Cybersecurity Project Documentation

Daniele Russo, Nicola Modugno

November 2025

1 Introduction

1.1 Context and motivation

The endpoint security industry has increasingly adopted machine learning (ML)-based tools as integral components of their defense strategies. In particular, classifiers that use features derived from static binary analysis are commonly employed for rapid pre-execution detection and prevention, often serving as the first line of defense for end users. However, these systems are vulnerable to adversarial attacks. Historically, the main focus has been on evasion attacks, in which the adversary aims to alter the data point during inference to induce a misclassification.

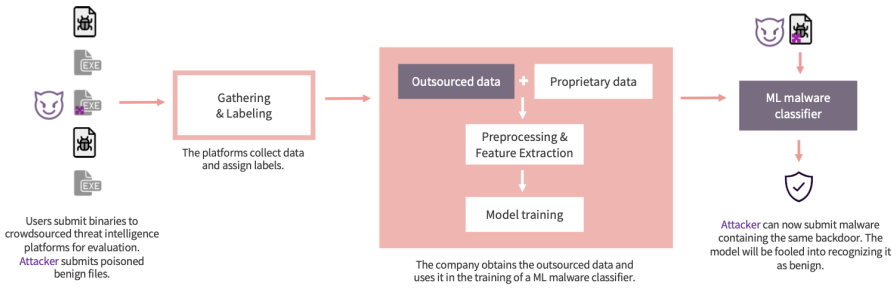


Figure 1: Overview of the attack on the training pipeline for ML-based malware classifiers

However, in this security context, a sneaky problem is poisoning attacks, which try to mess with the ML model training process, especially backdoor poisoning attacks. The training pipeline of many security vendors presents a natural injection point for such attacks, as companies often rely on crowdsourced threat feeds to obtain the large and diverse stream of labeled binaries (both goodware and malware) needed to achieve satisfactory detection performance. In a backdoor attack, the adversary introduces a carefully chosen pattern (trigger or watermark) into the feature space so that the victim model learns to associate its presence with a class chosen by the attacker. Backdoor attacks can be particularly difficult to detect when they fall into the category of "clean-label backdoor attacks," where the adversary injects benign samples with the backdoor without manipulating the original label assigned by third-party analysis systems. This project aims to address the difficulty of defending against such stealthy attacks by exploring the use of sparsity techniques to attempt to detect poisoned samples in the dataset.

1.2 Project objectives

The primary objective of this project is to use sparsity and noise injection techniques to develop a dataset poisoning detector. To achieve this goal, the project involves the following operational phases.

1. Find or build a poisoned malware dataset. This dataset must contain backdoored samples that simulate poisoning attacks against malware classifiers (such as those studied in the context of Windows PE, PDF, or Android applications).

2. Train a neural network as a malware classifier (detector) using the poisoned dataset.

3. Apply two methods to mitigate the effects of poisoning, namely:

- Introduce Gaussian noise into the internal weights of the neural network, and dynamically evaluate how much to disturb the network.
- Sparsify the network (e.g., using pruning techniques that can generate sparse subnetworks, a central concept in the "lottery ticket" or LTH hypothesis). Pruning techniques can reduce the number of parameters while maintaining accuracy, making inference more efficient.

4. Verify a correlation between the classification result obtained after adding noise or pruning techniques and the poisoned nature of the samples, using the results of the paper as a benchmark.

1.3 Working hypothesis: resilience of poisoned samples to misclassification

The central hypothesis that this project aims to verify is that poisoned (or backdoored) samples are resilient to misclassification when noise is introduced into the network, and that the introduction of noise or sparsity can therefore help identify adversarial samples. This hypothesis is based on the idea that the portions of the network that encode the backdoor effect (the trigger) may behave differently from the portions that encode legitimate features. Sparsity techniques involve removing unnecessary weights from neural networks (pruning). The lottery ticket hypothesis (LTH) suggests that dense, randomly initialized neural networks contain sparse subnetworks ("winning tickets") that, when trained in isolation, can achieve accuracy comparable to the original network. It has been observed that weight initialization is crucial to the effectiveness of a sparse subnetwork. In fact, even though adding Gaussian noise to the initialization of a winning ticket tends to reduce its accuracy, winning tickets show surprising robustness to noise (especially at low levels) and continue to outperform randomly reinitialized models. If poisoned samples exploit a specific path or weight configuration (similar, in its uniqueness, to a winning ticket or a "stealth path") that was established during poisoning, their response to weight manipulation (noise or sparsity) may be distinctive. In particular, if the backdoor effect is encoded in weights that remain unusually stable or are particularly important (e.g., weights that move farther during training, as highlighted in a study on LTHs), noise injection may not alter their expected classification (intentional misclassification), thus revealing their adversarial nature. The use of noise injection has also been historically explored in the context of adversarial attacks, where deliberate poisoning of the dataset with well-designed noise aimed to prevent the generation of useful signatures (as in the case of the Polygraph system). This project reverses the logic, using noise not to attack, but to diagnose and detect the presence of existing poisoning attacks.

