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.

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.

2 DATA ANALYSIS AND PREPROCESSING

2.1 Raw Dataset Profile

The dataset, derived from CICIDS2017, contains labeled network flows with statistical, temporal, and content-based features. It initially included **31,507 samples**, combining benign and attack traffic. The class distribution (Figure 1) shows a strong imbalance, with benign samples dominating.

Exploration of numerical attributes revealed high variability and heavy-tailed distributions (Figures 2), indicating the presence of outliers and the need for normalization.

2.2 Data Cleaning and Partitioning

After removing missing, duplicate, and infinite entries, the dataset was reduced to **29,386 samples**, meaning **2,121 rows** were discarded (2,114 missing/duplicates and 7 infinite values).
The cleaned data were divided into training (60%), validation (20%), and test (20%) subsets using stratified sampling with a fixed seed, ensuring consistent class proportions and reproducibility.

*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.

Publication date: November 2025.

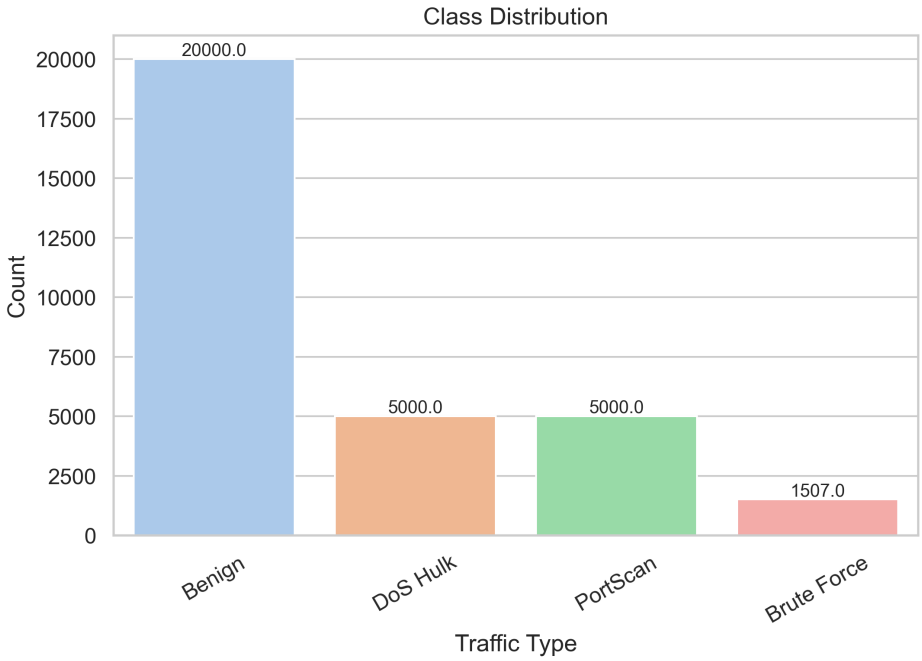


Fig. 1. Class distribution in the raw dataset.

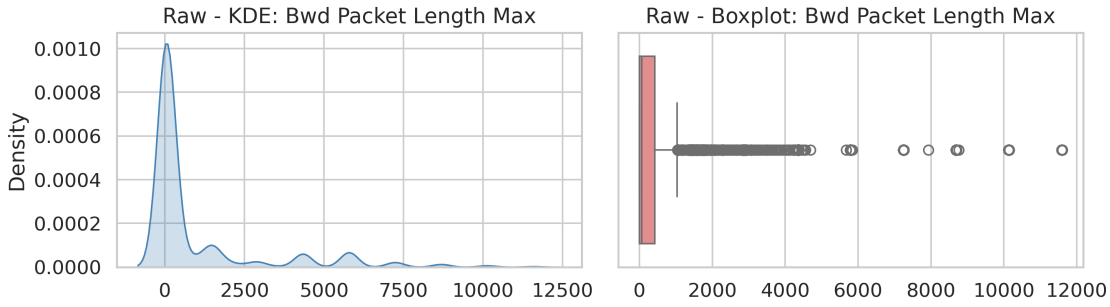


Fig. 2. Examples of feature (Bwd Packet Length Max) kernel density plot and boxplot, highlighting outliers.

