

# Laboratory 1 Report

ANDREA BOTTICELLA\*, s347291, Politecnico di Torino, Italy

ELIA INNOCENTI\*, s345388, Politecnico di Torino, Italy

SIMONE ROMANO\*, s344024, Politecnico di Torino, Italy

This report documents the development of a supervised intrusion-detection pipeline for the CICIDS2017 dataset using feed-forward neural networks. We first align the data with standard machine-learning practice by removing corrupted or duplicated flows, stratifying train/validation/test splits, and applying feature scaling tailored to the observed outlier distribution. Building on this foundation, we benchmark single-hidden-layer models, study the inductive bias introduced by the destination port feature, and progressively extend the architecture depth while varying batch size, optimizer, and loss formulations.

Our experiments show that ReLU-activated shallow models already capture most benign and frequent attack patterns, while class-weighted cross-entropy substantially improves the recall of rare ports and brute-force events. Deeper configurations offer additional gains in macro-averaged performance and training stability, provided that mini-batch sizing and optimization hyperparameters are co-tuned. Finally, regularization strategies based on dropout, batch normalization, and weight decay mitigate overfitting when the network capacity increases, yielding models that remain robust once high-leverage features are neutralised. These results highlight the importance of carefully balancing architecture complexity and regularization to deploy resilient FFNN-based intrusion detectors.

## CONTENTS

Abstract	1
Contents	1
1 INTRODUCTION	1
2 DATA ANALYSIS AND PREPROCESSING	2
3 SHALLOW NEURAL NETWORK (1 LAYER)	4
4 THE IMPACT OF SPECIFIC FEATURES (DESTINATION PORT)	4
5 THE IMPACT OF THE LOSS FUNCTION	4
6 DEEP NEURAL NETWORK	4
7 OVERFITTING AND REGULARIZATION	4
8 CONCLUSIONS	4

## 1 INTRODUCTION

This laboratory focuses on deploying feed-forward neural networks for intrusion detection within the CICIDS2017 benchmark. The assignment emphasises disciplined experiment design: clarify preprocessing assumptions, measure how architectural depth influences classification, and evaluate strategies that mitigate class imbalance. The overall goal is to gain practical insight into how model capacity, hyperparameters, and regularization jointly shape detection performance in cybersecurity settings.

The provided dataset captures flows spanning benign traffic and three attack families—PortScan, DoS Hulk, and Brute Force—collected during the CICIDS2017 campaign. From the original feature space, seventeen attributes were retained to balance relevance and tractability: timing statistics, directional packet metrics, and protocol-level indicators such as the destination port and SYN flag counts. The data exhibits moderate imbalance and includes duplicated flows, motivating both careful cleaning and the later analysis on the inductive bias introduced by port-based attributes.

This is the outline of the report: Section 2 details the preprocessing pipeline and motivates the normalization and split strategy. Section 3 introduces single-layer baselines and examines activation choices, while Section 4 studies the effect of manipulating and removing the destination port feature. Sections 5 through 7 expand the networks with weighted losses, deeper architectures, optimizer comparisons, and regularization controls. The report closes with Section 8, summarising findings and outlining future work.

---

\*The authors collaborated closely in developing this project.

---

Authors' Contact Information: Andrea Botticella, andrea.botticella@studenti.polito.it, s347291, Politecnico di Torino, Turin, Italy; Elia Innocenti, elia.innocenti@studenti.polito.it, s345388, Politecnico di Torino, Turin, Italy; Simone Romano, simone.romano2@studenti.polito.it, s344024, Politecnico di Torino, Turin, Italy.

## 2 DATA ANALYSIS AND PREPROCESSING

### 2.1 Raw Dataset Profile

The initial dataset corresponds to a subset of the CICIDS2017 network traffic traces, selected to study binary intrusion detection using a Feed-Forward Neural Network (FFNN). It contains multiple network flow features capturing statistical, temporal, and content-based characteristics of connections. Before preprocessing, the dataset comprised both numerical and categorical attributes, together with a target variable identifying normal and attack traffic.

A first inspection of the class distribution revealed a strong imbalance between benign and malicious samples, which could potentially bias the learning process. The frequency of each class is shown in Figure 1. The dominance of the benign class justified later balancing strategies and careful partitioning to maintain a representative sample across subsets.