1.4 Observations: Implications of the "Lottery Ticket Hypothesis" on Backdoor Resilience

The working hypothesis of the project is that poisoned samples are resilient to misclassification errors and that introducing noise into the network (or sparsifying it) can reveal adversarial samples through a distinctive correlation in classification results. However, empirical results from research on scarcity in neural networks (such as the Lottery Ticket Hypothesis - LTH) suggest that learning mechanisms that lead to effective representations may be inherently resistant to perturbations, a factor that could complicate the detection strategy based on noise injection.

1.4.1 Resilience of Noise-Scattered Subnetworks

The Lottery Ticket Hypothesis posits that dense, randomly initialized neural networks contain sparse subnetworks, called "winning tickets," which, when trained in isolation, can achieve accuracy comparable to the original network. The importance of the original initialization is crucial to the success of these winning tickets, as randomly re-initialized sparse subnetworks fail to match the performance of the original network. A specific investigation into the robustness of these winning tickets reveals that adding Gaussian noise to their initializations (a robustness test related to the introduction of noise in the weights) reduces accuracy and slows down learning. Nevertheless, winning tickets show surprising robustness to noise. For example, even after adding noise with a standard deviation of 3σ, winning tickets continue to outperform random reinitialization.

1.4.2 Potential Implication for Poisoning Detection

If we assume that the backdoor effect (the mapping between the trigger and the misclassification as "benign") is encoded in a particularly optimized subnetwork that has "won the initialization lottery" for that specific misclassification task, it follows that this portion of the network (the backdoor) could be robust by nature. An effective backdoor poisoning attack aims to create an area of density in the feature space so that the classifier adapts its decision boundary. More stealthy attack strategies, such as "Greedy Combined Selection," seek to camouflage the backdoor in areas with a high density of legitimate benign samples, making the trigger semantically consistent and difficult to detect using clustering or anomaly techniques. Considering the noise robustness of winning tickets, it is plausible that the subnetwork implementing the trigger→benign association is not significantly perturbed by the addition of noise or network sparsification. Outcome Projection (Critical Assumption): If the robustness of the backdoor is analogous to the robustness of winning tickets to noise, the direct consequence for the project could be that misclassification (misclassification of poisoned samples as benign) will persist even after the introduction of noise or sparsity. This phenomenon would weaken the ability to establish a clear correlation between the variation in classification induced by noise and the presence of the poisoned sample. In summary, the outcome of the project may not be the identification of a break in the classification of poisoned samples, but rather the confirmation of their classification stability (resilience), suggesting that their robustness may derive from an optimal, albeit adversarial, learning mechanism.

2 State of the art

2.1 Data Poisoning and Backdoor Attacks in ML Models

Endpoint security systems increasingly use machine learning (ML)-based tools, particularly classifiers that employ features derived from static binary analysis, for rapid detection and prevention prior to execution. However, these models are exposed to vulnerabilities related to adversarial attacks. Adversarial attacks against ML models fall into two main categories: evasion attacks and poisoning attacks. Evasion attacks aim to perturb a sample during testing to induce misclassification. Poisoning attacks, on the other hand, attempt to manipulate training data by injecting new data or modifying existing data in order to influence the training process and cause misclassifications at inference time.

2.1.1 Clean-label attacks and feature-based backdoors

Poisoning attacks are further subdivided based on their objective, including availability attacks (which degrade overall accuracy) and targeted attacks (which cause misclassifications on specific instances or under certain conditions). Backdoor attacks represent a key subcategory of targeted attacks, where the adversary introduces a carefully chosen pattern or scheme (trigger or watermark) into the feature space. The victim model learns to associate the presence of this pattern with a class chosen by the attacker. When the same watermark is injected into a test sample, it forces the desired (often malicious) classification. This approach, originally introduced for neural networks in image recognition, provides attackers with a generic evasion capability.