2.3 Outlier Detection and Normalization

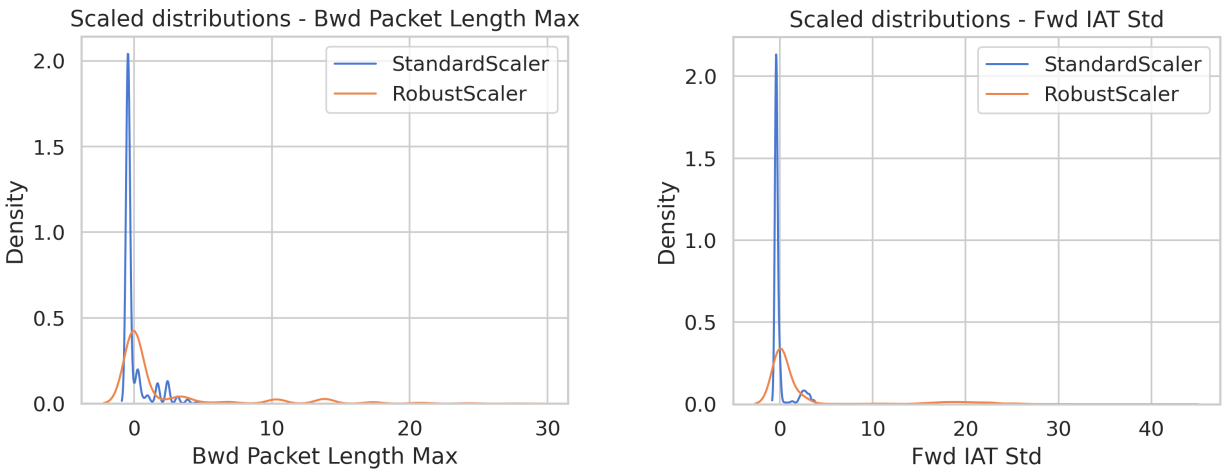
Outliers were analyzed through the *Z-score* and *IQR* methods. Both confirmed extreme values in features such as *Bwd Packet Length Max*, *Flow Duration*, and *Fwd IAT Std*. Outliers were retained to preserve data realism, and normalization was applied to reduce their effect.

Two scaling methods were tested: *StandardScaler* and *RobustScaler*. The latter, based on median and IQR, produced tighter and less skewed distributions, while StandardScaler—though sensitive to outliers—yielded smoother training dynamics. Hence, **StandardScaler** was chosen for the final preprocessing stage.
This preprocessing ensured consistent, balanced, and properly scaled data, forming a robust foundation for training the Feed-Forward Neural Network.

3 SHALLOW NEURAL NETWORK (1 LAYER)

3.1 Model Configuration

A single-layer Feed-Forward Neural Network (FFNN) was trained with 32, 64, and 128 neurons to study the effect of network size on learning and generalization. Each model used the Adam optimizer (lr = 0.001), linear activation



(a) Bwd Packet Length Max.

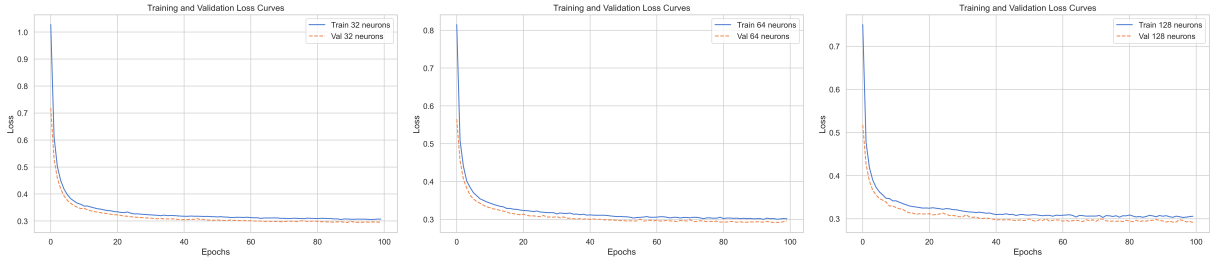
(b) Fwd IAT Std.

Fig. 3. Comparison of normalization effects using StandardScaler and RobustScaler.

function, and early stopping over 100 epochs, minimizing categorical cross-entropy on the same partitions defined in Task 2.

3.2 Training Dynamics

The training and validation loss curves (Figures 4a–4c) show consistent convergence for all models, with rapid loss reduction during early epochs followed by stable plateaus. No overfitting was observed. The 64-neuron network achieved the lowest validation loss (~0.289), while the 128-neuron model reached a similar value before early stopping at epoch 78.



