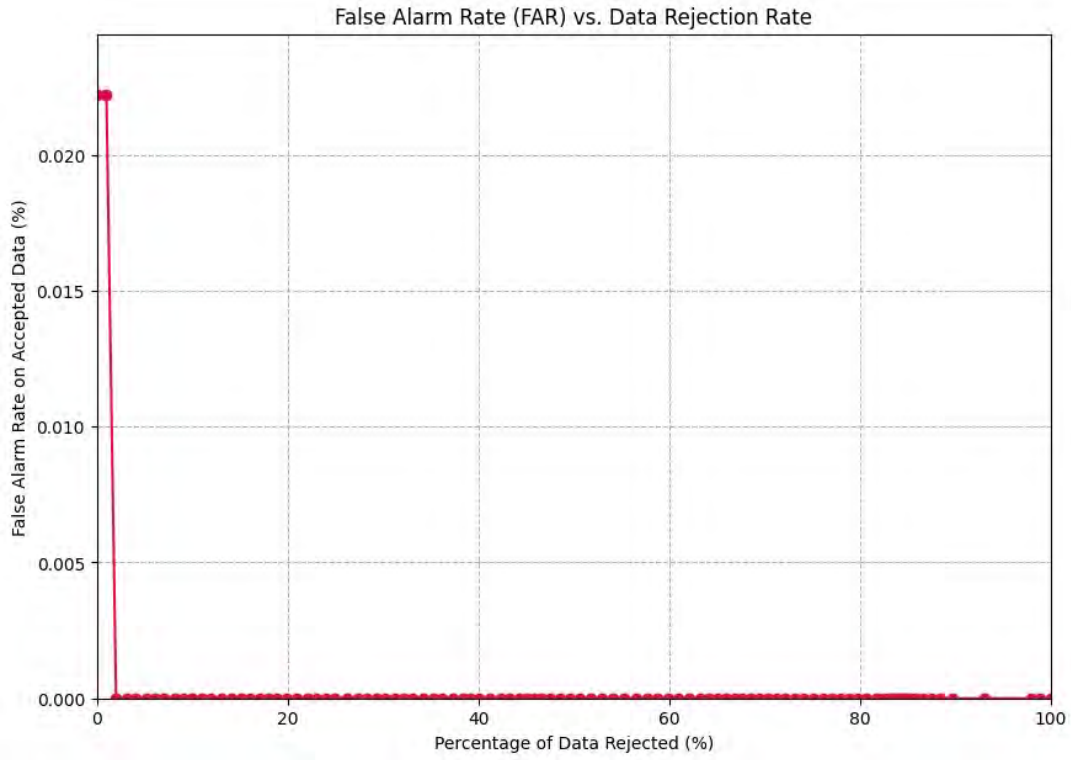

Figure 5.5: False-Alarm Rate vs Data Rejection

The figure 5.5 is obtained by sweeping the same entropy threshold τ used for abstention (cf. Sec. 5.4). For each τ we accept samples with $H(\hat{\pi}(x)) \leq \tau$ and compute the false-alarm rate on the *accepted* set,

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \times 100 \tag{5.9}$$

In 0% rejection the accepted-set FAR is $\approx 2.2\%$. Rejecting only the most uncertain $\sim$ 1–2% of samples collapses FAR to $\approx 0\%$, where it remains essentially flat. At

the chosen operating point $\tau = 0.5142$ (rejecting $\approx 5.1\%$), FAR on the accepted set is $\approx 0\%$ while the accepted-set accuracy is $\approx 99\%$. The steep initial drop and sustained floor near zero indicate that predictive entropy ranks error-prone cases effectively, enabling a practical trade-off: defer a small fraction of uncertain traffic and deliver near-zero false alarms on the remainder. This behaviour is consistent with the near-diagonal accepted-set confusion matrix and the high per-class precision in the classification report, and it explains the improvement from $\sim 97.4\%$ validation accuracy to $\sim 99\%$ accuracy on the accepted set at $\tau = 0.5142$. In short, this graph shows that the model's uncertainty measurement is extremely effective with higher confidence and excellent performance, proving our model not only knows the right answer most of the time but also knows when it's likely to be wrong.

5.7 Comparative Analysis on NF-UQ-NIDS-v2

We compare our probabilistic BVAE–NF classifier only against works whose settings are closest to ours on *NF-UQ-NIDS-v2*. Our model (70 epochs) attains a validation accuracy of $\approx 97.4\%$ without abstention, and—with an entropy-based abstention of $\sim 5\%$ most-uncertain samples—reaches an accepted-set accuracy of $\approx 99\%$. This exposes an accuracy coverage dial, higher reliability on the portion of traffic the model is confident about, while deferring borderline flows for secondary inspection.