Orthogonally, poisoning attacks can also be classified based on the label manipulation strategy:

- Dirty-label attacks (or flip-label attacks): The adversary manipulates both the features and the label of the poisoned sample. In the context of security, this approach is often impractical because labels for crowdsourced data are generated by multiple third-party antivirus engines, which the attacker cannot directly control.

- Clean-label attacks: These advanced and stealthy attacks circumvent the obstacle by not manipulating the original label of the poisoning data. The attacker injects benign samples (modified with subtle perturbations) into the training set, while maintaining the benign label. The goal is to alter the training process so that the model learns malicious associations without raising suspicions.

Clean-label attacks can apply to various objectives and mechanisms:

- Availability (untargeted) clean-label attacks: These aim to degrade the overall performance of the model, making it less accurate on a large dataset, without focusing on specific examples.

- Targeted clean-label attacks: These aim to influence the model's behavior on specific instances (for example, causing it to misclassify a particular image as belonging to a different class) without degrading overall performance.

  - Triggerless targeted attacks: These cause misclassifications on specific instances without requiring a trigger. One example is the "Poison Frogs" attack[2], which uses collisions in feature space for precise targeting.
  - Backdoor (trigger-based) attacks: They insert a hidden "trigger" (e.g., an imperceptible pattern in an image or audio) into the poisoned data, which activates malicious behavior only in the presence of the trigger. They are stealthy because the data remains correctly labeled, but the model learns to depend on the trigger for certain predictions.

Initially, the project opted to use the "flipping label" attack, but following metrics that were too far removed from those reported in the paper, it was decided to use the same attack reported in the literature, namely the "backdoor clean-label" attack.

2.2 Detection and mitigation techniques

Defense against backdoor attacks, particularly in the model-agnostic and clean-label context, is inherently difficult. Many of the current countermeasures focus on deep neural networks for computer vision and assume that attackers manipulate labels. Mitigation strategies evaluated include:

1. Spectral Signatures: This method, adapted to operate on a reduced feature space, relies on Singular Value Decomposition (SVD) to detect two spectrally separable subpopulations ϵ. Samples with the highest outlier scores are filtered out.

2. HDBSCAN (Hierarchical Density-Based Clustering): Assuming that watermarked samples form a high-density subspace, this density-based clustering approach can identify anomalous clusters.

3. Isolation Forest: An unsupervised anomaly detection algorithm that identifies rare and different points. This technique has been observed to be effective in isolating backdoored points, but only when trained on a transformed dataset (reduced to the most important features). When applied to the original feature space, the algorithm detects only a small fraction of the poisoned points, confirming that the subpopulations are not sufficiently separable in the original space.

It has been empirically demonstrated that attacks generated using the Combined strategy (which camouflage themselves in regions with a high density of legitimate samples) are much more stealthy than Independent attacks and are difficult to isolate with these mitigation techniques. The difficulty in defense stems from the combination of the low separability of subpopulations induced by clean-label attacks and the difficulty of distinguishing the dense regions generated by the attack from the dense regions that occur naturally in different benign datasets.
For our project, Isolation Forest was selected as the benchmark method with respect to the paper, as it seems to have given the best results, providing an excellent methodology for comparing the effectiveness of our mitigation methods.

2.3 Sparsity and robustness of neural models

Sparsity techniques focus on eliminating unnecessary weights from neural networks (pruning), reducing the parameter count by over 90% without compromising accuracy, thereby improving inference efficiency. This inherent robustness of sparse subnetworks to noise offers a potential analogy for the resilience of poisoned samples. If the backdoor mechanism exploits an optimized weight path (an adversarial "winning ticket"), this structure could be robust to introduced perturbations (such as noise or sparsification), leading to the persistence of misclassification (misclassification as benign). We can hypothesize that these characteristics may also be present in our solution, as the winning ticket process shortens the training process, since the weights are very similar to the value of the model at the end of training. This means that even without implementing strategies to maximize winning tickets by training the model, it is plausible that there will ultimately be some winning tickets within it, even if they are not obtained in the most efficient and effective way.