(a) 32 neurons

(b) 64 neurons

(c) 128 neurons

Fig. 4. Training and validation losses for single-layer FFNNs.

3.3 Validation Performance Analysis

Validation accuracy reached about 88% for 32 and 64 neurons and 89.7% for 128 neurons (macro F1 ~0.76). While the 64-neuron model minimized validation loss, the 128-neuron model yielded the most balanced class performance, correctly identifying minority attacks such as *Brute Force*. Increased neuron count improved representational capacity and class separation.

3.4 Activation Function Study and Generalization

Replacing the linear activation function with ReLU (64 neurons) accelerated convergence and enhanced minority-class recognition (Brute Force F1 = 0.85), as shown in Figure 5. Validation and test metrics remained closely aligned, confirming strong generalization. The 64-neuron ReLU network was selected as the reference configuration for subsequent tasks.

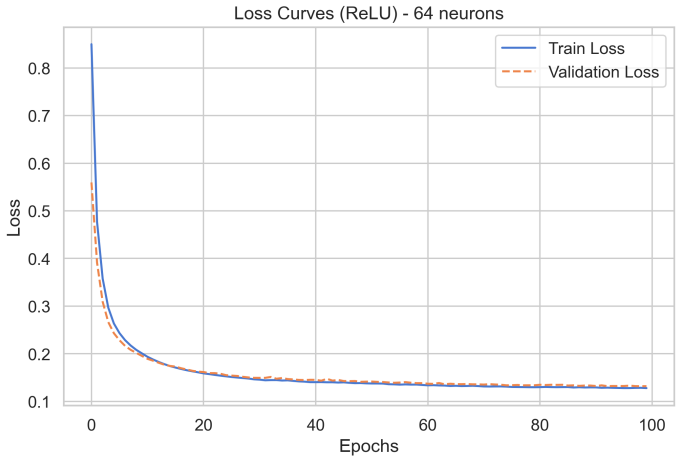


Fig. 5. Loss curves of the 64-neuron model using ReLU activation.

4 THE IMPACT OF SPECIFIC FEATURES (DESTINATION PORT)

4.1 Dataset Bias and Feature Dependence

During the dataset inspection, it was observed that most of the *Brute Force* attacks shared the same *Destination Port* value (port 80). This is an unrealistic scenario, as Brute Force attacks can occur on any service requiring authentication. This systematic bias introduces a wrong inductive pattern, leading the model to associate port 80 exclusively with Brute Force traffic instead of learning meaningful behavioral features. As a result, the model risks overfitting to an artifact of data collection rather than generalizing to real-world cases.

4.2 Evaluating Bias via Port Substitution

To quantify this effect, the trained 64-neuron ReLU model was evaluated on a modified test set in which all Brute Force samples had their destination port changed from 80 to 8080. While the model performed well on the original validation set (Brute Force: F1 = 0.85, overall accuracy ~94%), its performance degraded sharply on the modified test set (Brute Force: F1 = 0.08, overall accuracy ~90.4%). This dramatic decline confirms that the model learned a spurious dependency on the port feature, failing to recognize attacks when this shortcut was removed.

4.3 Effect of Removing the Destination Port Feature

To mitigate this bias, the destination port attribute was excluded, and the dataset was reprocessed. After duplicate and NaN removal, the number of *PortScan* samples decreased drastically from 5,000 to only 285, as shown in Figure 6. This reduction indicates that many PortScan flows were nearly identical except for their port values; removing the feature exposed these redundancies.

4.4 Class Balance Considerations

Even after preprocessing, the dataset remains imbalanced, with benign samples far exceeding attack ones and minority classes (Brute Force, PortScan) underrepresented. Although dropping the destination port feature improves robustness against spurious correlations, addressing class imbalance remains necessary to prevent the model from favoring majority classes.

In summary, this task highlights how biased or overly discriminative features can mislead model learning. Removing such features or introducing data diversity is essential for ensuring fair and generalizable intrusion detection performance.

