Politecnico di Torino

Master's Degree in Computer Engineering

A.a. 2025/2026

Graduation Session December 2025

Autoencoder-Based Feature Extraction and Explainable Anomaly Detection in Network Security

Supervisors:
Alessio Sacco
Guido Marchetto

Candidates:
Christian Colella

Abstract

The growing complexity and heterogeneity of modern network traffic pose a significant challenge to anomaly detection in cybersecurity. Traditional models often fail to generalize across datasets with differing distributions and feature spaces, resulting in limited robustness when applied to unseen environments. This thesis proposes a unified framework for network anomaly detection that leverages multiple datasets to build a generalizable classification model.

The proposed approach utilizes AutoEncoders (AEs) to transform multiple datasets into a common feature space, thereby enabling their integration. We train an independent AE on each dataset to learn a compact, latent representation of its specific traffic patterns (both normal and anomalous). Once trained, only the encoder portion of each AE is retained to map the data into its latent space. This process generates meaningful and comparable features across all datasets, neutralizing inconsistencies like different scaling or feature definitions. These encoded representations are then merged into a single, unified dataset.

Finally, this combined dataset is used to train a Multi-Layer Perceptron (MLP) classifier to distinguish between benign and malicious traffic.

The approach was evaluated using three benchmark datasets — CIC-IDS2017, BoT-IoT, and UNSW-NB15 — each representing distinct network conditions and types of attacks. Experimental results demonstrate high detection performance, achieving F1-scores of 96.1% on CIC-IDS2017, 99.9% on BoT-IoT, and 90.5% on UNSW-NB15, with an overall cross-dataset F1-score of 99.5%. These outcomes confirm the strong generalization capability of the proposed method and its robustness across heterogeneous data sources. Finally, the SHAP (SHapley Additive exPlanations) framework was employed to interpret the model's predictions, offering insights into the most influential features and providing transparency in the decision-making process. Additionally, SHAP values were explored as a feature selection strategy to assess whether model performance could be improved.

Overall, the results confirm that the unified representation provides a reliable and effective strategy for network anomaly detection in heterogeneous environments.

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rapid growth and increasing complexity of modern computer networks have made them a critical component of both personal and enterprise infrastructures. Alongside these advancements, the sophistication and frequency of cyber threats have increased significantly, posing serious risks to the integrity, confidentiality, and availability of networked systems.

In this context, anomaly detection (AD) has emerged as a fundamental approach for identifying unexpected or malicious behavior within complex datasets. AD techniques aim to detect patterns that deviate from normal behavior, providing a flexible and data-driven alternative to traditional rule-based or signature-based security methods. By learning from observed data, these methods can identify previously unseen threats, making them particularly suitable for dynamic and evolving network environments.

A specific application of AD is network anomaly detection (NAD), which focuses on identifying unusual patterns in network traffic. NAD provides a way to monitor and detect potential threats in networks, complementing traditional security mechanisms. The increasing volume and diversity of network data, combined with constantly evolving threats, make it essential to develop methods capable of capturing abnormal behaviors effectively and efficiently.

1.1 NAD

Network Anomaly Detection (NAD) is a specialized area of anomaly detection that focuses on identifying unusual patterns or behaviors within network traffic. Its primary goal is to distinguish between normal and potentially harmful activities, providing a data-driven approach to safeguarding networked systems.

Unlike traditional security mechanisms, which often rely on predefined rules or known attack signatures, NAD methods can adapt to evolving network behaviors.

This capability is particularly important in modern networks, where traffic patterns are increasingly complex and dynamic. By learning from historical network data, NAD techniques can identify previously unseen threats, such as novel intrusion types, emerging malware campaigns, or unusual communication patterns between devices.

The study of NAD encompasses a variety of data types and sources, including packet-level information, flow statistics, and system logs. Each type of data presents unique challenges in terms of volume, dimensionality, and variability, motivating the development of specialized detection methods capable of handling diverse network environments. Modern networks, such as enterprise infrastructures, cloud systems, and IoT ecosystems, produce heterogeneous traffic that further increases the difficulty of detecting anomalies reliably.

Several approaches exist for detecting anomalies in networks. Classical statistical methods focus on deviations from expected distributions of traffic features, while machine learning techniques aim to learn patterns directly from the data. Both approaches offer complementary advantages: statistical methods are often simple and interpretable, whereas machine learning models can capture complex, non-linear patterns and generalize better to unseen situations.

Overall, NAD provides a critical layer of defense, complementing conventional security tools and enabling proactive monitoring of networks. Its importance continues to grow as network infrastructures expand and cyber threats become increasingly sophisticated, highlighting the need for robust and adaptive detection techniques that can operate effectively in diverse and dynamic network environments.

1.2 Challenges for NAD

Despite the significant progress achieved in recent years, the development of effective Network Anomaly Detection systems still faces numerous challenges. These difficulties arise from both the intrinsic complexity of modern networks and the limitations of current analytical techniques.

A first major challenge lies in the high variability of network traffic. Normal network behavior is not static — it evolves over time depending on user activity, applications, and infrastructure changes. As a result, defining what constitutes “normal” behavior becomes difficult, and models trained on past data may rapidly become outdated. This phenomenon, known as concept drift, often leads to a degradation of detection performance in real-world deployments.

Another crucial aspect is the imbalance between normal and anomalous samples. In network environments, malicious events are relatively rare compared to the large volume of legitimate traffic. This imbalance can bias learning algorithms toward normal behavior, making it difficult to detect subtle or novel attacks. Furthermore, anomalies themselves are highly diverse — ranging from scanning attempts to data exfiltration — and cannot always be easily generalized.

The scarcity of labeled data represents another major obstacle. Annotating network traffic requires significant domain expertise and time, especially in large-scale environments. As a consequence, supervised learning approaches are often impractical, driving the adoption of unsupervised or semi-supervised techniques that can operate with limited supervision.

Scalability is also a key concern. Network traffic generates massive amounts of data in real time, and the Network anomaly detection systems must process and analyze this information efficiently without introducing significant delays. Ensuring both high detection accuracy and computational efficiency is a difficult trade-off, especially when models are deployed in high-throughput or resource-constrained environments.

Finally, interpretability and adaptability remain open issues. While complex machine learning models, such as deep neural networks, have demonstrated strong detection performance, their decision processes are often opaque. This lack of transparency can hinder trust and prevent network administrators from understanding or validating alerts. Moreover, adapting these models to new network conditions or attack types without complete retraining remains a significant research challenge.

1.3 Limitations of the current approaches

Although numerous methods have been proposed to address the challenges of Network Anomaly Detection, existing approaches still exhibit significant limitations that hinder their practical effectiveness and generalization capabilities.

Traditional signature-based systems, such as intrusion detection systems (IDS), rely on predefined patterns of known attacks. While effective against previously observed threats, they fail to identify novel or evolving attacks that do not match existing signatures. This lack of adaptability has made signature-based methods increasingly inadequate in modern, rapidly changing network environments.

To overcome these limitations, the research community has extensively explored machine learning–based techniques, which aim to automatically learn patterns of normal and abnormal behavior from data. However, despite their promise, these methods also suffer from several drawbacks. Many models are highly dataset-dependent, meaning that their performance significantly degrades when applied to data from a different network or environment. This issue highlights a lack of generalization and transferability, which prevents most models from being deployed effectively in real-world, heterogeneous contexts.

Another limitation concerns the requirement for labeled data. Supervised learning methods can achieve high accuracy when sufficient labeled samples are available, but this is rarely the case in network security. Manual labeling of network traffic is both costly and error-prone, leading to incomplete or inconsistent ground truth. As a result, unsupervised and semi-supervised methods have gained popularity, yet these approaches often produce a high number of false positives, reducing their reliability in operational settings.

Furthermore, many existing Network anomaly detection systems struggle with scalability and computational efficiency. Deep learning–based solutions, while powerful, require substantial computational resources for both training and inference. This makes them difficult to deploy in real-time monitoring systems or on edge devices with limited processing capabilities. Balancing detection accuracy with processing latency remains an ongoing challenge.

Finally, model interpretability is still a major concern. The black-box nature of many machine learning models makes it difficult for analysts to understand why a particular event was flagged as anomalous. This lack of transparency complicates the validation process and undermines trust in automated detection systems, particularly in security-critical applications.

These limitations collectively underline the need for more robust, adaptive, and generalizable approaches capable of handling diverse datasets and evolving network behaviors — an aspect that motivates the exploration of multi-dataset and meta-learning techniques.

1.4 Multi-Dataset Techniques

The limitations observed in current Network Anomaly Detection systems have highlighted a fundamental issue: most approaches are designed, trained, and evaluated using a single dataset. While this setup simplifies experimentation and benchmarking, it often results in models that perform well only within the specific conditions of that dataset, failing to generalize to different network environments or traffic characteristics. This lack of generalization remains one of the most significant obstacles to deploying network anomaly detection systems effectively in real-world scenarios.

To address this limitation, recent research has started to explore multi-dataset techniques, which aim to leverage data from multiple sources or domains to build more robust and adaptable detection models. The core idea is that exposure to heterogeneous network data — collected under varying conditions, topologies, and attack scenarios — enables models to learn representations that are less sensitive to the peculiarities of a single environment. As a result, these approaches seek to enhance the transferability and reliability of anomaly detection systems across diverse network contexts.

Multi-dataset strategies can be interpreted from two complementary perspectives.

The first focuses on joint or unified training, where datasets from different domains are combined or aligned to provide a richer and more diverse training base. This may involve harmonizing features, normalizing traffic statistics, or adopting shared representations that make different datasets comparable. The second perspective emphasizes transfer and adaptation, where a model trained on one dataset is adjusted to operate effectively on another. This can be achieved through fine-tuning, domain adaptation, or meta-learning techniques that promote knowledge reuse and adaptability.

Despite their promise, multi-dataset methods introduce new challenges. Differences in data collection procedures, feature definitions, and labeling standards often hinder integration. Moreover, managing the trade-off between generalization and overfitting becomes increasingly complex as the diversity of the training data grows. Still, the benefits of this paradigm are substantial: by embracing variability instead of avoiding it, multi-dataset approaches represent a meaningful step toward more resilient, scalable, and generalizable anomaly detection systems capable of operating effectively in real-world, heterogeneous network environments.

1.5 Goal

The main goal of this thesis is to develop a unified approach for network anomaly detection that can perform effectively across multiple datasets. Traditional detection systems are usually trained and tested on a single dataset, which reduces their generalization ability and limits their use in real-world network environments.

This work addresses this issue by studying how combining heterogeneous datasets and applying consistent preprocessing and feature representation can improve the robustness of machine learning models. In addition, explainable AI techniques, in particular SHAP, are employed to support feature selection and provide transparency in the model's decision-making.

The specific objectives of the thesis are:

1. **Analyze** the main challenges and limitations of current anomaly detection methods when applied to different datasets.

2. **Create** a unified dataset by merging multiple publicly available sources to enable a more general evaluation.

3. **Design** a detection framework capable of maintaining consistent performance across datasets.

4. **Evaluate** the proposed approach and discuss its effectiveness and possible improvements.

5. **Use** explainable AI techniques (SHAP) to interpret the most relevant features and support feature selection.

The final goal is to contribute to the development of more general and adaptable anomaly detection models that can be reliably applied across diverse network environments and employ explainable AI techniques to provide insight into their decision-making.

1.6 Thesis Structure

The thesis is organized into five chapters:

- **Chapter 1 – Introduction**
  Provides background on anomaly detection and network anomaly detection, introduces the main challenges in Network Anomaly Detection systems, and defines the goals of the study.
- **Chapter 2 – Related Works**
  Reviews previous research on anomaly detection and network anomaly detection, focusing on key methodologies, datasets, and evaluation strategies. Special attention is given to multi-dataset approaches and existing research gaps.
- **Chapter 3 – System Overview**
  Describes the design of the proposed detection framework, including data preprocessing, dataset unification, latent space representation, and the classification model.
- **Chapter 4 – Evaluations and Results**
  Presents the datasets used in the study, describes the experiments conducted to evaluate the proposed system, analyzes performance across multiple datasets, assesses the model's generalization capability, and discusses feature importance and feature selection using SHAP to interpret the classifier's decisions.
- **Chapter 5 – Conclusions and Future Work**
  Summarizes the main findings, discusses the contributions of the study, and outlines potential directions for future research.

Chapter 2

Related Works

Network Anomaly Detection has become increasingly critical due to the growing volume, diversity, and complexity of modern network traffic. While many approaches have been proposed, achieving robust and generalizable detection across heterogeneous datasets remains a key challenge.

2.1 Deep Learning Approaches for Anomaly Detection

Anomaly detection (AD) methods aim to identify patterns in data that deviate from normal behavior. Deep learning approaches have been widely adopted in this context due to their ability to capture complex, non-linear relationships in high-dimensional data. Several works have focused on improving feature learning, handling imbalanced datasets, and leveraging latent representations to enhance detection performance.

Jing et al. (2021) [1] propose a multiset feature learning approach for highly imbalanced data classification. Their method combines multiple feature sets to better capture the variability of normal and anomalous instances, demonstrating improved performance on benchmark datasets. Although this work does not specifically target network traffic, it provides useful insights into handling imbalanced datasets, a common issue in network anomaly detection.

Sakurada and Yairi [2] introduce an autoencoder-based approach for unsupervised anomaly detection. The model learns to reconstruct normal data and flags instances with high reconstruction error as anomalies. This foundational work highlights the effectiveness of autoencoders in learning compact representations of normal behavior, which later inspired several NAD models.