Nearest apples-to-apples comparator (multi-class, deterministic): Gouda *et al.* [18] report multi-class (20-class) accuracy on *NF-UQ-NIDS-v2* using classical and deep baselines on a 100K subset: Random Forest 99.07%, CatBoost 99.04%, Decision Tree 98.78%, LSTM 98.87%, and CNN 98.56%. These numbers set a strong deterministic reference point but provide no calibrated confidence nor selective prediction.

Multi-class with limited labels (data efficiency rather than peak accuracy): "Always-Be-Pretraining" [16] tackles the same dataset as a label-efficiency problem: with only $\sim 3.7\%$ labeled data it achieves F1 = 93.22%, which is $\sim 99.8\%$ of its full-data F1. Although the target is different, it is a relevant multi-class comparator that prioritizes efficiency over headline accuracy.

Closest binary baselines (task differs, but setup is comparable): AE–MLP hybrid [17]: On a 100K *NF-UQ-NIDS-v2* subset, a shallow Autoencoder+MLP reaches 99.98% accuracy (binary), with RF 99.59% and XGB 99.54%. The scores are very high but reflect a reduced-size, binary setting without uncertainty estimates.

ExtraTrees + RFE [21]: After correlation analysis and recursive feature elimination, an 8-feature ExtraTrees model attains 98.13% accuracy, AUC 99.73%, and FAR 0.3589% (binary). This emphasizes the low-false-alarm axis; calibration and selective prediction are not reported.

Minimal-feature ensemble [19]: Using rank aggregation to select 8 features and tree ensembles, accuracy is $\sim 95.3\%$ (binary) while maintaining high throughput (classifying $\sim$ 7.6M samples in $\sim$ 0.95s). This highlights an efficiency trade-off rather than maximum accuracy.

All of the findings are summarized in the table 5.3 below.

Table 5.3: Representative results from recent literature

Study (venue, year)	Approach	Best metrics (reported)
Our work	BVAE–NF + Bayesian classifier (probabilistic; abstention)	Val. Acc. ≈ 97.4%; Accepted-set Acc. ≈ 99% @ ∼ 5% reject
Gouda et al. (Neural Comput. & Appl., 2024) [18]	RF / CatBoost / DT / LSTM / CNN	RF 99.07%; CatBoost 99.04%; DT 98.78%; LSTM 98.87%; CNN 98.56% (100K subset)
Amokrane et al. (Acta Polytechnica Hungarica, 2025) [21]	ExtraTrees + RFE (post-correlation)	Acc. 98.13%; AUC 99.73%; FAR 0.3589% (8 features)
Krupski et al. (Applied Sciences, 2024) [19]	Rank-aggregated FS + DT/RF	Acc. ∼95.3% (8 features)
AE–MLP hybrid (Computers, 2022) [17]	Autoencoder + MLP (hybrid)	AE–MLP 99.98%; RF 99.59%; XGB 99.54% (100K subset)
Always-Be-Pretraining GNNs (2024) [16]	GNN with self-supervised pretraining	Few-shot F1 93.22% using 3.7% labels (≈99.8% of full-data F1)

Where our method stands out:

- **Calibrated probabilities:** Enables better decision thresholds and reduces overconfident errors, which is particularly valuable for Security Operations Center (SOC) triage.
- **Selective prediction (abstention):** Achieves accepted-set accuracy near 99% with approximately 5% rejection, allowing operators to trade a small review rate for a meaningful false-alarm reduction.
- **Epistemic uncertainty:** Monte Carlo inference quantifies model uncertainty, supporting robust detection under traffic distribution shifts and for emerging or unseen attack types.
- **Better performance under distribution shift and novel classes:** The Bayesian framework generalizes more effectively when encountering traffic patterns or attack types not present in training data.
- **Actionable outputs:** Produces confidence, uncertainty, and abstention scores that assist analysts in prioritizing alerts and automating triage workflows.
- **Scientific clarity:** By modeling both likelihoods and calibrated posteriors, the framework explicitly distinguishes what the model *knows* from what it *does not know*, enhancing interpretability and transparency.

In short, our approach is *competitive in accuracy* and *differentiated in reliability.*

Chapter 6

Conclusion

6.1 Conclusion

This paper proposed a Bayesian Variational AutoEncoder with Normalizing Flows (BVAE-NF) model to reduce the false alarm rates (FAR) and achieve high multi-class intrusion detection rates. By combining generative modeling with probabilistic inference, the framework is able to find efficient trade-offs between data representation and uncertainty, producing dependable and calibrated predictions that outperform classical deterministic deep learning classifiers.