5 THE IMPACT OF THE LOSS FUNCTION

5.1 Removing the Destination Port Feature

The best-performing model from previous tasks (64 neurons, ReLU activation) was retrained after excluding the *Destination Port* feature to eliminate the bias discussed earlier. This modification had a mixed effect: overall accuracy remained stable, and performance on the *Brute Force* class slightly improved, confirming that the model no longer

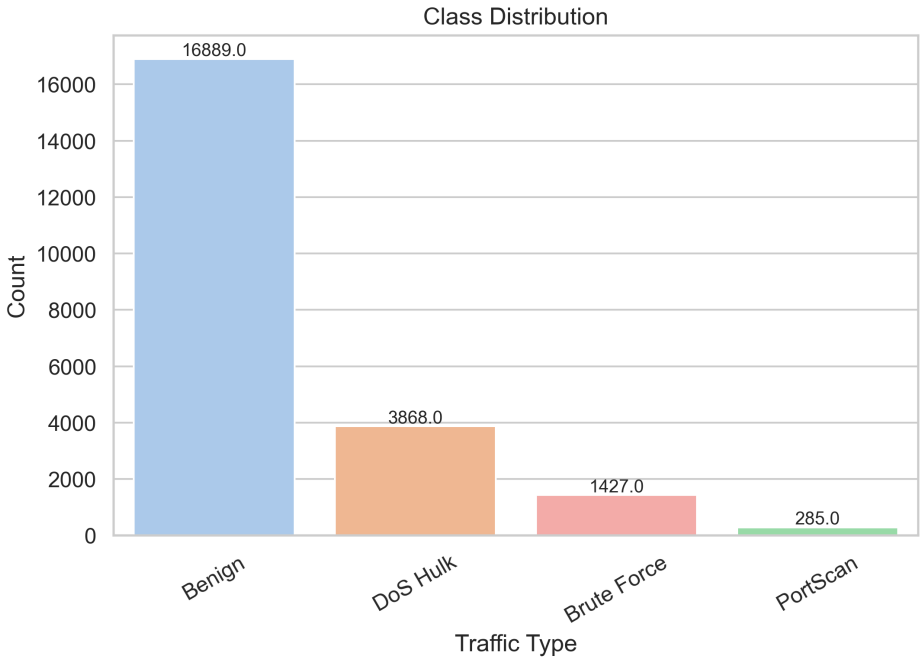


Fig. 6. Updated class distribution after removing the destination port feature.

relied on a biased feature. However, the ability to recognize the rarest class, *PortScan*, declined significantly (F1-score dropped from 0.92 to 0.30), suggesting that the model had previously exploited this feature as a strong shortcut for PortScan detection.

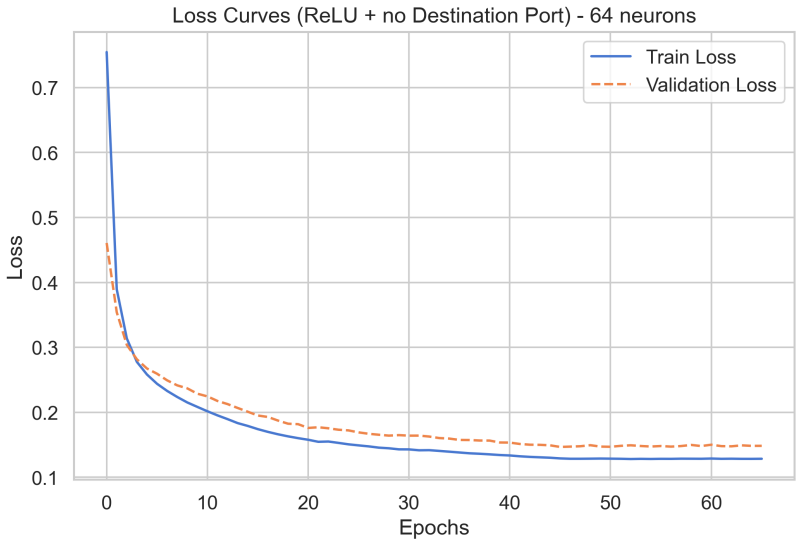


Fig. 7. Training and validation loss curves for the 64-neuron ReLU model without the Destination Port feature.

5.2 Class Weights and Weighted Loss

To counter the imbalance identified in previous tasks, class weights were computed from the *training set* to avoid data leakage. Weighted cross-entropy was then adopted to penalize misclassifications of minority classes more strongly, encouraging the model to learn balanced decision boundaries.

5.3 Effect of Weighted Cross-Entropy

Training with the weighted loss produced smoother convergence and more balanced class performance (Figure 8). Although overall accuracy and weighted F1 slightly decreased, the *macro* F1-score and recall for minority classes improved markedly. This confirms that the weighted cross-entropy promotes fairness across classes, making the model less biased toward dominant traffic types.