An and Cho (2015) [3] extend this idea by employing variational autoencoders (VAE) for anomaly detection. By modeling the probabilistic distribution of the data in the latent space, VAEs provide a principled way to estimate the likelihood of new instances, enabling the detection of previously unseen anomalies. This methodology has influenced subsequent network-focused models that rely on probabilistic latent representations.

Overall, these deep learning-based AD approaches provide the methodological foundation for detecting abnormal patterns in complex datasets and motivate the application of similar techniques in the network anomaly detection domain.

2.2 Deep Learning for Network Anomaly Detection

Network Anomaly Detection (NAD) focuses on identifying abnormal patterns specifically within network traffic. Deep learning methods have become increasingly popular in this area due to their ability to model complex temporal and spatial relationships in network data. These approaches often leverage autoencoders, recurrent neural networks, or generative models to capture normal network behavior and detect deviations.

Fu et al. (2023) [4] propose GANAD, a GAN-based method for network anomaly detection. By training a generative adversarial network to model normal traffic patterns, anomalies are identified as instances that the generator fails to reproduce accurately. While this approach demonstrates strong detection performance, it also highlights challenges related to training stability and computational cost, especially in large-scale network environments.

Sharma et al. (2024) [5] provide a holistic review of unsupervised learning methods for NAD, evaluating the performance of various deep learning techniques across multiple network datasets. Their analysis emphasizes the trade-offs between detection accuracy, computational complexity, and interpretability, and underscores the need for methods that can generalize beyond a single dataset.

Abdelkhalek and Mashaly (2023) [6] address the class imbalance problem in network intrusion detection systems by combining deep learning models with data resampling techniques. They show that applying oversampling and undersampling strategies alongside architectures such as LSTM or CNN improves the detection of rare attack classes, a challenge particularly relevant when integrating multiple datasets. This study highlights how differences in data distribution affect model performance, emphasizing the importance of handling class imbalance in generalizable NAD systems.

These studies demonstrate the versatility of deep learning for network anomaly detection and highlight recurring challenges, including imbalanced data, dataset dependence, and computational constraints. Such insights provide a foundation for exploring multi-dataset approaches aimed at improving generalization across diverse network environments.

2.3 Multi-Dataset Approaches in NAD

While many network anomaly detection models perform well on a single dataset, their effectiveness often decreases when applied to other network environments. Multi-dataset approaches aim to improve generalization by leveraging heterogeneous datasets, standardizing features, and applying transfer or representation learning techniques.

FaaC (Feature as a Count) (Magán-Carrión et al., 2024) [7] proposes a method for unifying multiple network datasets by transforming network traffic features into count-based representations. This approach facilitates the combination of datasets with different structures and scales, enabling models to learn patterns that generalize beyond a single data source.

NetFlow-based datasets (Sarhan et al., 2020) [8] provide a standardized representation of network traffic by aggregating packet-level information into flow-based features such as duration, byte counts, and packet statistics. This structure enables the integration of heterogeneous datasets by offering a consistent feature space and labeling scheme, facilitating the development of models capable of generalizing across multiple network environments.

Learn-IDS (Wang et al., 2024) [9] introduces a framework that bridges gaps between datasets using representation learning and transfer learning. By adapting models trained on one dataset to operate effectively on another, Learn-IDS improves detection accuracy and robustness in multi-dataset scenarios, demonstrating the benefits of knowledge transfer for generalization.

Meta (Wali et al., 2024) [10] proposes a unified, multimodal dataset for network intrusion detection systems. By combining multiple network datasets into a single framework with standardized features and labels, Meta enables the development and evaluation of models that are inherently more robust to dataset heterogeneity and better suited for real-world deployment.

These studies collectively highlight the importance of addressing dataset heterogeneity, feature standardization, and knowledge transfer to improve the generalization of NAD models. They provide a framework for developing more robust and adaptable detection systems capable of performing reliably across diverse network environments.

Chapter 3

System Overview

3.1 Proposed Solution

The proposed solution to address the challenge of building a generalizable classification model is to employ multiple datasets in order to make the model as generic as possible. Each dataset represents different types of network traffic, providing a diverse foundation that enhances the model's capacity to generalize across varying traffic patterns. The final objective is a binary classification task, distinguishing between benign and malicious traffic.

Three distinct datasets were selected — CIC-IDS2017, BoT-IoT, and UNSW-NB15 — with the goal of unifying them into a single combined dataset to train the classification model. Details on the datasets, including features and statistics, are presented in Chapter 4.

Initially, the unification strategy consisted of selecting a fixed number of features and aligning the datasets accordingly. Datasets with fewer features than the chosen reference were padded with a constant value to match the reference feature set. However, this approach introduced a major drawback: the padding values effectively acted as dataset-specific identifiers, allowing the model to infer from which dataset a record originated and bias its predictions toward the corresponding class distribution.

To overcome this issue, a two-step solution is adopted. In the first step, each dataset is mapped into a latent space using an autoencoder, generating a smaller and standardized representation that preserves the original data distribution while removing dataset-specific identifiers. In the second step, an MLP model is trained for binary classification, using as input the unified dataset obtained by combining the transformed representations. The feed-forward architecture was chosen for its simplicity, efficiency, ability to model complex patterns in tabular data, and its widespread use in binary classification tasks. This approach offers several advantages: it mitigates dataset-specific biases, reduces the risk of overfitting to any single dataset, and enables the model to effectively leverage diverse datasets with heterogeneous feature sets.

In addition, explainable AI techniques, such as SHAP, are employed to analyze the influence of individual features on the model's predictions and to provide insight into the decision-making process(see Section 4.3.3)

In summary, the proposed solution relies on autoencoders to unify heterogeneous datasets into a shared latent space, followed by an MLP classifier trained to perform robust binary classification.

The overall structure of the proposed approach is illustrated in Figure 3.1, which summarizes the main phases of the implemented solution.

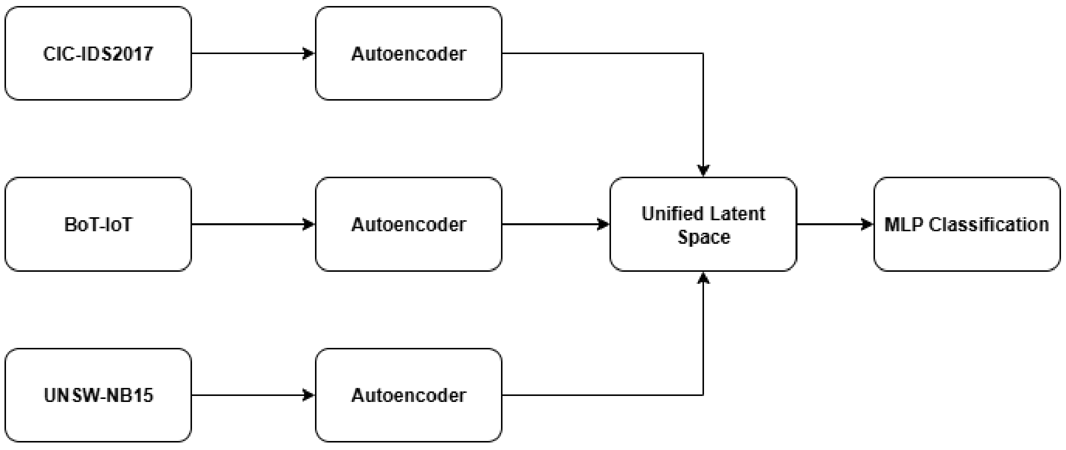

Figure 3.1: Overall architecture of the proposed solution.

This solution is described in detail in the following section, where the adopted system architecture is presented.

3.2 Architecture

In this section, the system architecture supporting the proposed solution is presented.

Below is a list of the main modules:

1. **Data Pre-Processing**
2. **Latent Space and Autoencoders**
3. **MLP Classification**
4. **Model Training**

The following subsections provide a detailed explanation of each module.

3.2.1 Data Pre-Processing

As the first module in the pipeline, Data Pre-Processing is responsible for producing a clean and accurate representation of the data. This step is fundamental in any machine learning workflow, since the quality of the data directly affects the performance of the trained model. Preprocessing ensures that the datasets are consistent, comparable, and free from noise and inconsistencies that could compromise the results or reduce the reliability of the classification task. In a typical machine learning pipeline, Data Pre-Processing may include the following steps:

- **Data cleaning** - this step ensures the dataset is accurate and reliable by addressing issues such as missing values, noisy data, and duplicates.
- **Data Transformation** - this step converts the data into a format suitable for analysis and model training. Common techniques include scaling, normalization, standardization, and feature engineering.
- **Data Reduction** - this step reduces the number of features while preserving the essential information.Typical techniques include feature selection, dimensionality reduction methods such as Principal Component Analysis (PCA), and sampling.
- **Data Augmentation** - this step generates synthetic data to enhance the dataset, increasing its size and diversity. It is commonly applied in domains such as image and text processing, but can be conceptually used for any type of data to improve model robustness.
- **Data Encoding** - this step converts categorical or non-numerical variables into numerical representations that can be processed by machine learning models. Common techniques include one-hot encoding, label encoding, and binary encoding.
- **Data Integration** - this step combines data from multiple sources into a single unified dataset, ensuring consistency and reducing redundancy. Typical techniques include schema matching, to align heterogeneous structures, and data fusion, to resolve conflicts among overlapping records.

The preprocessed data is now ready for further transformation and model training. The details on the specific preprocessing steps applied in the proposed solution are described in Section 4.1.

3.2.2 Latent Space and Autoencoders

A latent space is a lower-dimensional representation of the original data in which the essential structure and relationships among features are preserved, while redundant or dataset-specific variations are minimized. Mapping datasets into a common latent space allows heterogeneous datasets with different feature sets to be aligned, standardized, and ultimately unified into a single dataset suitable for machine learning. Latent space representations have several key applications, including:

- **Dimensionality Reduction** - It compresses high-dimensional data into a lower-dimensional space while retaining the most relevant features, improving computational efficiency and facilitating model training.
- **Data Alignment and Integration** - It enables the alignment of heterogeneous datasets into a common space, allowing their unification without introducing dataset-specific biases.
- **Feature Extraction for Generalization** - It extracts the most meaningful representations from the data, helping models learn general patterns and enhancing their ability to generalize to new, unseen data.

Latent space representations can be obtained using various techniques. Traditional dimensionality reduction methods, such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA), reduce dimensionality while retaining the most relevant information. More advanced approaches rely on representation learning with neural networks, such as autoencoders, which can capture complex, non-linear structures in the data. In the proposed solution, autoencoders were chosen among these approaches to obtain a compact and informative latent representation.

Autoencoders

Autoencoders are a type of neural network designed to learn efficient, compact representations of data in an unsupervised manner. They consist of two main components: an encoder, which maps the input data into a lower-dimensional latent space, and a decoder, which reconstructs the original data from this latent representation. The network is trained to minimize the reconstruction error between the input and the output, thereby forcing the model to capture the most relevant features and underlying structure of the data. Formally, given an input vector $x \in \mathbb{R}^n$, the encoder function $f_\theta(x)$ maps x to a latent vector $z \in \mathbb{R}^m$, where $m < n$, and the decoder function $g_\phi(z)$ reconstructs $\hat{x}$ such that the reconstruction loss $L(x, \hat{x})$ is minimized. Common choices for the loss function include mean squared error (MSE) for continuous data and cross-entropy loss for binary or categorical data. Autoencoders have several important properties that make them suitable for latent space representation in machine learning pipelines:

1. **Dimensionality Reduction** - By forcing the input data through a bottleneck layer, autoencoders learn a compressed representation that preserves essential information while discarding redundant or noisy features.

2. **Non-linear Feature Extraction** - Unlike linear methods such as Principal Component Analysis (PCA), autoencoders can model complex, non-linear relationships in the data, making them more expressive for high-dimensional and heterogeneous datasets.

3. **Data Integration and Standardization** - When multiple datasets with different feature sets are mapped into the same latent space, inconsistencies are reduced, and the resulting representations can be effectively combined for downstream tasks.

There exist several variants of autoencoders, each designed to address specific challenges:

- **Variational Autoencoders (VAEs)** - Introduce a probabilistic approach to modeling the latent space, allowing for generative modeling and smoother interpolations between points in the latent space.

- **Denoising Autoencoders (DAEs)** - Trained to reconstruct the original input from a corrupted version, enhancing robustness and generalization.

- **Sparse Autoencoders** - Impose sparsity constraints on the latent representation to encourage the network to learn more meaningful features.

In summary, autoencoders offer an effective way to obtain compact and meaningful representations of data, providing a solid basis for further processing and machine learning tasks. Figure 3.2 illustrates the architecture of an autoencoder, highlighting the encoder and decoder components and showing the transformation of input data into a latent representation and its subsequent reconstruction.

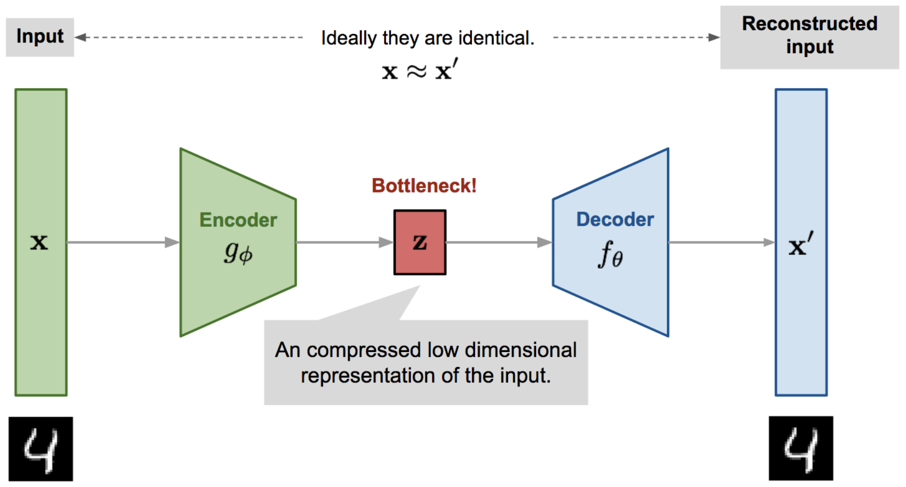
Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 3.2: An image of a variational autoencoder

The following paragraph presents the specific implementation used in the proposed solution.