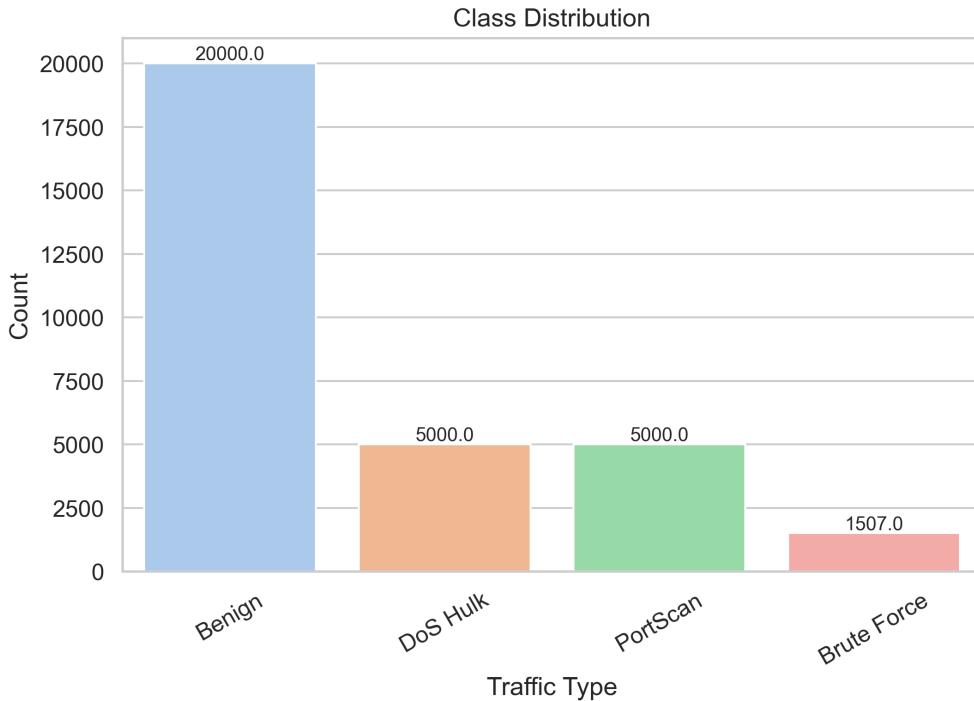


Fig. 1. Distribution of classes in the raw dataset showing the imbalance between benign and attack samples.

To better understand the dataset characteristics, several numerical features were explored to assess scale variability, skewness, and the presence of extreme values. Representative examples of the raw feature distributions are illustrated in Figures 2 and 3. Most features exhibit heavy-tailed distributions and large dynamic ranges, confirming the need for normalization before model training.

### 2.2 Data Cleaning Strategy

The cleaning phase involved identifying and handling inconsistent or missing data, as well as removing irrelevant attributes. Non-numerical fields such as timestamps or identifiers were excluded, since they did not contribute to model learning and could introduce noise. Missing values were inspected; their occurrence was minimal and resolved by removing incomplete rows rather than applying imputation, preserving data consistency. Duplicated entries were also dropped to ensure that each sample represented a unique network flow instance.

During this stage, the data types were unified and all categorical features were encoded numerically to enable direct input to the FFNN. After cleaning, the dataset maintained the same number of relevant features across all samples, ensuring compatibility with the normalization and partitioning steps.

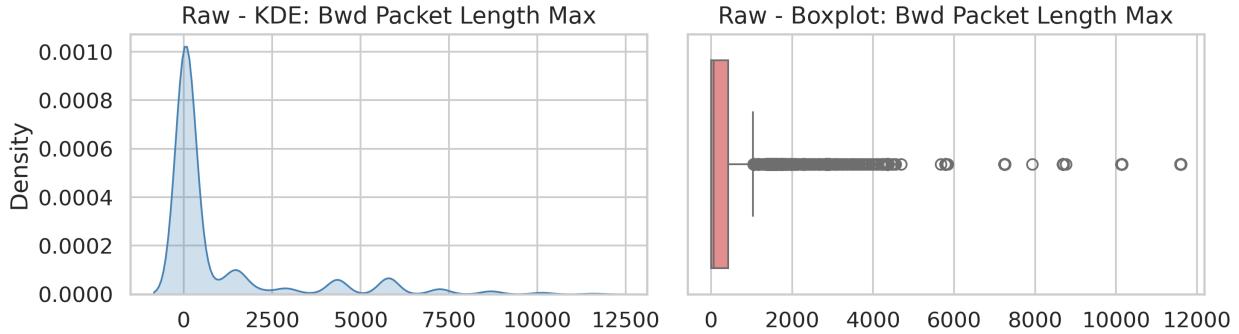


Fig. 2. Example distributions of selected raw numerical features. Many variables show high variance and skewness.

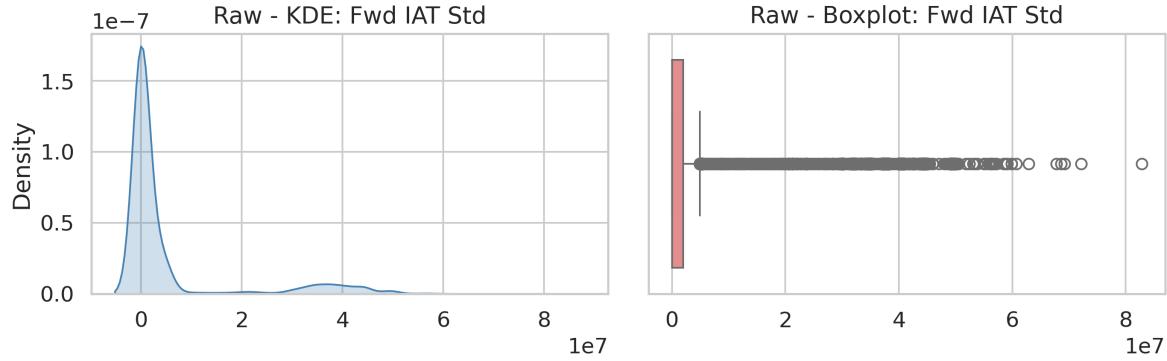


Fig. 3. Additional examples of feature distributions before preprocessing, highlighting differences in scales and the presence of outliers.