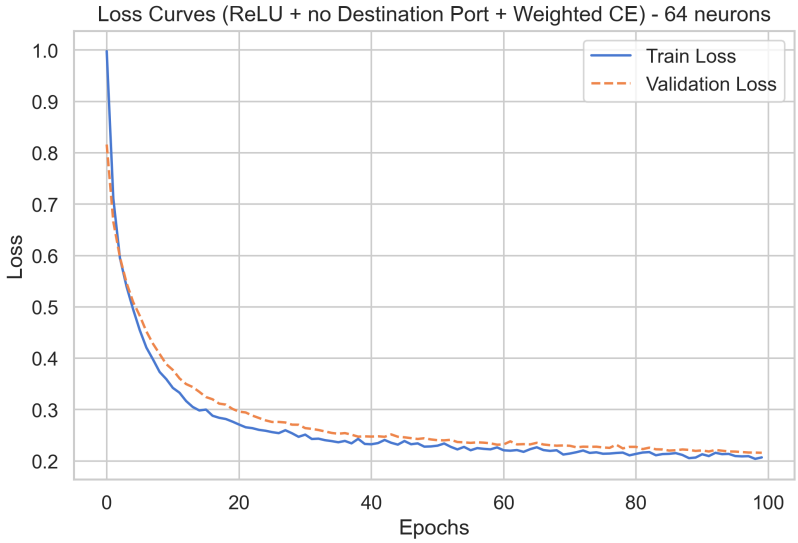


Fig. 8. Training and validation loss curves for the 64-neuron model using weighted cross-entropy.

In summary, applying a class-weighted loss enhanced the detection of underrepresented attacks such as *PortScan* and *Brute Force*, at a minor cost in overall accuracy. This experiment highlights the trade-off between global accuracy and per-class metrics, emphasizing the importance of tailored loss design in imbalanced network intrusion datasets.

6 DEEP NEURAL NETWORK

6.1 Model Architectures and Convergence

Deeper Feed-Forward Neural Networks (FFNNs) were trained to assess the influence of network depth, width, batch size, and optimizer on convergence and generalization. All architectures used ReLU activations, AdamW or SGD-based optimizers, and were trained for 50 epochs. Across all configurations, both training and validation losses showed smooth and consistent convergence without divergence, confirming stable optimization.

6.2 Architecture Selection

Among the evaluated models, the three-layer network with widths [32, 16, 8] achieved the best validation performance (accuracy = 95.55%, macro F1 = 0.80). Deeper configurations achieved slightly higher accuracy but exhibited poorer macro F1, showing weaker handling of minority classes. Hence, the [32, 16, 8] model was selected for further experiments as it offered the best trade-off between complexity and generalization.

6.3 Test Performance

The selected deep model achieved high test performance: accuracy = 0.953, weighted F1 = 0.952, and macro F1 = 0.828. The model generalized well to unseen data, maintaining strong accuracy across all major classes, although *PortScan* remained the most challenging (F1 = 0.50).

6.4 Batch Size Effects

Batch sizes of 4, 64, 256, and 1024 were compared. Smaller batches (4-64) achieved higher validation accuracy (compared to larger batches) (up to 94.9%) and macro F1 (~0.69), while larger batches led to underfitting (accuracy ~89.7%). This occurs because smaller batches introduce stochasticity in gradient updates, enhancing generalization. Considering both stability and efficiency, a batch size of 64 was adopted for subsequent runs.

Publication date: November 2025.

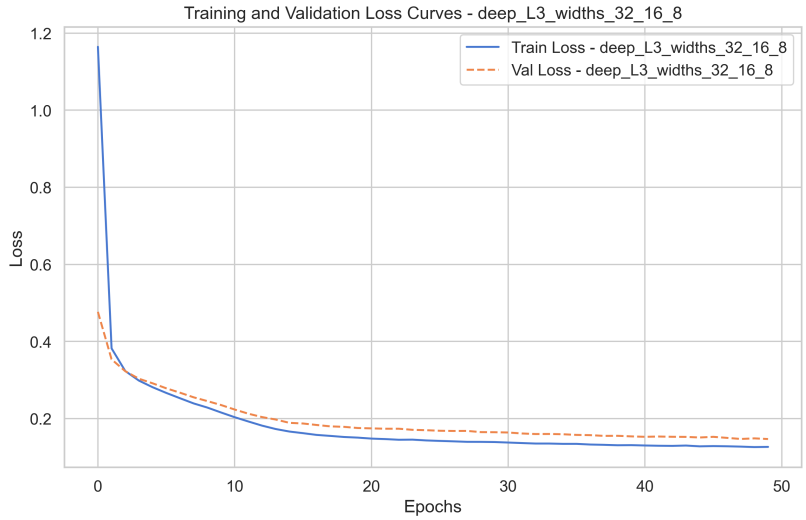


Fig. 9. Training and validation loss of the best deep model (3 layers, widths [32, 16, 8]).