Implementation

As mentioned in the previous section, this part illustrates the specific implementation of the autoencoder adopted in the proposed solution. The goal is to map each dataset into a lower-dimensional latent space through its corresponding encoder, ensuring a common feature dimensionality that enables their subsequent unification. To this end, a separate autoencoder was trained for each dataset, and only the encoder part was retained for transforming the data into the desired latent representation. The autoencoder features a feed-forward architecture for both encoder and decoder. The encoder consists of multiple fully connected layers, each followed by batch normalization and LeakyReLU activation functions, progressively compressing the input features into a latent space of fixed dimensionality. The decoder mirrors this design, gradually expanding the latent representation back to the original input dimension.

Model architecture In the proposed solution, the encoder consists of three layers arranged as follows:

1. **Layer 1 (128 features)** – expands the input to a higher dimensionality, allowing the network to capture complex, non-linear interactions among features.

2. **Layer 2 (64 features)** – gradually compresses the representation, preserving essential information while reducing dimensionality.

3. **Layer 3 (N features)** – maps the data to the fixed-size latent space, producing a standardized representation suitable for unifying multiple datasets.

Note: N represents the dimension of the latent space, which can be chosen based on the results obtained during experimentation.

LeakyReLU activation functions in the hidden layers introduce non-linearity and allow the network to capture complex feature interactions, while avoiding the vanishing gradient problem common in standard ReLU activations.

This design ensures that the latent representation captures the essential structure of the data while maintaining the capability of accurate reconstruction.

3.2.3 MLP Classification

After obtaining the latent representations of the datasets through the encoders, the next step in the pipeline is the training of a classification model. The objective is to learn a mapping from the unified latent feature space to the target classes, enabling the detection and classification of different types of network traffic. Several machine learning models can be employed for this task, including decision trees, support vector machines, logistic regression, and neural networks. Among the latter, Multi-Layer Perceptrons (MLPs) are particularly suitable for tabular data and effective in binary classification tasks. Therefore, an MLP was chosen for the proposed solution, balancing simplicity, efficiency, and the ability to model complex patterns in the data.

Multi-Layer Perceptrons (MLP)

A Multi-Layer Perceptron (MLP) is one of the most fundamental and widely used neural network architectures in supervised learning. It consists of a series of fully connected layers that transform input data through a combination of linear operations and non-linear activation functions. Despite its relative simplicity compared to more advanced architectures, the MLP remains highly effective for a broad range of classification and regression tasks.

An MLP is organized into three main components:

- **Input layer** – receives the data and defines the dimensionality of the problem. Each unit corresponds to one feature of the input vector.
- **Hidden layers** – one or more intermediate layers that progressively transform the input representation, enabling the model to learn hierarchical and abstract features.
- **Output layer** – produces the final prediction. Its structure and activation function depend on the task (e.g., softmax for multi-class classification, sigmoid for binary classification, or linear activation for regression).

The MLP is trained using supervised learning: the model parameters are iteratively updated to minimize a loss function, with gradients computed via backpropagation and parameters updated by an optimizer. A separate validation set is employed to monitor performance and guide training decisions such as learning rate adjustments and early stopping.

Figure 3.3 illustrates the architecture of the MLP model used for the solution.

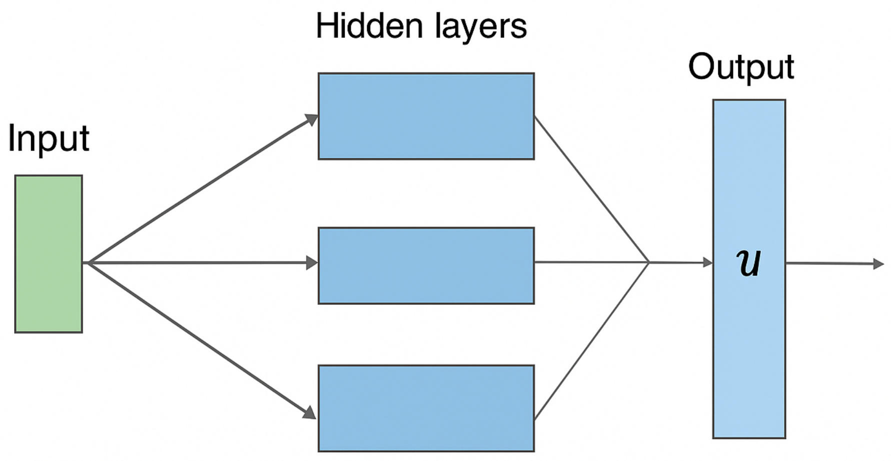

Figure 3.3: An image of the MLP model

The main advantages of using MLPs are:

- **Versatility** – MLPs can be applied to a wide range of problems, including both classification and regression tasks.
- **Non-linearity** – the use of activation functions allows them to model complex, non-linear relationships in the data.
- **Parallel computation** – training computations can be efficiently accelerated using GPUs.

Limitations include the potential for overfitting the training data if proper regularization techniques are not applied, and the high computational cost associated with increasing the number of layers. In the following paragraph, the specific implementation used in the proposed solution is illustrated.

Implementation

The MLP classifier is implemented as a single model applied to the unified latent space, unlike the autoencoders where a separate model was trained for each dataset. This design provides a consistent mapping from latent features to target classes. The architecture consists of multiple fully connected layers with normalization and non-linear activations, and includes regularization mechanism such as weight decay. The final output layer applies a sigmoid activation to produce a probability score for binary classification.

Model architecture The architecture of the MLP can be summarized as follows:

- **Input layer** – receives as input the latent features obtained from the unified dataset.
- **Hidden layers** – multiple fully connected layers, each followed by normalization (LayerNorm) and ReLU activation function, allowing the model to capture complex non-linear relationships in the data.
- **Regularization** – weight decay is employed to mitigate the risk of overfitting, improving the generalization ability of the network.
- **Output layer** – The final layer produces a single raw output (often referred to as a logit), which is then passed through a sigmoid activation to obtain a probability score for binary classification.

The number of neurons per layer and the total number of layers constitute hyperparameters that are selected during experimentation. The specific values used in the proposed solution are detailed in Chapter 4.

3.2.4 Model Training

Model training is the process through which a machine learning model learns patterns and relationships from data in order to make accurate predictions on unseen inputs. During training, the model iteratively adjusts its parameters, such as weights and biases, to minimize a loss function that quantifies the error between predictions and expected outputs. This process represents the actual "learning" phase in machine learning, where the model's knowledge is encoded in the optimized parameter values.

The training workflow typically involves running the model with training data, measuring its performance through a loss function, updating parameters using optimization algorithms, and validating its performance on a separate validation set to monitor generalization. In addition, "hyperparameters", which define structural aspects of the model and influence learning dynamics, may need to be tuned to achieve optimal performance.

Different machine learning models are typically trained according to one of several learning paradigms, each defining a distinct approach to how knowledge is acquired and applied. ML models are typically categorized as belonging to one of two distinct machine learning paradigms:

- **Supervised learning** - is used when a model is trained to predict the “correct” output for an input. It applies to tasks that require some degree of accuracy relative to some external “ground truth,” such as classification or regression.

- **Unsupervised learning** - is used when a model is trained to discern intrinsic patterns and correlations in data. Unlike supervised learning, unsupervised learning doesn’t assume the existence of any external ground truth against which its outputs should be compared.

Hybrid approaches also exist, such as self-supervised or semi-supervised learning, which combine elements of these paradigms.

In the proposed solution, both supervised and unsupervised paradigms are employed: autoencoders are trained in an unsupervised fashion to learn latent representations of each dataset, while the MLP classifier is trained in a supervised manner to perform binary classification on the unified latent space. The training strategies for both the autoencoder and the MLP follow a common workflow, as summarized below. Specific details for each model are described later in this part.

General Training Strategy Both models share a common training workflow. In particular:

- a separate validation set is used to monitor performance and guide learning decisions;

- early stopping is applied to avoid overfitting and unnecessary computation;

- a learning rate scheduler (ReduceLROnPlateau) is employed to adapt the learning rate dynamically;

- at each epoch, the model achieving the lowest validation loss is saved as the best version;

- training history, including losses and learning rate values, is recorded for subsequent analysis.

Autoencoder Training

The autoencoder was trained with a self-supervised objective where the input data also served as the target, with the goal of minimizing reconstruction error and using the encoder to transform each dataset.

- **Loss function** - Mean Squared Error (MSE)

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{3.1}$$

  (where x_i is the input and $\hat{x}_i$ is the reconstructed output)

- **Optimizer** - Adam
- **Scheduler parameters** - learning rate reduced by factor 0.5 after 3 epochs without validation loss improvement

Once training was completed, the encoder from the best-performing model was used to project the datasets into the latent space.

MLP Training

The MLP was trained in a supervised fashion on the unified latent representations for binary classification.

- **Loss function** – Binary Cross Entropy with Logits (BCEWithLogitsLoss):

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log\Big(\sigma(z_i)\Big) + (1 - y_i)\log\Big(1 - \sigma(z_i)\Big)\Big] \tag{3.2}$$

  where $y_i \in \{0,1\}$ is the true label, z_i is the predicted logit, and σ is the sigmoid function. Class weighting was applied to mitigate imbalance.

- **Optimizer** – AdamW with weight decay = 0.0001.
- **Scheduler parameters** – learning rate reduced by a factor of 0.5 after 3 epochs without validation loss improvement.

After training, the best-performing MLP was loaded from disk and employed for final classification. In the next chapter are illustrated the details of datasets,the experimentation performed, and the final results.

Chapter 4

Evaluations and Results

4.1 Datasets

As mentioned earlier, three different datasets were selected for the proposed solution: CIC-IDS2017, BoT-IoT and UNSW-NB15. The idea behind this choice is to rely on a diverse set of network traffic data, so that the machine learning model can be trained to generalize better. For this reason, the chosen datasets all belong to the network traffic domain and include a wide range of attack types, making them suitable for evaluating the robustness of the proposed approach. Each dataset is described below, together with the data preparation steps and the latent rappresentations obtained in this solution.

4.1.1 CIC-IDS2017

Introduction

The CIC-IDS2017 dataset was created by the Canadian Institute for Cybersecurity and is one of the most widely used benchmarks for intrusion detection research. It contains realistic network traffic collected over a period of five days, simulating both normal user behavior and a wide range of modern cyberattacks. The traffic was generated in a controlled environment that included typical activities such as web browsing, email, file transfer, and streaming, along with attack scenarios such as brute force, denial of service (DoS), distributed denial of service (DDoS), infiltration, botnet, and web-based attacks.

The dataset includes raw network flows in pcap format as well as structured flow-based features extracted using the CICFlowMeter tool. In total, it provides more than 80 statistical features per flow, covering information about packet-level characteristics, time-based features, and content-based attributes. These characteristics make the CIC-IDS2017 dataset particularly suitable for evaluating anomaly detection systems, as it reflects heterogeneous attack strategies and diverse benign traffic patterns in a realistic setting.

For this solution, the full CIC-IDS2017 dataset is employed in its original form, encompassing all 78 features and all recorded flows before any preprocessing is applied, as summarized in Table 4.1.

Table 4.1: Parameters of dataset CIC-IDS-2017

Parameters	**original**
number of features	78 (+1 Label)
number of instances	2830743
number of normals	2273097
number of attacks	557646

As shown in Table 4.2, normal traffic represents approximately 80% of the dataset, while attack traffic accounts for the remaining 20%.

Table 4.2 shows the distribution of the dataset by attack type, with the number of instances for each category.

Table 4.2: CIC-IDS-2017 Traffic Instances

Traffic	**Instances**
BENIGN	2273097
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack & Brute Force	1507
Web Attack & XSS	652
Infiltration	36
Web Attack & Sql Injection	21
Heartbleed	11

A detailed description of any preprocessing applied, including feature selection or cleaning operations, is presented in the following section.

Data Preparation

The CIC-IDS2017 dataset was prepared through a sequence of cleaning and transformation steps aimed at ensuring consistency and reliability. All CSV files provided by the authors were first loaded and concatenated into a single dataset. To improve data quality, missing values, infinite values, duplicate records, and constant features (i.e., columns with only a single unique value) were removed.

The original multi-class labeling was then simplified into a binary format, distinguishing between benign traffic (label 0) and attack traffic (label 1). Since the dataset includes both numerical and categorical features, the categorical attributes were converted into numerical form using label encoding, making them suitable for model training.

After these preprocessing steps, the dataset retained 71 features and 2,520,798 instances, with 2,095,057 benign and 425,741 attack flows. This slight reduction compared to the original dataset (Table 4.1) is mainly due to removal of duplicates, constant features, and incomplete records.

The dataset was split into training (70%), validation (15%), and test (15%) sets. The split was stratified to preserve the proportion of benign and attack flows, ensuring a balanced representation across subsets.

Before training the autoencoder, all features were scaled to the [0,1] range using Min-Max normalization. The scaling parameters were computed exclusively on the training set and then applied to the validation and test sets to avoid information leakage. After normalization, the dataset was ready for latent representation learning.