### 2.3 Data Partitioning

The cleaned dataset was randomly partitioned into three subsets: 60% for training, 20% for validation, and 20% for testing. A fixed random seed was used to ensure reproducibility. The stratified sampling approach was applied to preserve the class ratio across all splits, reducing the impact of the imbalance on model evaluation. The validation set was used for hyperparameter tuning and early-stopping monitoring, while the test set remained unseen until the final model assessment.

This strategy guarantees a balanced evaluation framework, preventing data leakage and ensuring that the model generalizes to unseen samples.

### 2.4 Outlier Detection and Normalization

Outliers were investigated through boxplots and distribution analysis, identifying samples with exceptionally high or low feature values. While a small number of extreme points were present, they were retained to maintain the realism of the network traffic. Instead of removing them, normalization was employed to mitigate their influence on model training.

Two normalization techniques were compared: Min-Max scaling and Z-score standardization. Both approaches were evaluated to determine which produced more stable distributions and training behavior. Figures 4 and 5 illustrate the effect of normalization on representative features. The transformation successfully compressed the range of values and homogenized the feature scales, reducing the impact of extreme values and improving convergence stability.

Overall, the preprocessing pipeline ensured that the data were consistent, representative, and properly scaled for efficient training of the Feed-Forward Neural Network.

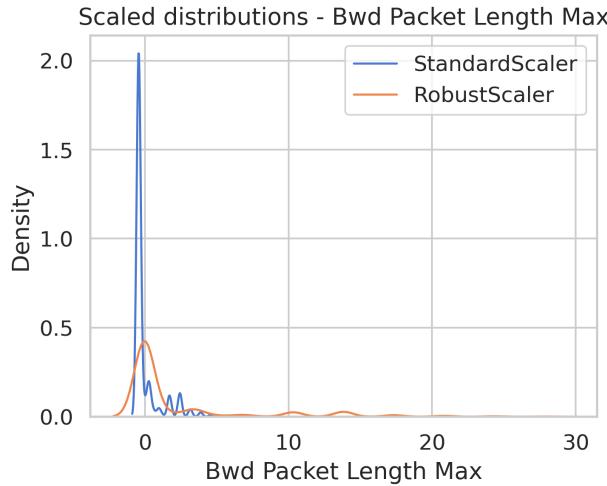


Fig. 4. Comparison between raw and normalized distributions for a subset of features. Normalization reduces scale disparities.

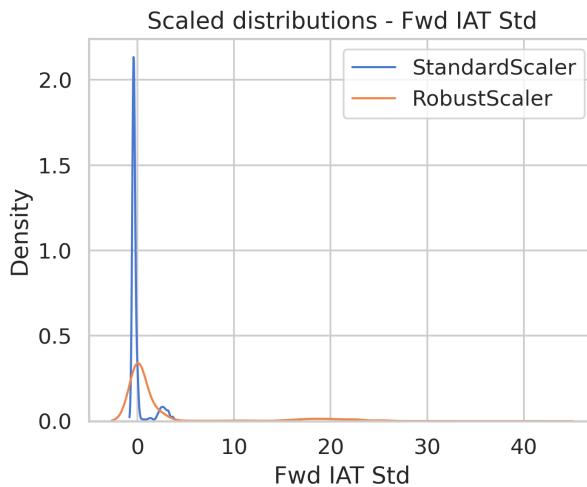


Fig. 5. Further comparison after normalization showing more compact and uniform feature ranges.

### 3 SHALLOW NEURAL NETWORK (1 LAYER)

#### 3.1 Model Configuration

#### 3.2 Training Dynamics

#### 3.3 Validation Performance Analysis

#### 3.4 Activation Function Study

#### 3.5 Generalization Assessment

### 4 THE IMPACT OF SPECIFIC FEATURES (DESTINATION PORT)

#### 4.1 Destination Port Bias Assessment

#### 4.2 Test Set Perturbation Analysis

#### 4.3 Pipeline Update Without Destination Port

#### 4.4 Post-Removal Class Distribution

### 5 THE IMPACT OF THE LOSS FUNCTION

#### 5.1 Baseline Cross-Entropy Evaluation

#### 5.2 Class Weight Estimation Procedure

#### 5.3 Weighted Loss Performance Analysis

### 6 DEEP NEURAL NETWORK