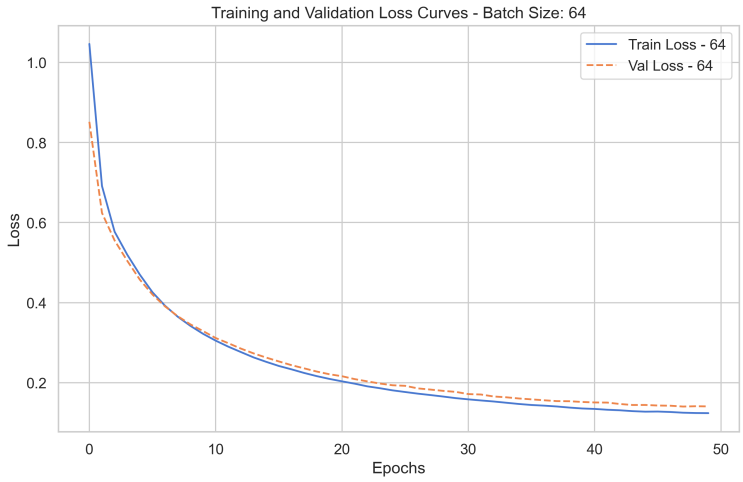


Fig. 10. Loss evolution for the 64 batch-size configuration.

6.5 Optimizer Comparison

Different optimizers were evaluated: SGD, SGD with momentum (0.1, 0.5, 0.9), and AdamW. All converged, but AdamW displayed the fastest and most stable loss decrease, reaching the lowest validation loss. SGD without momentum converged slowly, while momentum improved convergence speed progressively. In terms of training time, plain SGD was fastest due to fewer computations, while AdamW was slower but achieved superior accuracy and F1 performance.

6.6 Learning Rate and Final Evaluation

With AdamW and the [32, 16, 8] architecture, fine-tuning the learning rate confirmed stable convergence and robust test results. The final test classification report showed accuracy = 93.3% and weighted F1 = 0.93, with particularly strong detection of *Benign* and *Brute Force* traffic. These results validate the chosen optimizer and architecture as the most effective configuration for balanced performance and generalization.

Overall, this task highlights the trade-offs among network depth, batch size, and optimizer design, emphasizing that moderate depth and adaptive optimization yield the best balance between performance and efficiency.

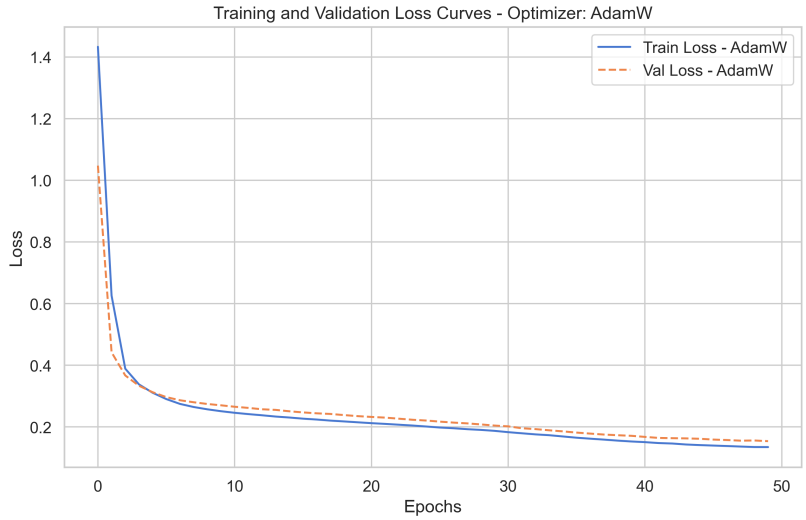


Fig. 11. Loss curve of the AdamW optimizer, showing faster and smoother convergence.