Latent Representation

Once the CIC-IDS2017 dataset was preprocessed and normalized, the autoencoder was trained to learn a compact latent representation of the input flows. Several latent dimensions were tested, as described in the Hyperparameters Section, and the configuration with a latent dimension of 32 provided the best overall performance.

Figure 4.1 shows the training and validation loss across the epochs for this configuration. Both curves decrease steadily to very low values and remain close to each other, indicating stable convergence without signs of overfitting. This confirms that the autoencoder successfully captures the underlying structure of the dataset while preserving good generalization ability and providing a meaningful compressed representation.

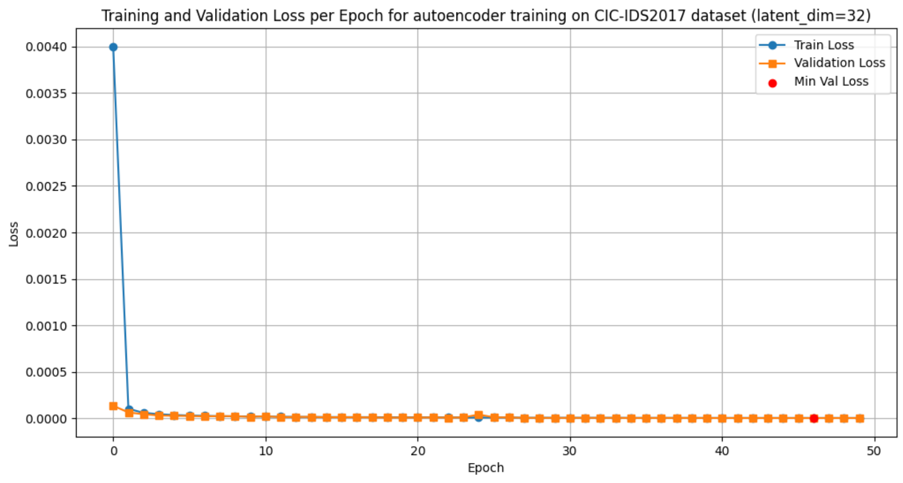

Figure 4.1: Training and validation loss per epoch for autoencoder training on CIC-IDS2017 dataset (latent dimension = 32).

The latent encoding of CIC-IDS2017 completes the preparation of this dataset and allows its integration with other datasets in a unified latent space

4.1.2 BoT-IoT

Introduction

The BoT-IoT dataset was created by the University of New South Wales (UNSW) and is widely used in research on IoT network intrusion detection. It contains realistic network traffic generated from an Internet of Things (IoT) testbed, capturing both normal device communications and a variety of cyberattacks. The dataset includes activities such as sensor data transmission, device-to-device communication, and standard network protocols, along with attack scenarios including denial of service (DoS), distributed denial of service (DDoS), reconnaissance, and data exfiltration.

The dataset provides structured flow-based features extracted from network traffic, covering packet-level, time-based, and statistical attributes. These features make BoT-IoT suitable for evaluating anomaly detection models in IoT environments, as it reflects heterogeneous device behaviors and diverse attack patterns.

For this solution, a reduced version corresponding to 5% of the full dataset, publicly available on the BoT-IoT website, is employed. This subset provides representative samples of both normal and attack traffic while significantly reducing computational requirements for training and evaluation. Table 4.3 summarizes the general parameters of this dataset before any preprocessing step applied.

Table 4.3: Parameters of dataset BoT-IoT

Parameters	**original**
number of features	45 (+1 Label)
number of instances	3668522
number of normals	477
number of attacks	3668045

As shown in Table 4.3, normal traffic represents only a very small fraction of the dataset, while attack traffic constitutes the vast majority of flows. This highly imbalanced distribution highlights the challenging nature of anomaly detection in the BoT-IoT dataset.

Table 4.4 shows the distribution of the dataset by attack type with the number of instances for each category.

Table 4.4: BoT-IoT Traffic Instances

Traffic	**Instances**
Normal	477
DDoS	1926624
DoS	1650260
Reconnaissance	91082
Theft	79

A detailed description of any preprocessing applied, including feature selection or cleaning operations, is presented in the following section.

Data Preparation

A 5% subset of the BoT-IoT dataset was used for this study. The selected CSV files were loaded and concatenated into a single dataset. Columns containing redundant, constant, or non-informative values were removed, and missing or infinite values were handled. Duplicate records were also eliminated to improve data quality.

For the target label, the existing attack field was used, already in numerical format (attack = 1, normal = 0). Features were separated from the target label, and any categorical attributes were encoded numerically using label encoding.

After these preprocessing steps, the dataset contains 44 features with only 2 lost features ('category' and 'subcategory'), while the total number of instances remains unchanged.

The dataset was split into training (70%), validation (15%), and test (15%) sets, using a stratified approach to preserve the ratio of normal and attack flows in each subset. Due to the highly imbalanced nature of BoT-IoT, the training set was further downsampled: the majority class (attacks) was randomly reduced to three times the size of the minority class (normal traffic) without replacement, and the resulting subset was shuffled. This created a more balanced dataset for model training while maintaining sufficient examples of attacks.

Before training the autoencoder, all features were scaled to the [0,1] range using Min-Max normalization. The scaling parameters were computed exclusively on the training set and then applied to the validation and test sets to avoid information leakage. After normalization, the dataset was ready for latent representation learning.

Latent Representation

Once the BoT-IoT dataset (5% subset) was preprocessed, encoded, and downsampled, the autoencoder was trained to learn a compact latent representation of the input flows. Several latent dimensions were tested, as described in the Hyperparameters Section, and the configuration with a latent dimension of 32 provided the best overall performance.

Figure 4.2 shows the training and validation loss across the epochs for this configuration. Both curves decrease steadily and remain closely aligned, indicating stable convergence without overfitting. Although the losses start at higher values due to the characteristics and imbalance of the BoT-IoT dataset, the gradual reduction demonstrates that the autoencoder effectively captures the underlying structure of the dataset, providing a meaningful latent representation suitable for integration with other datasets.

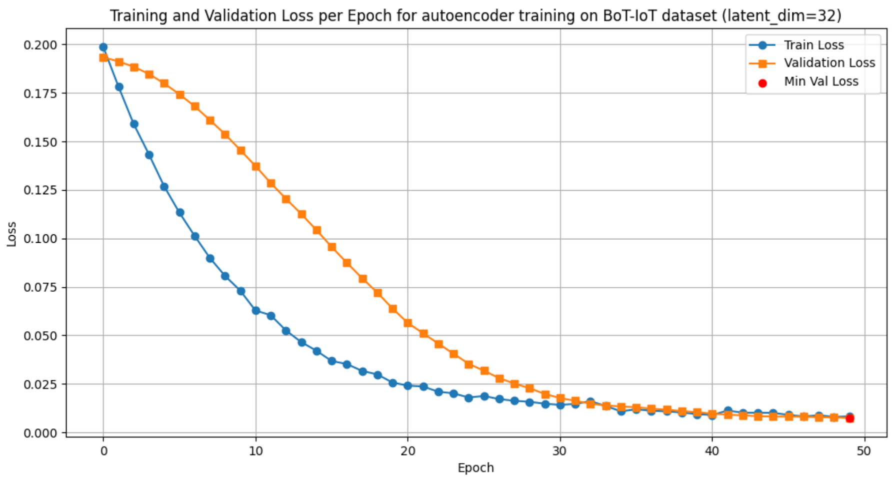

Figure 4.2: Training and validation loss per epoch for autoencoder training on BoT-IoT dataset (latent dimension = 32).

The latent encoding of BoT-IoT completes the preparation of this dataset and allows its integration with other datasets in a unified latent space.

4.1.3 UNSW-NB15

Introduction

The UNSW-NB15 dataset was developed by the Australian Centre for Cyber Security (ACCS) at the University of New South Wales (UNSW) and is one of the most widely used benchmarks for evaluating network intrusion detection systems. It contains realistic network traffic generated in a controlled cyber range environment, simulating both normal user activity and various modern attack scenarios. The captured traffic includes categories such as exploits, denial of service (DoS), reconnaissance, backdoors, and fuzzers, representing a diverse range of contemporary threats.

The dataset consists of flow-based records that include both continuous and categorical attributes describing the characteristics of network connections. These features cover aspects such as packet statistics, protocol behavior, and time-based measures. Owing to its realistic traffic composition and comprehensive labeling, UNSW-NB15 provides a robust foundation for training and evaluating anomaly detection models in general-purpose network environments.

For this solution, the full UNSW-NB15 dataset is employed in its original form, encompassing all 49 features and all recorded flows before any preprocessing is applied, as summarized in Table 4.5.

Table 4.5: Parameters of dataset UNSW-NB15

Parameters	**original**
number of features	48 (+1 Label)
number of instances	2540047
number of normals	2218764
number of attacks	321283

As shown in Table 4.5, normal traffic constitutes the majority of the dataset, while attack traffic represents a minor proportion of the total records. Although the imbalance is less pronounced compared to BoT-IoT, it still presents challenges for anomaly detection due to the relatively limited number of attack samples.

Table 4.6 shows the distribution of the dataset by attack type with the number of instances for each category.

Table 4.6: UNSW-NB15 Traffic Instances

Traffic	**Instances**
BENIGN	2218764
Generic	215481
Exploits	44525
Fuzzers	24246
DoS	16353
Reconnaissance	13987
Analysis	2677
Backdoor	1795
Shellcode	1511
Backdoors	534
Worms	174

A detailed description of any preprocessing applied, including feature selection or cleaning operations, is presented in the following section.

Data Preparation

The UNSW-NB15 dataset was preprocessed to ensure consistency, remove noise, and prepare it for model training. All CSV files were first loaded and concatenated into a single dataset. Columns with constant values, missing entries, or infinite values were removed, and duplicate records were discarded. The attack_cat column was dropped, while the Label column was kept as the target, where 0 indicates normal traffic and 1 indicates attack traffic.

Features were separated from the target label, and categorical attributes were encoded numerically using label encoding, allowing all features to be used in the autoencoder training.

After these preprocessing steps, the dataset retained 48 features and 1,022,132 instances, with 1,008,919 benign and 13,212 attack flows. This slight reduction compared to the original dataset (Table 4.5) is mainly due to removal of duplicates, constant features, and incomplete records.

The dataset was then split into training (70%), validation (15%), and test (15%) sets using stratified sampling to preserve the proportion of benign and attack flows. No downsampling was applied, as the UNSW-NB15 dataset is already dominated by normal traffic, in contrast to BoT-IoT.

Before training the autoencoder, all features were normalized to the [0,1] range using Min-Max scaling. Scaling parameters were computed exclusively on the training set and applied to the validation and test sets to avoid information leakage. After normalization, the dataset was ready for latent representation learning.

Latent Representation

Once the UNSW-NB15 dataset was preprocessed and normalized, the autoencoder was trained to learn a compact latent representation of the input flows. Several latent dimensions were tested, as described in the Hyperparameters Section, and the configuration with a latent dimension of 32 provided the best overall performance.

Figure 4.3 shows the training and validation loss across the epochs. Both curves steadily decrease to very low values and remain close to each other, indicating stable convergence without overfitting. This confirms that the autoencoder effectively captures the underlying structure of the dataset and produces a meaningful compressed representation.

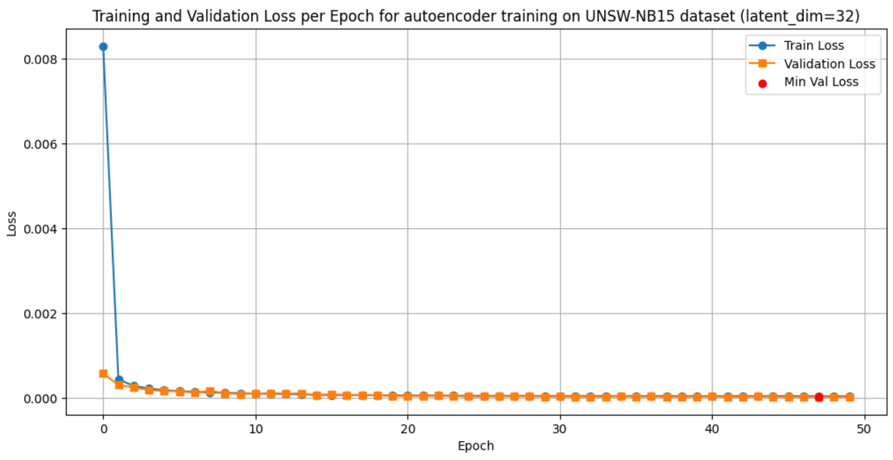

Figure 4.3: Training and validation loss per epoch for autoencoder training on UNSW-NB15 dataset (latent dimension = 32).

The latent encoding of UNSW-NB15 completes the preparation of this dataset and allows its integration with other datasets in a unified latent space.

4.1.4 Unification

After the preprocessing and latent representation stages of each dataset, the encoded training, validation, and test sets were combined into a unified dataset to enable joint model training across heterogeneous network environments. Specifically, the latent representations generated by the autoencoder were loaded for each dataset (CIC-IDS2017, BoT-IoT, and UNSW-NB15), preserving their respective binary labels for benign and attack traffic.

The unification process consisted of concatenating the encoded features and corresponding labels along the sample dimension for each data partition (training, validation, and test). This operation resulted in three integrated subsets:

- **Unified training set** — obtained by merging the latent features from the three datasets and shuffling the samples to ensure randomized distribution.
- **Unified validation set** — created by concatenating the validation subsets of each dataset, maintaining class proportions.
- **Unified test set** — constructed in the same way, allowing the evaluation of model generalization across mixed traffic scenarios.

Table 4.7 summarizes the size of the unified dataset after concatenation of the encoded representations from all sources.

Table 4.7: Unified dataset composition after feature encoding and concatenation.