3 Methodology

3.1 Construction of the dataset

3.1.1 Poisoned dataset

The **EMBER** (*Endgame Malware BEnchmark for Research*) dataset was used for the experiment, available in the 2017 and 2018 versions. This dataset was chosen because it represents a de facto standard for research in the field of machine learning-based malware detection, providing features extracted statically from Windows PE executables.

The dataset contains over a million samples labeled as benign or malware, and for computational reasons in feature selection or other computationally demanding processes, a representative subset was selected. Starting from the clean dataset, **controlled poisoning** was introduced by means of a *clean label backdoor* attack, consisting of modifying 1% and 3% of random samples of the dataset, changing a subset of features (16, 32, 48, 64, 128) of benign samples by injecting a malicious pattern into them, with the aim of simulating a realistic alteration of the training data as described in Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers[1].

3.1.2 Preprocessing and feature extraction

Preprocessing was performed using the official `ember` library, which allows extraction from JsonL file archives and their vectorization.
N.B. The *ember* library was not written correctly, as when attempting to vectorize the ember2017 dataset, it generated an error due to the lack of some features. For this reason, the `process_raw_features` function in the following folder */ember/ember/feature.py* had to be modified.

Subsequently, an analysis of the features was performed and the 2851 static features available were normalized.

3.2 Malware classifier architecture

3.2.1 Neural networks used

The classifier implemented is a **fully connected feed-forward neural network**, consisting of three hidden layers of 4000, 2000, and 100 neurons, followed by a sigmoid output layer. Each layer is followed by a *Batch Normalization* block and a ReLU activation function. A dropout rate of 0.5 was applied to reduce overfitting, and an L2 penalty with a weight decay factor of $1e^{-5}$ was applied.

The model was trained for 10 epochs with a batch size of 256, using the Adam optimizer with an initial learning rate of 0.001. The choice of this architecture is motivated by the compromise between representation capacity and computational sustainability on local machines, trying to remain as faithful as possible to the one used in the paper.
To manage class imbalance, a positive weight (pos_weight) equal to the ratio of benign to malware samples was calculated and integrated into the loss function, even though the dataset is almost balanced.

3.2.2 Evaluation Metrics

To comprehensively evaluate the model's performance under the five experimental conditions (*clean*, *poisoned*, *isolation forest*, *noisy*, *pruned*), we employed a multi-faceted evaluation approach based on the following metrics:

Confusion Matrix Components All evaluation metrics are derived from the confusion matrix, which provides a complete breakdown of the model's predictions: True Positives (TP),True Negatives (TN),False Positives (FP) and False Negatives (FN).

Standard Classification Metrics From the confusion matrix, we derive the following performance metrics: F1-Score, Accuracy, Precision, Recall and Specificity.

Threshold-Independent Evaluation. The **ROC-AUC (Area Under ROC Curve)** valuates the model's discriminative ability across all possible classification thresholds, providing a threshold-independent performance measure that is particularly robust to class imbalance.

3.2.3 Comparison with Baseline Defenses

The results are systematically compared with the baselines reported by Severi et al. [1] using the `print_defense_comparison()` function, which computes the variation rate:

$$\text{Variation} = \frac{Acc_{\text{defense}} - Acc_{\text{poisoned}}}{Acc_{\text{poisoned}}} \times 100\% \tag{1}$$

The variation rate quantifies the relative change in performance caused by the applied defense compared to the poisoned model. Positive values indicate an improvement over the poisoned performance, while negative values denote a degradation.

3.3 Introduction of noise and sparsification

3.3.1 Adaptive disturbance of internal weights

To assess the internal robustness of the network and implement a defense strategy against backdoor attacks, an adaptive Gaussian perturbation system for model weights was developed. Unlike approaches with fixed noise, the implementation uses automatic tuning that identifies the optimal noise level through the `tune_gaussian_noise()` function.

Automatic tuning algorithm The optimization process systematically tests different levels of standard deviation $\sigma \in \{0.0001, 0.0005, 0.001, 0.002, 0.003, 0.005, 0.01\}$, evaluating for each:

1. **Functionality preservation**: Verify that accuracy remains above a minimum acceptable threshold (default: 50%)
2. **Degradation maximization**: Among the "viable" levels, select the one that maximizes the drop in accuracy compared to the backdoored model

$$\text{score}(\sigma) = \begin{cases} \text{Acc}_{\text{backdoor}} - \text{Acc}_{\text{noisy}}(\sigma) & \text{se Acc}_{\text{noisy}}(\sigma) \geq \text{threshold} \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

The system automatically handles critical situations with a fallback mechanism: if no level meets the minimum threshold, the least destructive noise is selected with an explicit warning that the defense may not be effective.