7 OVERFITTING AND REGULARIZATION

7.1 Baseline Model

The baseline deep model (*AdamW*, no explicit regularization) shows consistent convergence with both training and validation losses stabilizing between 0.09-0.11 (Figure 12). The validation loss remains slightly higher than the training loss, as expected, indicating good generalization and no signs of overfitting. Both curves plateau together, confirming stable learning with high validation accuracy (∼96%).

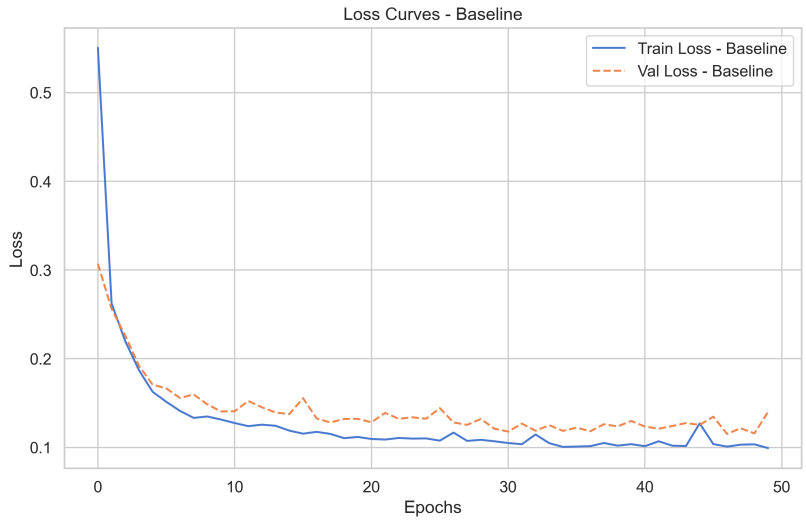


Fig. 12. Training and validation loss curves for the baseline model (AdamW).

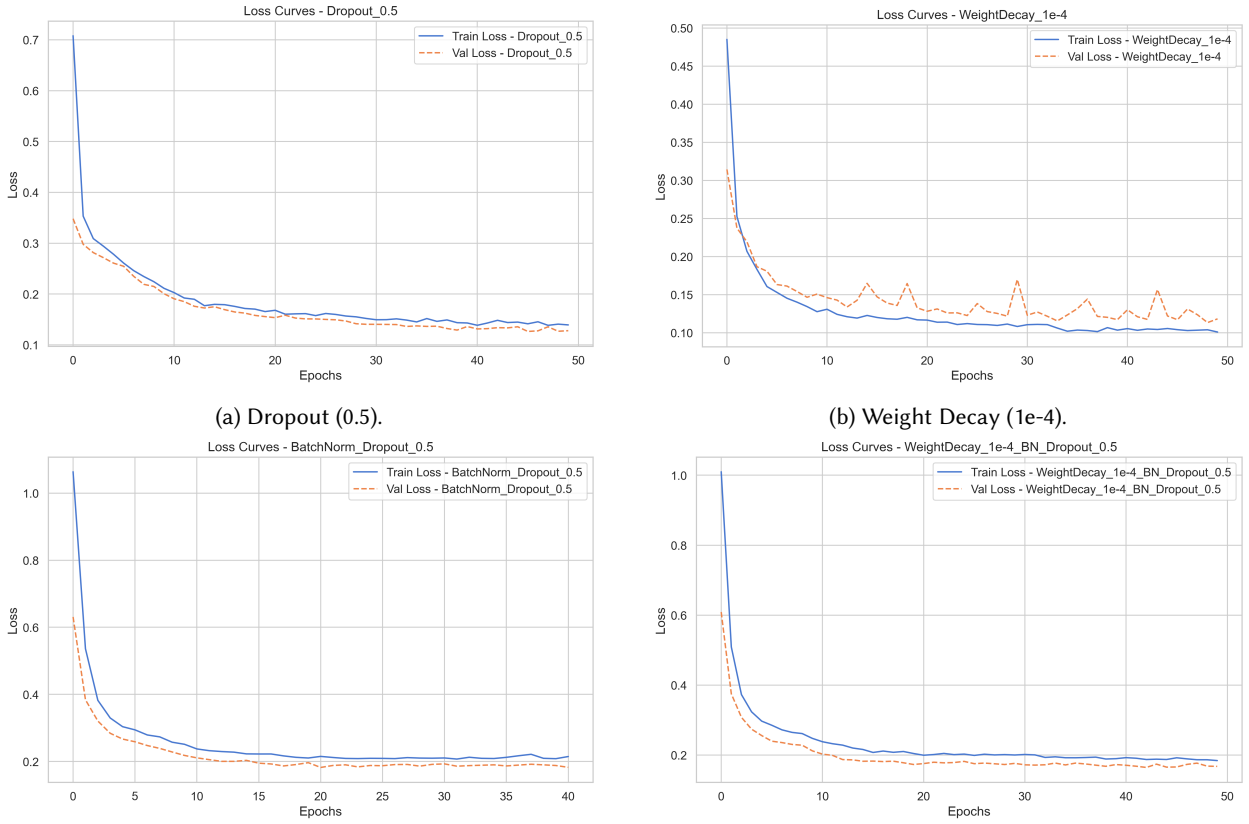
7.2 Effect of Normalization and Regularization