The first stage of this project involved a dynamic balancing pipeline based on Conditional Variational Autoencoders (CVAE), which are capable of generating class-consistent synthetic data for underrepresented attack categories. This approach corrected the extreme class imbalance of the NF-UQ-NIDS-v2 dataset, ensuring that minority classes were better represented in the model and thereby reducing overall bias.

The second stage implemented the Bayesian VAE-NF model that employed expressive latent representations with Monte Carlo–based uncertainty estimation. The epistemic uncertainty of the model weights was then quantified through a probabilistic approach, enabling an entropy-based rejection mechanism that significantly reduced incorrect classifications by avoiding uncertain predictions. The developed model achieved near-perfect ROC–AUC scores and an accepted-set accuracy of approximately 99%, demonstrating a high level of discriminative capability across 20 intrusion classes.

These contributions collectively demonstrate the success of Bayesian deep generative techniques in improving both the performance and reliability of intrusion detection. The findings indicate that uncertainty can also be leveraged for better interpretability and for developing practical implementation strategies that balance automation with human analyst oversight.

Societal Impact: The outcomes of this research hold meaningful implications for modern digital societies. As global communication networks, financial systems, and critical infrastructures increasingly depend on secure connectivity, cyberattacks pose growing risks to privacy, safety, and economic stability. The proposed BVAE-NF

framework contributes to strengthening cybersecurity resilience by reducing false alarms and enhancing the accuracy of detection systems. This leads to more efficient use of human resources in Security Operations Centers (SOCs), minimizes analyst fatigue, and ensures faster and more reliable responses to genuine threats. In a broader context, this work supports the creation of safer digital environments, protecting sensitive information, maintaining the trust of users, and sustaining the operational continuity of essential services.

In conclusion, this thesis presents a principled, uncertainty-sensitive approach to network intrusion detection that enhances both precision and reliability. By combining Bayesian inference with deep generative modeling, it takes a decisive step toward the development of credible, explainable, and robust intrusion detection systems capable of securing next-generation network infrastructures.

6.2 Limitations

Although the proposed Bayesian VAE-NF framework has shown significant improvements in the reduction of false-alarm rates and enhanced detection accuracy, still, there are some limitations that should be discussed.

- **Black-box nature:** The model's deep probabilistic architecture obscures the direct relationship between input features and predictions. This hinders analyst trust and complicates post-incident forensic analysis, as the decision-making process is not easily interpretable.
- **Limited detection of rare or evolving attacks:** Detection of attacks belonging to rare or dynamically evolving classes remains sub-optimal, even with the CVAE-based data balancing and synthetic data generation. This limitation arises from the scarcity of real samples, which restricts the diversity of learned decision boundaries and the fidelity of generative augmentation.
- **Single-dataset evaluation:** The model was evaluated only on the NF-UQ-NIDS-v2 dataset. Although this dataset is extensive and diverse, relying on a single benchmark restricts the generalizability of the results. Broader cross-dataset validation is required to confirm robustness under heterogeneous network environments.
- **Static uncertainty threshold:** The current uncertainty-aware rejection mechanism depends on a fixed entropy parameter. While effective, it lacks adaptability to dynamic traffic conditions and varying situational risks, which can lead to inconsistent abstention behavior across different network states.

6.3 Future Work

There are a number of avenues that may extend and enhance this research:

1. Future work will integrate Explainable AI (XAI) to enhance transparency. By using methods like SHAP, the model can provide feature-level explanations for its predictions, showing an analyst why a specific network flow was

flagged as malicious. Correlating these explanations with the model's inherent uncertainty will further augment its utility as a trustworthy forensic tool.

2. **Cross-Domain and Multi-Dataset Validation:** In future experiments, heterogeneous datasets should also be used to determine the transferability and overall robustness of the proposed model, including CIC-IDS-2017, UNSW-NB15, and BoT-IoT among others.

3. **Few-Shot and Meta-Learning Strategies:** To help with the lack of such rare attack samples, the use of few-shot or meta-learning strategies may enable the model to generalize on few examples and detect previously unseen threats in a more efficient way.

4. **Dynamic Uncertainty Calibration:** The combination of adaptive uncertainty abstentions or entropy-driven rejection with a decision policy founded on reinforcement learning can enable more context-sensitive abstentions and confidence calibration in live network conditions.