Noise is applied using `add_gaussian_noise_to_model()` with the following properties:

- **Selectivity**: Noise is applied only to `weight` layers, excluding biases to preserve optimization stability
- **Traceability**: Detailed statistics are recorded for each modified layer (original mean, mean shift, standard deviation variation)
- **Serializability**: All results are converted to JSON-compatible formats for further analysis

Post-disturbance statistical analysis shows that optimal noise maintains controlled variations:

$$|\Delta\mu_{\text{layer}}| \approx 10^{-4} \text{ to } 10^{-6}, \quad |\Delta\sigma_{\text{layer}}| \approx 10^{-4} \text{ to } 10^{-5} \tag{3}$$

confirming that the disturbance acts as calibrated micro-noise rather than destructive noise.

3.3.2 Applying structural pruning techniques

In parallel with noise-based defense, a progressive pruning strategy inspired by Frankle and Carbin's Lottery Ticket Hypothesis was implemented, but specifically adapted for the detection and mitigation of backdoor attacks.

Detection via Weight Pruning The `WeightPruningDetector` class implements an innovative approach based on the assumption that poisoned samples depend on **specific patterns in weights**, while clean samples are based on **distributed and robust representations**.
The detection process takes place in three stages:

1. **Progressive pruning**: k models are created with increasing pruning percentages $\rho \in \{0\%, 5\%, 10\%, \ldots, 90\%\}$, where for each level, weights with a magnitude less than the ρ-th quantile are reset to zero:

   $$W_{pruned} = W \cdot 1_{\{|W| > Q_{1-\rho}(|W|)\}} \tag{4}$$

   Optimized implementation: For large models (> 10M parameters), quantile calculation uses **random sampling** on a subset of 10M elements, drastically reducing memory usage without significantly impacting threshold accuracy.

2. **Calculation of stability scores**: For each sample x_i, the fraction of pruning levels in which the prediction remains identical to that of the original model is measured:

   $$\text{stability}(x_i) = \frac{1}{k} \sum_{j=1}^{k} 1_{f_{\rho_j}(x_i) = f_{\text{orig}}(x_i)} \tag{5}$$

3. **Adaptive detection**: Unlike fixed thresholds, the system uses the **10th percentile** of stability scores as a threshold, assuming that poisoned samples show low stability:

   $$\text{suspected\_poisoned} = \{i : \text{stability}(x_i) < q_{0.1}(\text{stability})\} \tag{6}$$

Defense through Optimal Pruning The function `experiment_pruning_defense()` implements a "one-shot" pruning strategy for defense:

1. **Identification of the optimal point**: Analyze the prediction matrix $P \in \{0, 1\}^{n \times k}$ to find the maximum pruning rate that maintains 98% of the original predictions:

   $$\rho^* = \max\{\rho_j : \text{mean}(1_{P[:,j] = P\_\text{orig}}) \geq 0.98,\ j \in \{1, \ldots, k\}\} \tag{7}$$

2. **Safety guardrail**: If $\rho^* < 0.1$, it is forced to 0.3 to avoid ineffective under-pruning.

3. **Application and validation**: The pruned model is created, saved, and evaluated, with comprehensive metrics compared to the baseline.

Displaying results :

The function `plot_pruning_detection_results()` generates a **6-panel diagnostic graph** (Figure 2) showing:

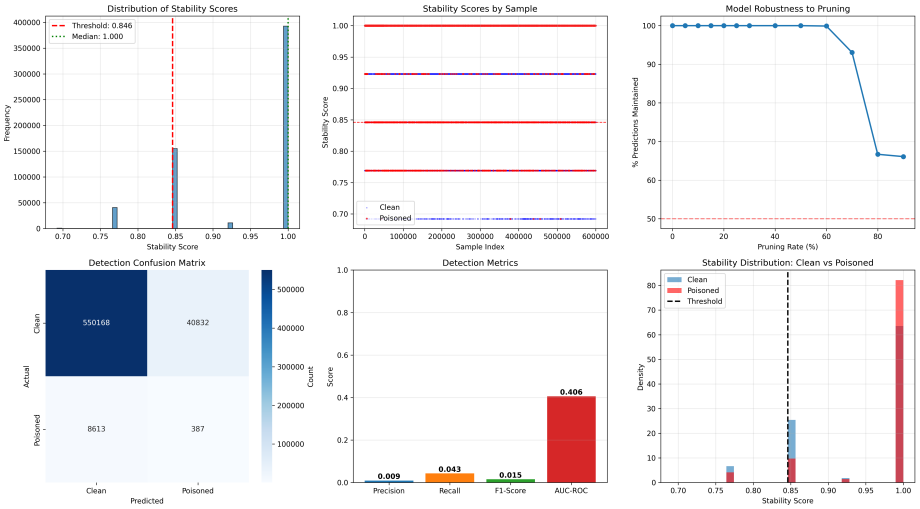


Figure 2: Graph generated to evaluate the optimal pruning level

- **Row 1**:
  - Distribution of stability scores (5) with highlighted threshold
  - Scatter plot of stability vs. sample index with ground truth (red=poisoned)
  - Robustness curve: % predictions retained vs. pruning rate
- **Row 2**:
  - Detection confusion matrix (if ground truth available)
  - Detection metrics (Precision, Recall, F1, AUC-ROC)
  - Comparison of distributions: Clean vs. Poisoned stability

3.4 Dataset Analysis and Result Visualization

3.4.1 Single-Run Evaluation Metrics

For each experimental configuration, the system generates a comprehensive visualization (`backdoor_comparison_enhanced.png`) that consolidates all performance metrics and provides a complete picture of both attack effectiveness and defense performance. This visualization employs a 2-row, 5-column layout designed to facilitate rapid assessment of model behavior across the attack-defense pipeline.

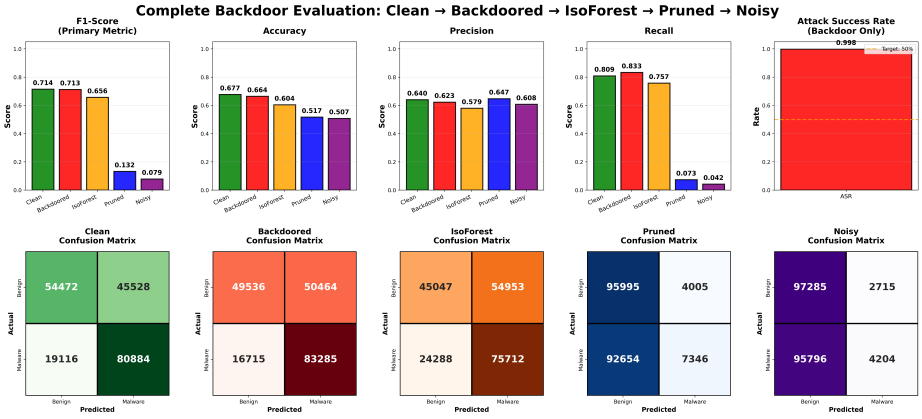


Figure 3: Complete backdoor evaluation pipeline showing performance metrics (top) and confusion matrices (bottom) across clean baseline, backdoored model, and three defense mechanisms (Isolation Forest, Weight Pruning, Gaussian Noise). ASR of 99.8% demonstrates successful backdoor implantation, while defenses exhibit varying recovery effectiveness.

Top Row: Performance Metrics. The first row presents five critical scalar metrics in bar chart format, enabling direct quantitative comparison across all evaluated models (Clean, Backdoored, Isolation Forest Defense, Weight Pruning Defense, Gaussian Noise Defense), paying particular attention to:

- **Attack Success Rate (Column 4)**: Exclusive to the backdoored model, this metric measures the fraction of triggered malware samples (malware + trigger pattern) misclassified as benign. ASR directly quantifies attack danger: high ASR (> 0.5) indicates the backdoor successfully enables evasion, while low ASR suggests attack failure. A target reference line at 50% is overlaid to contextualize effectiveness. This metric is undefined for clean and defense models, as it specifically evaluates backdoor trigger effectiveness.

Bottom Row: Confusion Matrices. The second row presents confusion matrices for all five models, providing granular insight into classification behavior beyond scalar metrics.

The side-by-side arrangement facilitates direct visual comparison of classification behavior. For instance, a successful backdoor attack manifests as increased FN in the backdoored matrix relative to clean, while an effective defense shows FN reduction toward clean baseline. The confusion matrices also reveal whether defenses introduce new failure modes (e.g., increased FP from overly conservative pruning).