Several regularization strategies were applied to investigate their impact on convergence and performance:

- **Dropout (0.5)**: Increased generalization pressure but led to underfitting. The validation loss dropped below the training loss, and minority classes were ignored during prediction.
- **Batch Normalization**: Produced unstable validation loss and irregular convergence, revealing sensitivity to batch statistics in tabular data.

- **BatchNorm + Dropout (0.5)**: Excessive regularization; validation loss flattened prematurely, causing underfitting and low recall on minority classes.
- **Weight Decay (1e-4)**: Improved stability and slightly enhanced generalization, maintaining accuracy close to the baseline while preventing minor overfitting trends.
- **Weight Decay + BN + Dropout (0.5)**: Over-regularized; convergence slower and validation performance degraded.

Representative examples of the most relevant configurations are reported in Figures 13a-13d.



(a) Dropout (0.5).

(b) Weight Decay (1e-4).

(c) BatchNorm + Dropout (0.5).

(d) Weight Decay + BN + Dropout (0.5).

Fig. 13. Loss evolution across selected regularization strategies.

7.3 Final Observations

Among all tested setups, the **AdamW optimizer with mild weight decay (1×10^{-4})** achieved the best balance between stability and performance, reaching 96.3% validation and test accuracy. Stronger regularization methods such as Dropout or BatchNorm caused underfitting and reduced recall on minority classes, confirming that excessive normalization can harm tabular network training. Hence, light weight decay was identified as the most effective regularization approach for this task.

8 CONCLUSIONS

This laboratory work systematically explored the design and optimization of Feed-Forward Neural Networks (FFNNs) for network intrusion detection using the CICIDS2017 dataset. Starting from data preprocessing, each task progressively refined the model architecture, training strategies, and evaluation procedures to ensure robustness and generalization.

The initial data analysis revealed strong class imbalance and feature biases, particularly in the *Destination Port* attribute, which introduced spurious correlations in model learning. After thorough cleaning, normalization, and bias mitigation, the dataset became suitable for supervised training.

Publication date: November 2025.

The shallow network experiments demonstrated that increasing the number of neurons improved representational capacity, with the 64-neuron model using ReLU activation achieving the best balance between convergence speed and performance. Subsequent analyses confirmed that this model generalized well to unseen data. Removing the biased *Destination Port* feature exposed the model's dependency on non-generalizable patterns, validating the need for careful feature selection.

The introduction of a class-weighted loss improved fairness by enhancing recall for minority classes at the expense of slight accuracy reduction. Deep architectures further improved macro F1-score and accuracy, with the three-layer [32, 16, 8] configuration and AdamW optimizer providing the best trade-off between complexity, efficiency, and generalization. Among the tested optimizers, AdamW exhibited the fastest and most stable convergence, while smaller batch sizes yielded better generalization.

Finally, experiments on overfitting and regularization confirmed that the baseline model trained with AdamW and mild weight decay (1×10^{-4}) achieved the best overall performance. Aggressive techniques such as Dropout or Batch Normalization led to underfitting, indicating that excessive regularization is not optimal for tabular intrusion detection data.

Overall, this study highlights the importance of balanced preprocessing, careful hyperparameter tuning, and moderate regularization in designing neural networks for intrusion detection. The developed FFNN pipeline achieved stable convergence, high accuracy, and good generalization while maintaining interpretability and computational efficiency, providing a solid foundation for future extensions such as convolutional or recurrent models for sequence-based traffic analysis.