Subset	**Number of Instances**
Training set	1,949,657
Validation set	1,081,719
Test set	2,163,437
Total	**5,194,813**

As shown in Table 4.7, the unified dataset consists of over five million samples derived from the integration of multiple traffic sources. This diverse collection of flows captures a wide range of normal and malicious behaviors, providing a solid foundation for evaluating anomaly detection models under heterogeneous network conditions.

The resulting unified dataset integrates diverse traffic patterns from enterprise, IoT, and hybrid network environments, providing a comprehensive benchmark for anomaly detection. This unified representation serves as the foundation for the subsequent model training stage, where a shared classifier is optimized to detect anomalies consistently across heterogeneous network domains.

4.2 Evaluations

The evaluation phase represents the process of assessing how well the proposed model generalizes beyond the training data, by verifying its effectiveness and limitations under realistic conditions on a test set. This section presents the methodology adopted for the evaluation. First, the experimental setup is described, including the two evaluation modes and the hyperparameter configuration. Then, the metrics used to assess performance are detailed.

4.2.1 Experimental Setup

The evaluation of the proposed model is conducted under three main aspects:

- **Evaluation modes**
- **Decision thresholds**
- **Hyperparameter configuration**

Evaluation modes

Two complementary setups are considered. In the *dataset-specific* mode, which serves as a baseline, the model is trained and evaluated on each individual dataset to assess its performance when tailored to specific traffic patterns and attack characteristics. In the *unified* mode, all datasets are combined during training to test the model's ability to generalize across heterogeneous sources, providing information on its robustness and adaptability to different network behaviors.

Decision thresholds

Instead of adopting a fixed threshold (e.g., 0.5), the classification threshold is chosen based on the evaluation mode. In the *dataset-specific* configuration, the decision threshold was optimized individually for each dataset using its test set. This choice ensures that the model achieves the best possible metrics on each dataset, effectively providing an upper-bound baseline against which other settings can be compared. Conversely, in the *unified* configuration, a single threshold was determined on the validation set of the unified dataset and then applied consistently across all test subsets. This strategy provides a more realistic evaluation of the model's robustness, as it enforces a common decision boundary across heterogeneous network traffic.

Hyperparameter Configuration

Hyperparameters are settings that govern the learning process of a machine learning model but are not updated during training. They define aspects such as the network architecture, learning rate, batch size, and activation functions, influencing how effectively the model can learn from data and generalize to unseen examples. Choosing appropriate hyperparameters is crucial for achieving optimal performance, balancing convergence speed, stability, and model capacity. In the proposed model, the configuration is specified separately for the autoencoder used for feature extraction and for the MLP classifier. Table 4.8 summarizes the set of hyperparameters used for the autoencoder and MLP training for the proposed model. Other hyperparameters are detailed in Section 3.2.4.

Table 4.8: Hyperparameter configuration of the proposed model.

Component	**Hyperparameters**
Autoencoder	• Latent dimension tested: [4,8,16,32,64] • Batch size: 1024 • Epochs: 50 • Initial learning rate: 0.001
MLP Classifier	• Hidden layers: 512 • Num layers: 4 • Batch size: 1024 • Epochs: 30 • Initial learning rate: 0.0001

4.2.2 Evaluation Metrics

To assess the effectiveness of the proposed model, several evaluation metrics are considered. These metrics are widely used in intrusion detection tasks and are particularly suited for imbalanced datasets, providing a comprehensive view of classification performance. In the proposed solution, they are used to evaluate the final performance of the model on the test sets.

Basic Definitions In binary classification, the following quantities are used to evaluate performance:

- **TP (True Positives)**: instances correctly classified as positive (attack).
- **TN (True Negatives)**: instances correctly classified as negative (normal).
- **FP (False Positives)**: instances incorrectly classified as positive.
- **FN (False Negatives)**: instances incorrectly classified as negative.

These definitions form the basis for calculating accuracy, precision, recall, F1-score, and other evaluation metrics.

Accuracy Accuracy measures the proportion of correctly classified instances among the total number of samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precision Precision quantifies the proportion of correctly predicted positive instances among all predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

Recall Recall measures the proportion of actual positive instances correctly identified by the model:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

F1-score The F1-score is the harmonic mean of precision and recall, providing a balanced measure of classification performance:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

ROC-AUC The Receiver Operating Characteristic – Area Under the Curve (ROC-AUC) summarizes the trade-off between true positive rate and false positive rate at different thresholds:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{4.5}$$

where

$$\text{TPR} = \text{Recall}, \quad \text{FPR} = \frac{FP}{FP + TN}. \tag{4.6}$$

These metrics collectively provide a detailed assessment of model performance. For evaluating and comparing configurations such as different latent dimensions, the F1-score was used as the primary reference metric due to its balance between precision and recall.

The following section presents the experimental results obtained using these metrics, highlighting the model's performance across different datasets and configurations.

4.3 Results

4.3.1 Individual Dataset Results

In this section, the model is trained and evaluated separately on each dataset, using the corresponding latent representations produced by the autoencoders to ensure consistency with the proposed approach. The aim is to compare the performance achieved when relying on a single dataset—thus yielding a dataset-specific model—with the results obtained in the unified setting, where the model is expected to generalize across heterogeneous data sources. This comparison also allows to assess the performance differences on the same dataset under the two training strategies. For each dataset, the F1-scores obtained by varying the size of the latent space are reported in Figure 4.4. As shown, the optimal latent dimension differs depending on the dataset.

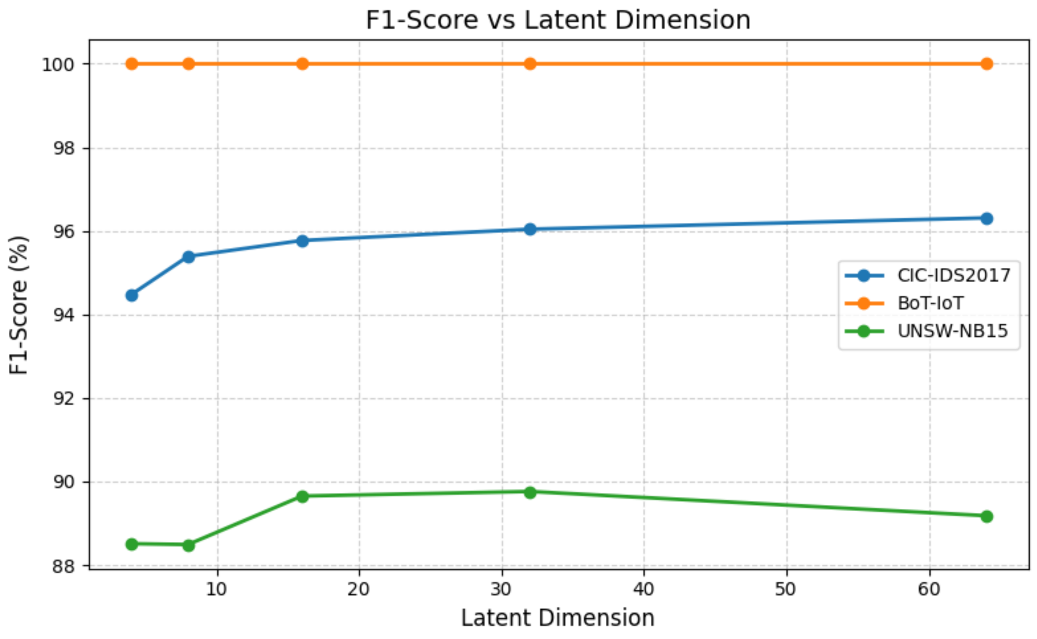

Figure 4.4: F1-score on each dataset across different latent space dimensions.

Table 4.9 summarizes the F1-scores obtained for each dataset across different latent space dimensions.

Table 4.9: F1-scores for each dataset and latent space dimension (numerical values corresponding to Figure 4.4).

Latent Dimension	CIC-IDS2017	BoT-IoT	UNSW-NB15
4	94.47	100	88.51
8	95.39	100	88.49
16	95.77	100	89.65
32	96.04	100	89.76
64	96.31	100	89.18

As shown in Figure 4.4, the three datasets present distinct behaviors with respect to the latent space dimension:

- **BoT-IoT** - exhibits a constant F1-score of 100 across all latent space sizes, indicating that this dataset is linearly separable even in very small latent spaces. This behavior is likely due to its strong class imbalance and simpler feature distribution.

- **CIC-IDS2017** - shows a steadily increasing trend, from 94.47 at 4 dimensions up to 96.31 at 64 dimensions, suggesting that larger latent spaces help capture more complex traffic patterns.

- **UNSW-NB15** - achieves its best performance at 32 dimensions (89.76) and slightly decreases afterwards, indicating that excessive dimensionality may introduce redundancy or noise, resulting in minor drops in F1-score.

These results show that the optimal latent space dimension is dataset-dependent: CIC-IDS2017 benefits from larger representations, UNSW-NB15 favors a more compact latent space, while BoT-IoT remains largely unaffected, highlighting the challenge of designing a unified model that generalizes across heterogeneous datasets.

To complement the F1-score analysis, Figures 4.5, 4.6, and 4.7 report the training and validation loss curves when the model is trained separately on each dataset with a latent dimension of 32. These plots allow for a direct comparison with the unified setting, highlighting differences in convergence speed and generalization behavior.

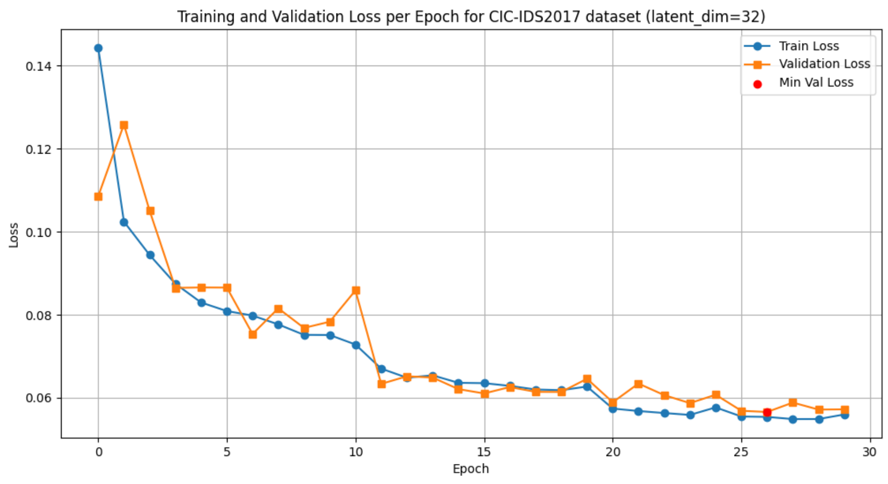

Figure 4.5: Training and validation loss per epoch for the CIC-IDS2017 dataset (latent dimension = 32).

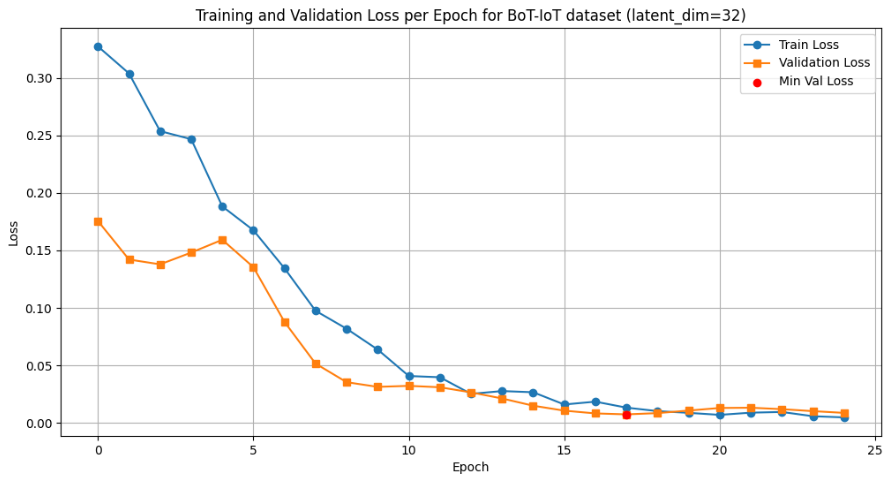

Figure 4.6: Training and validation loss per epoch for the Bot-IoT dataset (latent dimension = 32).

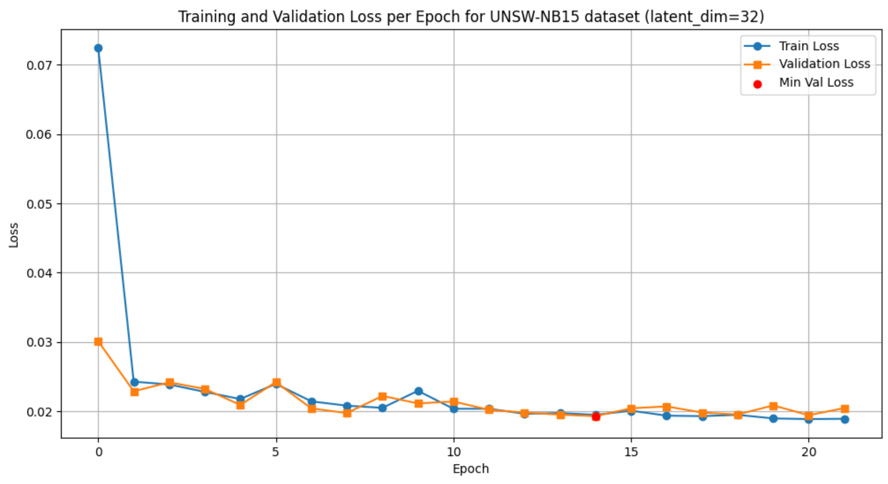

Figure 4.7: Training and validation loss per epoch for the UNSW-NB15 dataset (latent dimension = 32).