5. **Better Generative Modeling:** It is possible to extend the data-augmentation pipeline with more advanced architectures, e.g., Contrastive Tabular VAE (CTVAE), Diffusion-based tabular generators, or Transformer-VAEs, to generate more realistic synthetic flows and broaden the types of attacks in low-frequency categories while minimizing bias.

Following these guidelines, the suggested framework has the potential to evolve into a highly adaptable, uncertainty-sensitive intrusion detection system capable of operating effectively in a wide variety of dynamic and large-scale network environments.

Bibliography

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009. DOI: 10.1109/TKDE.2008.239.

[2] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[3] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *2017 IEEE international conference on big data and smart computing (BigComp)*, IEEE, 2017, pp. 313–316.

[4] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors*, vol. 17, no. 9, p. 1967, 2017. DOI: 10.3390/s17091967. [Online]. Available: https://doi.org/10.3390/s17091967.

[5] H. Mohamed, H. Hefny, and A. Alsawy, "Intrusion detection system using machine learning approaches," *Egyptian Computer Science Journal*, vol. 42, no. 3, 2018.

[6] S. Gurung, M. K. Ghose, and A. Subedi, "Deep learning approach on network intrusion detection system using nsl-kdd dataset," *International Journal of Computer Network and Information Security*, vol. 11, no. 3, pp. 8–14, 2019.

[7] S. Aldhaheri, D. AlGhazzawi, L. Cheng, B. A. Alzahrani, and A. Al-Barakati, "Deepdca: Novel network-based detection of iot attacks using artificial immune system," *Applied Sciences*, vol. 10, no. 6, p. 1909, 2020. DOI: 10.3390/app10061909.

[8] H. Hindy, R. Atkinson, C. Tachtatzis, J. N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, no. 10, p. 1684, 2020.

[9] H. Alkahtani and T. H. H. Aldhyani, "Intrusion detection system to advance internet of things infrastructure-based deep learning algorithms," *Complexity*, vol. 2021, 2021. DOI: 10.1155/2021/5579851.

[10] A. Rosay, K. Riou, F. Carlier, and P. Leroux, "Multi-layer perceptron for network intrusion detection," *Annals of Telecommunications/Annales Des Télécommunications*, vol. 77, no. 5–6, pp. 371–394, 2021. DOI: 10.1007/s12243-021-00852-0.

[11] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Towards a standard feature set of NIDS datasets," *CoRR*, vol. abs/2101.11315, 2021. arXiv: 2101.11315. [Online]. Available: https://arxiv.org/abs/2101.11315.

[12] M. Toğaçar, "Detecting attacks on iot devices with probabilistic bayesian neural networks and hunger games search optimization approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 1, 2021. DOI: 10.1002/ett.4418. [Online]. Available: https://doi.org/10.1002/ett.4418.

[13] E. C. P. Neto, S. Dadkhah, and A. A. Ghorbani, "Collaborative ddos detection in distributed multi-tenant iot using federated learning," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, IEEE, 2022, pp. 1–10.

[14] M. N. Ali, M. Imran, M. S. U. din, and B. S. Kim, "Low rate ddos detection using weighted federated learning in sdn control plane in iot network," *Applied Sciences*, vol. 13, no. 3, p. 1431, 2023.

[15] A. Awajan, "A novel deep learning-based intrusion detection system for iot networks," *Computers*, vol. 12, no. 2, p. 34, 2023.

[16] Authors, *Always be pre-training: Representation learning for network intrusion detection with graph neural networks*, 2024. DOI: 10.48550/arXiv.2402.18986.

[17] Authors, "Securing mobile edge computing using hybrid deep learning," *Computers*, 2024. DOI: 10.3390/computers13010025.

[18] H. A. Gouda et al., "Optimizing anomaly-based attack detection using classification machine learning," *Neural Computing and Applications*, 2024. DOI: 10.1007/s00521-023-09309-y.

[19] J. Krupski et al., "Extraction of minimal set of traffic features using ensemble of classifiers and rank aggregation for network intrusion detection systems," *Applied Sciences*, 2024. DOI: 10.3390/app14166995.

[20] A. Maiga, E. Ataro, and S. Githinji, "Intrusion detection with deep learning classifiers: A synergistic approach of probabilistic clustering and human expertise to reduce false alarms," *IEEE Access*, 2024.

[21] S.-B. Amokrane et al., "Enhancing intrusion detection system performance through feature selection," *Acta Polytechnica Hungarica*, 2025. DOI: 10.12700/APH.22.1.2025.1.10.

[22] 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, pp. 177–196, Jan. 2025. DOI: 10.12700/APH.22.1.2025.1.10.