The loss curves highlight clear differences in the training dynamics across the three datasets. For **CIC-IDS2017**, both training and validation losses decrease steadily and remain close throughout the epochs, with the validation loss showing minor oscillations but ultimately stabilizing. This indicates good convergence and balanced generalization, consistent with the gradual F1-score improvement observed as the latent dimension increases.

In the case of **BoT-IoT**, convergence is extremely rapid: the validation loss drops sharply within the first epochs and approaches near-zero values, reflecting the relative simplicity of the dataset. The small gap between training and validation confirms the absence of overfitting, in line with the constant F1-score of 100 obtained across all latent space sizes.

Conversely, **UNSW-NB15** presents a more challenging scenario. While both training and validation losses converge to relatively low values, the validation curve remains slightly higher and shows more fluctuations compared to the other datasets. This behavior suggests a tendency to slight overfitting, which also explains the lower F1-scores and the sensitivity to the choice of latent dimension.

Overall, these results confirm that single-dataset training produces distinct convergence behaviors depending on dataset characteristics: smoother and balanced for CIC-IDS2017, extremely fast for BoT-IoT, and more unstable for UNSW-NB15.

Table 4.10 summarizes the classification metrics obtained with a latent dimension of 32 for each dataset.

Table 4.10: Evaluation metrics for binary classification on each dataset with latent dimension = 32.

Test Dataset	Accuracy	Precision	Recall	F1-score	ROC-AUC
CIC-IDS2017	98.65	95.35	96.73	96.04	99.92
BoT-IoT	100	100	100	100	99.99
UNSW-NB15	99.72	84.45	95.79	89.76	99.96

To further analyze the classification performance at the class level, Figures 4.8, 4.9, and 4.10 present the confusion matrices for each dataset complementing the numerical metrics in Table 4.10, providing insight into which classes are most often misclassified

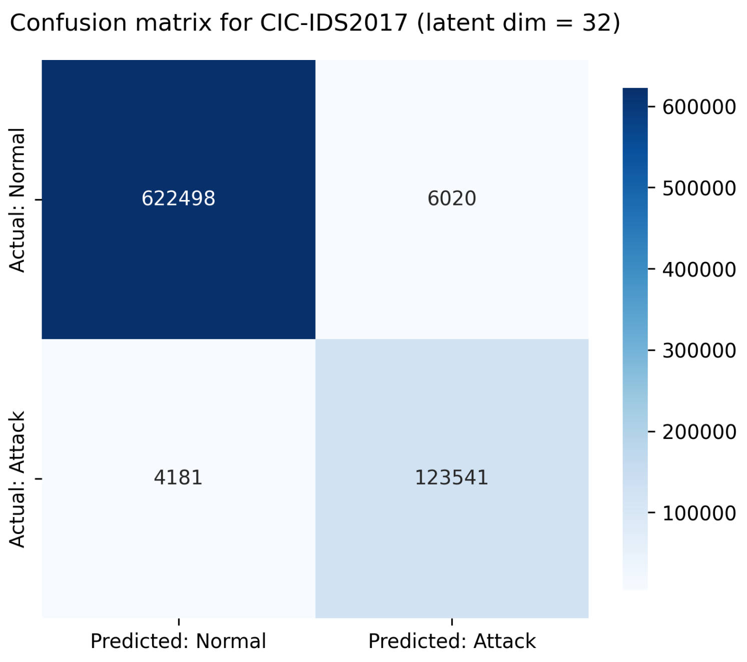

Figure 4.8: Confusion matrix for CIC-IDS2017 dataset (latent dimension = 32).

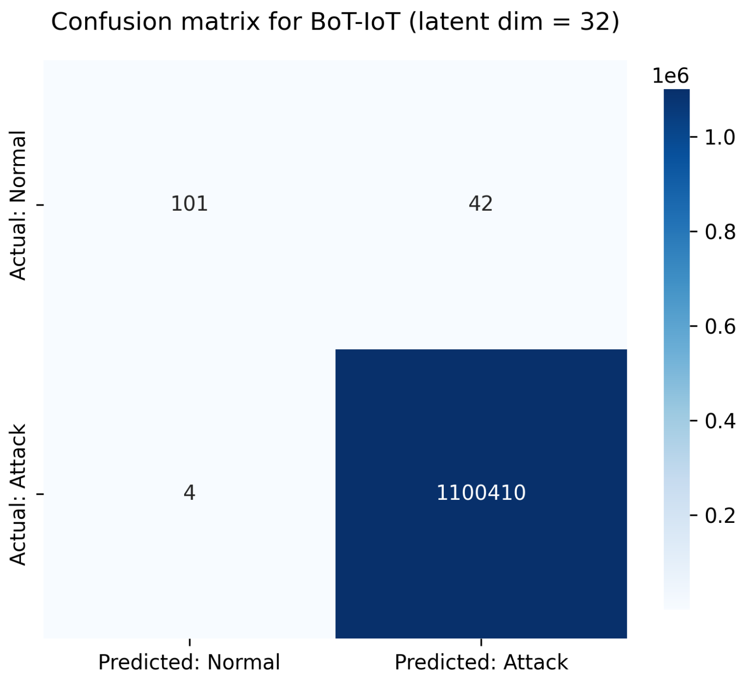

Figure 4.9: Confusion matrix for BoT-IoT dataset (latent dimension = 32).

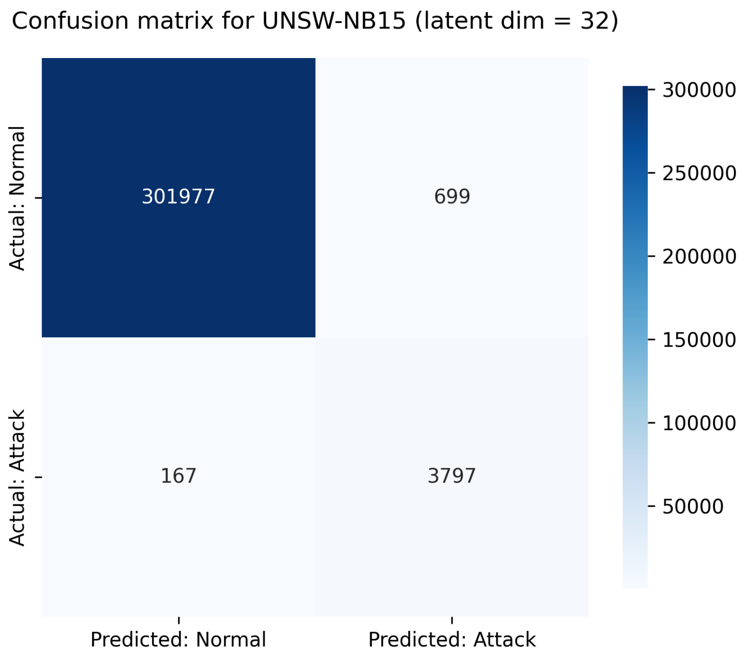

Figure 4.10: Confusion matrix for UNSW-NB15 dataset (latent dimension = 32).

As shown in Figures 4.8, 4.9, and 4.10, the confusion matrices provide a detailed view of the model's performance at the class level. For CIC-IDS2017, the model achieves a high true positive rate for both normal and attack traffic, with relatively few misclassifications, reflecting its overall strong F1-score. In BoT-IoT, misclassifications are negligible, highlighting the dataset's linearly separable nature and the model's ability to capture its patterns effectively. For UNSW-NB15, most normal instances are correctly classified, but a small number of attacks are misclassified as normal, consistent with the slightly lower F1-score observed in the metrics. These matrices complement the quantitative metrics by revealing which classes contribute most to residual errors and providing insights into dataset-specific challenges.

4.3.2 Unified Dataset Results

Building on the previous experiments on individual datasets, this section investigates the performance of a model trained on a unified dataset. By combining all datasets, the model is exposed to a more diverse and heterogeneous set of network traffic patterns. To assess its effectiveness, performance is first analyzed across different latent space sizes using the combined test set of CIC-IDS2017, BoT-IoT, and UNSW-NB15. This analysis provides insights into how well a single model can generalize across multiple datasets, and Figure 4.11 illustrates the trend of the F1-score with respect to the latent dimension.

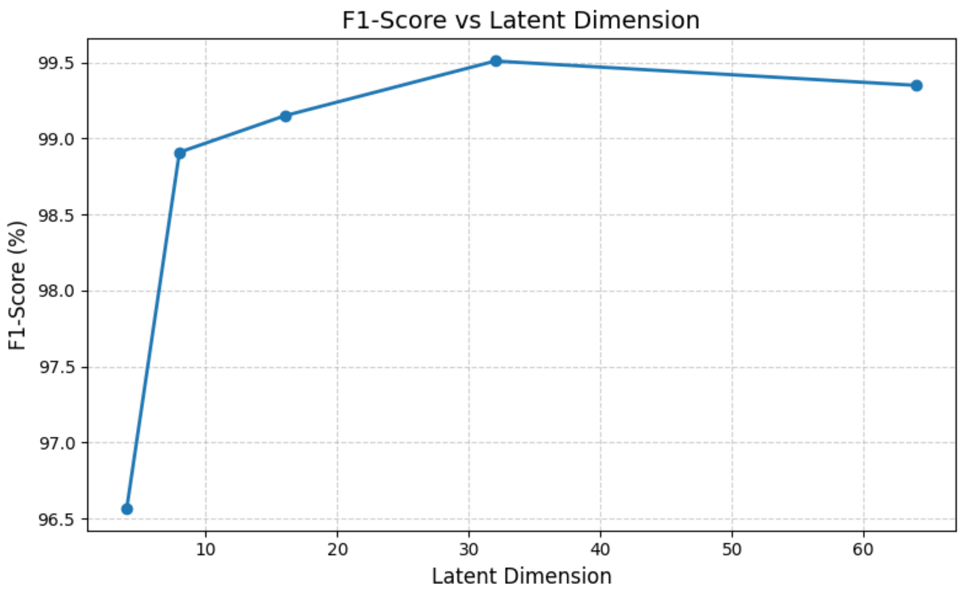

Figure 4.11: F1-score on the unified dataset across different latent space dimensions.

As shown in Figure 4.11, the F1-score increases with the latent space size up to 32 dimensions (F1 = 99.51) after which it decreases at 64 dimensions.The detailed values for each latent space size are reported in Table 4.11.

Table 4.11: F1-scores of unified dataset across latent space dimensions (numerical values corresponding to Figure 4.11).

Latent Dimension	F1-score
4	96.57
8	98.91
16	99.14
32	99.51
64	99.35

Overall,this trend suggests that very small latent spaces are not sufficient to capture the relevant patterns, while excessively large ones may introduce noise or redundancy, leading to a marginal drop in performance beyond the optimal size.

Figure 4.12 shows the training and validation loss curves for the unified dataset with latent dimension = 32. The training loss decreases steadily over the epochs, while the validation loss exhibits minor fluctuations but generally trends downward, indicating stable convergence. This explains the high overall F1-score and confirms that the model generalizes well without overfitting.

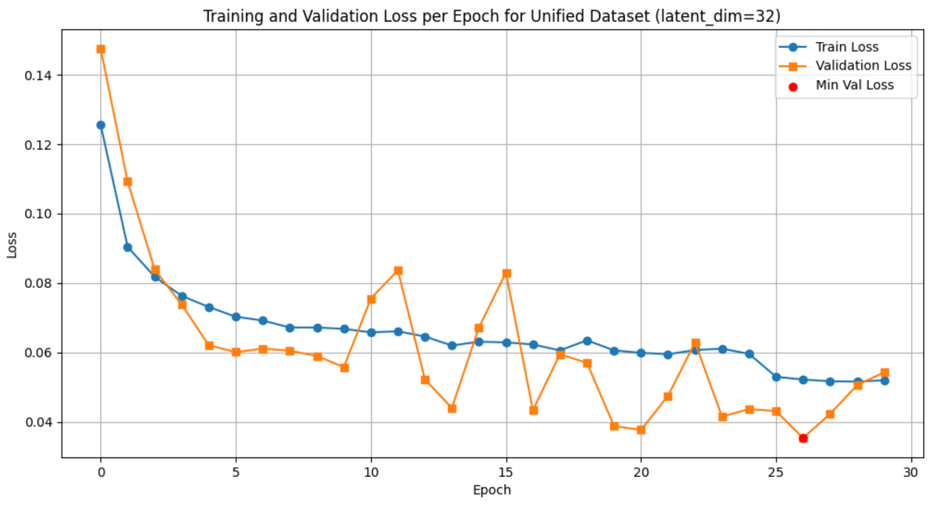

Figure 4.12: Training and validation loss per epoch for the unified dataset (latent dimension = 32).

Table 4.12 presents the performance metrics for the unified dataset using latent dimension = 32, which was identified as the optimal value in the F1 analysis (Figure 4.11). Results are reported both for each individual dataset and for the overall test set.

Table 4.12: Evaluation metrics for binary classification on the unified dataset with latent dimension = 32.

Test Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC-AUC**
CIC-IDS2017	98.68	95.28	96.97	96.12	99.90
BoT-IoT	99.89	100	99.89	99.94	99.99
UNSW-NB15	99.74	86.88	94.35	90.46	99.97
Total Test	99.44	99.46	99.57	99.51	99.99

The results show that the unified model achieves strong generalization across heterogeneous datasets, with an overall F1-score of 99.51 and ROC-AUC close to 1.0. Performance is nearly perfect on BoT-IoT, partly reflecting the strong class imbalance that makes this dataset dominate the unified training. CIC-IDS2017 also records high precision and recall, while UNSW-NB15 proves more challenging, with a lower F1-score (90.46) mainly due to reduced precision. Nevertheless, the unified setting mitigates the dominance of BoT-IoT, enabling the model to retain very high overall performance while preserving robustness across all datasets.

To further analyze the classification performance at the class level, Figures 4.13, 4.14, and 4.15 present the confusion matrices for each dataset in the unified setting, complementing the numerical metrics reported in Table 4.12.

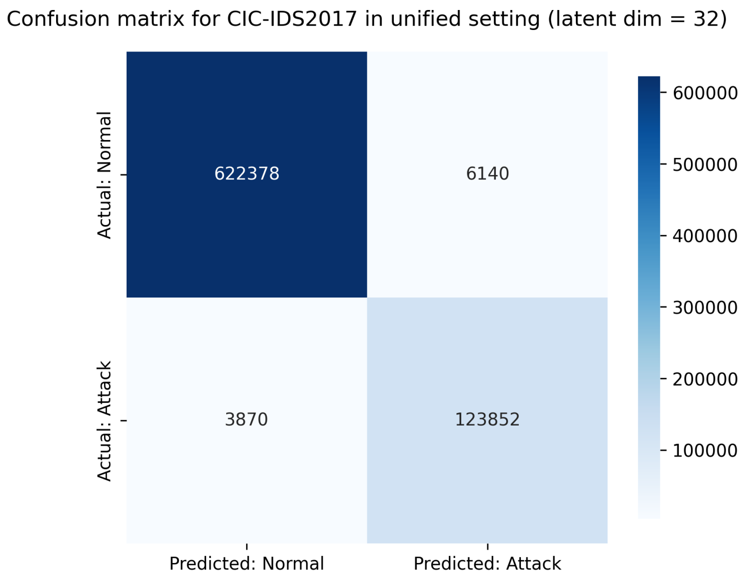

Figure 4.13: Confusion matrix for CIC-IDS2017 dataset in unified setting (latent dimension = 32).

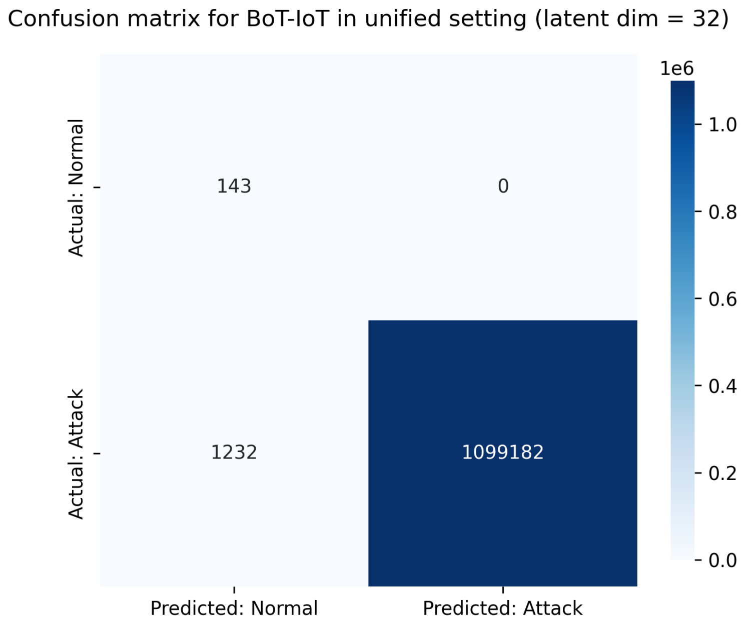

Figure 4.14: Confusion matrix for BoT-IoT dataset in unified setting (latent dimension = 32).

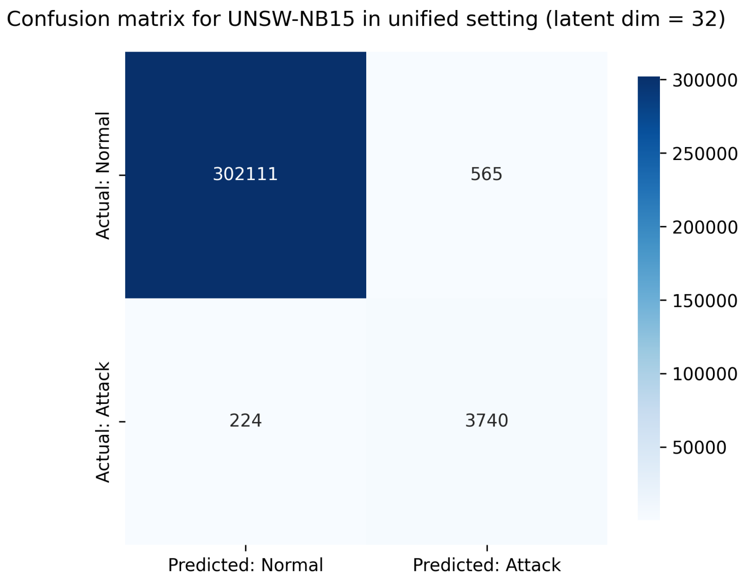

Figure 4.15: Confusion matrix for UNSW-NB15 dataset in unified setting (latent dimension = 32).

The confusion matrices for the unified model confirm its strong classification capabilities across all datasets. CIC-IDS2017 and BoT-IoT exhibit very few misclassifications, reflecting the model's ability to generalize effectively despite dataset heterogeneity. For UNSW-NB15, a slightly higher number of misclassified samples is observed, consistent with its lower F1-score, but the overall performance remains robust. These results highlight that the unified model maintains high discrimination between normal and attack traffic across diverse datasets.

4.3.3 Feature Importance

After obtaining the results from the trained model, Explainable AI (XAI) techniques were applied to analyze how the features of the different datasets influenced the binary classification. Based on the outcomes provided by XAI, the possibility of employing this approach as a feature selection method was also investigated. This section describes the adopted procedure and the results obtained.

SHAP

SHAP (SHapley Additive exPlanations) is an XAI technique for interpreting machine learning models. It is based on the principles of cooperative game theory, where each feature is considered a "player" contributing to the model's output. SHAP assigns a value (SHAP value) to each feature that quantifies its impact on the prediction, indicating whether it increases or decreases the output. These values are computed by considering all possible combinations of features, providing a theoretically sound measure of each feature's contribution. This approach allows features to be ranked according to their influence, making it possible to identify the most essential variables. By providing a transparent and consistent measure of feature importance, SHAP supports interpretable analyses and is particularly suitable for complex models such as neural networks.

Methodology

In this work, SHAP was applied to the trained Autoencoder-MLP model to evaluate the importance of the features extracted from each dataset, supporting the identification of the most informative variables for anomaly detection in network traffic. The goal was to investigate whether the results obtained from the explainability analysis could be used to select the most relevant features, thus reducing the dimensionality of the input data and assessing the potential of XAI as a feature selection method.

The analysis was performed separately for each dataset using the same input data of the previous Autoencoder. A wrapper consisting of the previously trained autoencoder and the previous unified MLP model was used as the evaluation model.

To compute the SHAP values, a GradientExplainer was employed, using the model's gradients to estimate the contribution of each feature to the prediction outcome. A random subset of 200 samples was selected both as background data and as the evaluation set to simplify the computation. The resulting SHAP values were aggregated by computing the mean absolute value across all samples, providing a measure of global feature importance. These values were used to generate global visualizations, such as bar and beeswarm plots, and to rank the features according to their relative influence on the model's output to perform subsequently feature selection. Based on the feature ranking obtained, the cumulative importance of the features was computed as the normalized cumulative sum of their mean absolute SHAP values, representing the proportion of total feature importance captured by the top-ranked variables. Features were then selected according to fixed thresholds specific to each dataset. For instance, a 95% threshold was applied to CIC-IDS2017 due to its higher number of features, whereas a 90% threshold was used for the other datasets. This process identified the subset that retained most of the model's explainability while reducing input dimensionality. These reduced feature sets were subsequently employed to retrain and evaluate the model, assessing the effect of SHAP-based feature selection on overall performance. In the next part, the results obtained from this analysis are presented, including both those derived from the original model and those achieved after applying SHAP-based feature selection.

Results

Two types of SHAP visualizations were used to analyze feature importance and the model's behavior. The summary bar plot provides an overall ranking of features based on the mean absolute SHAP values, indicating the average contribution of each variable to the model's predictions. The summary beeswarm plot offers a compact and informative overview of how the most important features affect the model's output. Each dot represents a single data instance for a given feature. The horizontal position of the dot corresponds to the SHAP value, showing the effect of the feature on the prediction (increasing to the right, decreasing to the left), while the vertical axis groups the dots by feature. Points accumulate along each row, showing the density of similar SHAP values. The color of the dots reflects the original feature value (blue for low, red for high), enabling observation of patterns between feature values and their influence on model predictions. For each dataset, the bar and beeswarm plots are displayed together in the same figure for easier comparison.

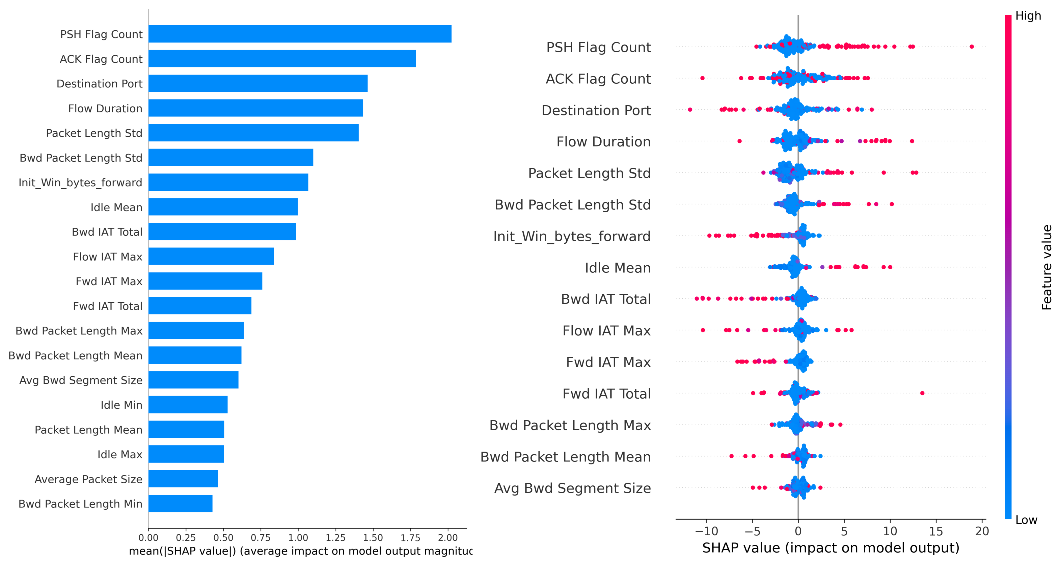

Figure 4.16: summary bar (L) and summary beeswarm (R) for CIC-IDS2017 dataset obtained using SHAP

As shown in Figure 4.16, the SHAP analysis on the CIC-IDS2017 dataset highlights that the features with the greatest influence on the model's output are `PSH Flag Count` and `ACK Flag Count`, highlighting the importance of TCP flag-related attributes in anomaly detection. Observing the beeswarm plot, it can be seen that high values of `PSH Flag Count` contribute more to the model's prediction, while for `ACK Flag Count` the contribution is more evenly distributed between normal and anomalous predictions. Other features, such as `Idle Mean` and `Bwd IAT Total`, exhibit clearer and more consistent patterns, showing stronger correlations between feature values and their impact on the model's output.

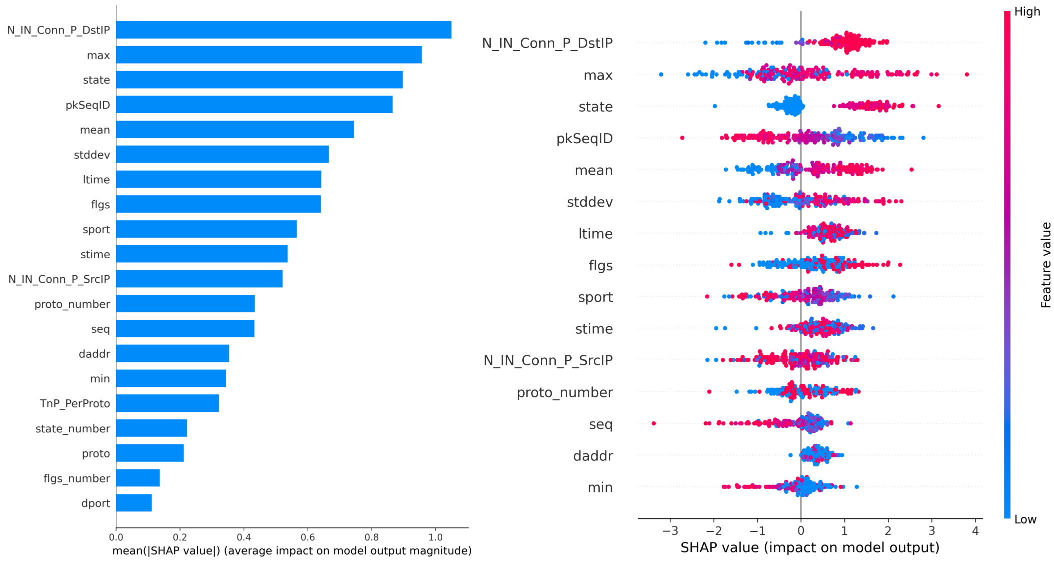

Figure 4.17: summary bar (L) and summary beeswarm (R) for BoT-IoT dataset obtained using SHAP

The SHAP analysis on the BoT-IoT dataset, shown in Figure 4.17, highlights `N_IN_Conn_P_DstIP`, `max`, and `state` as the most important features, indicating the relevance of connection statistics and state-related attributes in anomaly detection. Looking at the beeswarm plot, a clear pattern can be observed for `N_IN_Conn_P_DstIP`, where high feature values are associated with a positive SHAP impact and low values with a negative impact. A similar behavior can be seen for the `state` feature, while the `max` feature appears more evenly distributed across the SHAP value range. Finally, an interesting observation concerns the `daddr` feature, which shows a single point with a negative impact, possibly corresponding to a specific destination address strongly associated with normal traffic.

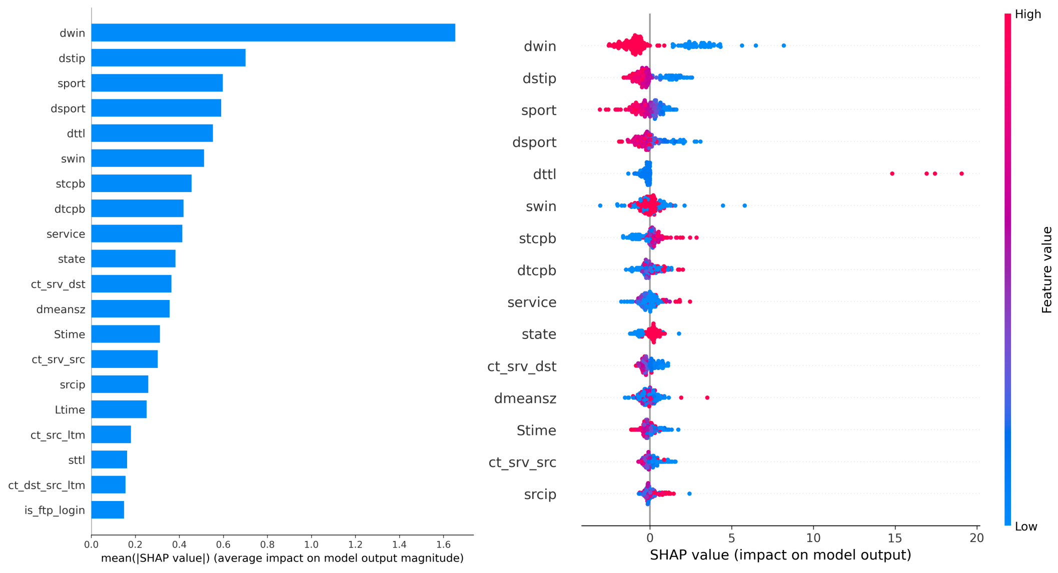

Figure 4.18: summary bar (L) and summary beeswarm (R) for UNSW-NB15 dataset obtained using SHAP

Finally, the SHAP analysis on the UNSW-NB15 dataset, shown in Figure 4.18, highlights a single feature, `dwin`, as clearly more influential than the others, with a large gap in the bar plot. Observing the beeswarm plot, it can be seen that low values of `dwin` are associated with a positive impact on the model's prediction, while high values push the prediction toward normal traffic. An additional interesting pattern appears in `dttl`, which shows an isolated point with a negative impact, similar to the behavior observed for the `daddr` feature in BoT-IoT. These observations suggest that `dwin` plays a dominant role in the model's decision process for UNSW-NB15, while other features contribute more sporadically or in specific conditions.

The SHAP visualizations described above were used to guide feature selection for each dataset. Two configurations were evaluated, both based on the 95% cumulative importance threshold:

- **First configuration** — A fixed number of 20 features was selected for all datasets, corresponding to the smallest number of features required by any dataset to reach the cumulative threshold.

- **Second configuration** — The number of features was chosen individually for each dataset, resulting in 37 features for CIC-IDS2017, 20 features for BoT-IoT, and 24 features for UNSW-NB15.

Both configurations were applied using the same unified model with unchanged hyperparameters, including a latent dimension of 32 for the autoencoder, while only the number of input features varied according to the SHAP-based feature selection.

Table 4.13: Evaluation metrics of the unified model on each dataset using SHAP-based feature selection (first configuration).

Test Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC-AUC**
CIC-IDS2017	97.70	91.04	95.81	93.37	99.75
BoT-IoT	99.58	100	99.58	99.79	99.97
UNSW-NB15	99.69	81.95	97.33	88.98	99.96
Total Test	98.94	98.96	99.18	99.07	99.96

Table 4.14: Evaluation metrics of the unified model on each dataset using SHAP-based feature selection (second configuration).

Test Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC-AUC**
CIC-IDS2017	98.64	95.11	96.91	96.00	99.90
BoT-IoT	99.88	100	99.88	99.94	100
UNSW-NB15	99.75	86.07	96.19	90.85	99.97
Total Test	99.43	99.43	99.56	99.50	99.99

The results in Tables 4.13 and 4.14 show that SHAP-based feature selection preserves strong overall performance across all datasets. CIC-IDS2017 exhibits the most noticeable performance drop when the feature count is reduced, reflecting its larger proportion of removed features. In contrast, BoT-IoT and UNSW-NB15 remain largely stable, with UNSW-NB15 even showing a slight improvement in the second configuration. This trend indicates that datasets with a high number of original features are more sensitive to dimensionality reduction, while the unified model maintains robust predictive capability overall, confirming SHAP's effectiveness as a feature selection method.

Building on the promising results obtained with the first SHAP-based configuration, where each dataset was individually reduced to 20 features, an additional experiment was conducted to evaluate whether a single autoencoder could effectively replace the three separate ones. For this purpose, each dataset was reduced to the selected features and then combined into a unified dataset. A single autoencoder with latent dimension 32 was trained on it, and the resulting encoded representation was used to train the MLP classifier.

Table 4.15: Evaluation metrics for binary classification using a single autoencoder with SHAP-based selection on the unified dataset with latent dimension = 32.

Test Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC-AUC**
CIC-IDS2017	97.71	90.22	96.94	93.46	99.77
BoT-IoT	99.37	100	99.37	99.68	99.96
UNSW-NB15	99.73	86.96	93.19	89.97	99.96
Total Test	98.84	98.87	99.10	98.98	99.96

The results in Table 4.15 show that using a single autoencoder for the unified dataset achieves performance very close to that obtained with separate autoencoders. Overall metrics are slightly lower in some cases (e.g., precision on CIC-IDS2017) but remain strong across all datasets. This suggests that a shared latent representation can capture the essential features for anomaly detection while reducing the complexity of training multiple autoencoders.

Chapter 5

Conclusions

5.1 Summary

This thesis addressed the challenge of achieving model generalization across heterogeneous datasets in network anomaly detection. Modern intrusion detection datasets exhibit substantial differences in feature sets, traffic distributions, and attack classification, making cross-dataset learning difficult and limiting the generalization capability of machine learning models.

To address these challenges, a separate AutoEncoder was used for each dataset to map the data into a 32-dimensional latent space. These latent spaces were then aligned to form a unified representation, enabling the construction of a single classification model trained on the integrated data. The system demonstrated strong and consistent performance across all datasets, confirming the effectiveness of latent-space integration as a solution to dataset heterogeneity.

Furthermore, SHAP-based explainability was employed to analyze feature importance and identify the most influential features. This analysis guided a feature selection process that reduced the dimensionality of the unified representation. The results obtained using the selected features were very close to those achieved with the full AutoEncoder representation on the unified dataset, demonstrating that SHAP-based feature selection preserves the model's high detection performance while enabling a more compact representation, confirming the effectiveness of this approach.

5.2 Implications

The results obtained in this thesis have several important implications for network anomaly detection. First, the use of AutoEncoders to project traffic into a shared latent space enables the harmonization of heterogeneous data sources while preserving the structure and semantics of the original traffic. This demonstrates the feasibility of building intrusion detection models capable of operating across diverse environments, remaining robust even when exposed to variations in traffic patterns or attack types.

Additionally, the incorporation of SHAP-based analysis shows that explainability can play a practical role beyond model interpretation. Using SHAP as a feature selection method makes it possible to reduce dataset dimensionality without sacrificing detection performance. This is particularly important for improving confidence in intrusion detection systems, thanks to the transparency provided by SHAP's feature importance analysis.

Moreover, the results observed with the single AutoEncoder configuration, as shown in Table 4.15 confirm that SHAP-based feature selection can support compact and unified representations across multiple datasets. This highlights the flexibility of the proposed workflow and its ability to maintain strong detection performance even in a simplified configuration.

In conclusion, the combined use of AutoEncoders and SHAP-based feature selection enables the development of a classification model that can detect different types of attacks with high performance while also providing interpretability through feature importance analysis.

5.3 Limitations

Although the proposed approach demonstrates strong performance, some limitations must be considered when interpreting the results.

First, although it is possible to use a single AutoEncoder leveraging SHAP for feature selection and achieving strong performance, the dataset-specific configuration currently performs slightly better. This indicates that separate AutoEncoders may provide more stable and well-separated latent spaces, which could limit scalability when integrating additional datasets.

Second, managing highly heterogeneous datasets remains a fundamental challenge. Since each AutoEncoder is trained independently on a different dataset, the meaning of each latent dimension may vary across datasets. Although all models encode the input traffic into a 32-dimensional latent space, there is no guarantee that a given dimension represents the same underlying concept in all datasets. This lack of consistent semantics can limit the interpretability of the unified representation and may affect the robustness and generalization of the model when applied to new or structurally different datasets.

Third, although SHAP is useful for identifying important latent features, it has inherent limitations. Its results can vary depending on model initialization, the choice of background dataset, or the specific model used. Additionally, calculating SHAP values can be computationally intensive for large datasets or complex models. Finally, since latent dimensions are abstract representations, even highly ranked features may not correspond directly to interpretable network behaviors, limiting the practical interpretability of the feature selection process.

Fourth, the models used in this work, including the AutoEncoders and the MLP classifier, are relatively simple. While they achieve strong performance, it is possible that more complex architectures could further improve detection metrics or better capture complex relationships in the data. This represents a limitation in terms of exploring the full potential of more sophisticated modeling approaches.

5.4 Future Research

Based on the framework developed in this thesis, several improvements and further research opportunities can be explored. A first improvement regarding the SHAP analysis could involve further investigation into the optimal number of selected features across all datasets to achieve better results with a single AutoEncoder. In addition, investigating the most appropriate latent dimensionality for the AutoEncoder in combination with SHAP-based feature selection could help further enhance detection performance while maintaining interpretability.

Another potential improvement could involve investigating cross-dataset generalization by training the model on two datasets and evaluating it on a third, previously unseen dataset. This setup allows assessing how well the model generalizes to new environments and traffic distributions. In cases of suboptimal performance, it would also help determine whether retraining is necessary and the number of samples required from the new dataset to achieve high detection performance.

Finally, further improvements could be explored by employing more complex model architectures for both the AutoEncoder and the classification model. Using deeper or more advanced models may help capture complex relationships in the data, potentially improving detection performance. Additionally, exploring the framework with alternative datasets that are more complex and heterogeneous than those considered in this study could provide a better understanding of its generalizability and robustness.

Bibliography

[1] Xiao-Yuan Jing, Xinyu Zhang, Xiaoke Zhu, Fei Wu, Xinge You, Yang Gao, Shiguang Shan, and Jing-Yu Yang. «Multiset Feature Learning for Highly Imbalanced Data Classification». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (2021), pp. 139–156 (cit. on p. 7).

[2] Mayu Sakurada and Takehisa Yairi. «Anomaly detection using autoencoders with nonlinear dimensionality reduction». In: *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*. 2014, pp. 4–11 (cit. on p. 7).

[3] Jinwon An and Sungzoon Cho. «Variational autoencoder based anomaly detection using reconstruction probability». In: *Special lecture on IE* 2.1 (2015), pp. 1–18 (cit. on p. 7).

[4] Jie Fu, Lina Wang, Jianpeng Ke, Kang Yang, and Rongwei Yu. «GANAD: A GAN-based method for network anomaly detection». In: *World Wide Web* 26.5 (2023), pp. 2727–2748 (cit. on p. 8).

[5] Niharika Sharma, Bhavna Arora, Shabana Ziyad, Pradeep Kumar Singh, and Yashwant Singh. «A holistic review and performance evaluation of unsupervised learning methods for network anomaly detection». In: *International Journal on Smart Sensing and Intelligent Systems* 1 (2024) (cit. on p. 8).

[6] Ahmed Abdelkhalek and Maggie Mashaly. «Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning.» In: *Journal of Supercomputing* 79.10 (2023) (cit. on p. 8).

[7] Roberto Magán-Carrión, Daniel Urda, Ignacio Diaz-Cano, and Bernabé Dorronsoro. «Evaluating the impact of different Feature as a Counter data aggregation approaches on the performance of NIDSs and their selected features». In: *Logic Journal of the IGPL* 32.2 (2024), pp. 263–280 (cit. on p. 9).

[8] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, and Marius Portmann. «Netflow datasets for machine learning-based network intrusion detection systems». In: *International Conference on Big Data Technologies and Applications*. Springer. 2020, pp. 117–135 (cit. on p. 9).

[9] Minxiao Wang, Ning Yang, Yanhui Guo, and Ning Weng. «Learn-ids: Bridging gaps between datasets and learning-based network intrusion detection». In: *Electronics* 13.6 (2024), p. 1072 (cit. on p. 9).

[10] Syed Wali, Yasir Ali Farrukh, Irfan Khan, and Nathaniel D Bastian. «Meta: Towards a unified, multimodal dataset for network intrusion detection systems». In: *IEEE Data Descriptions* (2024) (cit. on p. 9).