3.4.2 Other Tools

Two comprehensive analysis tools were developed to support systematic evaluation and interpretation of experimental results.

Dataset Quality Analysis (`dataset_analysis.py`)
The `EMBERDatasetAnalyzer` class performs exhaustive dataset characterization prior to experimentation, generating quantitative metrics and visualizations to guide preprocessing decisions. The analysis pipeline encompasses five components:

1. **Basic Statistics**: Dataset dimensions, memory footprint, class distribution, and imbalance ratios for both training and test sets. These metrics establish the fundamental characteristics of the evaluation scenario.

2. **Feature Quality Assessment**: Systematic evaluation of feature reliability through multiple dimensions: constant features (zero variance), near-constant features (variance $< 10^{-6}$), sparsity analysis ($> 95\%$ and $> 99\%$ zero values), missing value detection, outlier prevalence (IQR-based), scale heterogeneity (range > 1000), and distribution skewness ($|\text{skew}| > 2$). These metrics identify features requiring special handling or removal.

3. **Correlation Analysis**: Computation of pairwise Pearson correlations on a stratified sample (50,000-250,000 examples) to manage computational cost. The analysis generates: (a) correlation matrix heatmap for the first 100 features, revealing structural patterns; (b) distribution histogram of all pairwise correlations with threshold overlay ($r = 0.95$); (c) summary statistics including highly correlated pair counts and average correlation magnitude. This identifies feature redundancy that can be exploited for dimensionality reduction.

4. **Feature Importance Ranking**: Mutual Information scores quantify the discriminative power of each feature relative to the target label. The analysis produces: (a) bar plot of top-50 most important features; (b) distribution histogram of all MI scores; (c) cumulative importance curve identifying feature subsets that capture 80% and 95% of total information; (d) ranked MI scores on logarithmic scale. These visualizations guide feature selection by revealing diminishing returns and informative feature concentration.

5. **Feature Selection Impact Simulation**: Evaluation of correlation-based filtering at multiple thresholds ($r \in \{0.90, 0.95, 0.98, 0.99\}$), simulating the trade-off between dimensionality reduction and information retention. For each threshold, features from highly correlated pairs are removed (retaining higher-variance feature), and reduction statistics are computed. Visualizations include: (a) line plot showing remaining vs. removed features by threshold; (b) bar plot of reduction percentages. This analysis informs the selection of appropriate correlation thresholds.

All analysis outputs are saved to a dedicated directory (`Results/ember2018/dataset_analysis/`), including: correlation heatmap, feature importance plots, selection impact visualizations, comprehensive JSON report, and Markdown summary. These artifacts document dataset characteristics and support reproducible preprocessing decisions.

Experimental Results Analysis (`result_analysis.py`)
The `FinalAnalyzer` class aggregates and visualizes results across multiple experimental configurations (poison rates $\times$ trigger sizes), enabling systematic comparison of attack effectiveness and defense performance. The analysis operates in four stages:

1. **Result Aggregation**: Recursive traversal of experiment output directories (`Results/ember2018 - mac/` and `Results/ember2018 - cluster/`), loading JSON result files for all completed configurations. Each configuration is indexed by (p, k) tuple (poison rate, trigger size) and contains: clean baseline metrics, backdoored model metrics with ASR, defense results (Isolation Forest, pruning, noise), and attack metadata.

2. **Dataframe Construction**: Results are consolidated into a structured pandas DataFrame with computed derived metrics: accuracy drop (clean $\rightarrow$ backdoored), attack success rate, and

recovery percentages for each defense mechanism. The DataFrame is sorted by poison rate and trigger size for systematic presentation.

3. **Summary Table Export**: The complete DataFrame is exported to CSV (`comprehensive_results.csv`) with rounded numeric values for readability. This table serves as the primary reference for quantitative result comparison and supports subsequent statistical analysis.

4. **Visualization Generation**: Four publication-quality plots are generated:

   - *Stealthiness Analysis - Accuracy*: Line plot showing accuracy change (backdoor vs. clean) by trigger size, with separate curves for each poison rate (1% blue, 3% red). Positive values indicate the backdoor model performs better than clean (ultra-stealthy), while negative values indicate detectable performance degradation.
   - *Stealthiness Analysis - F1-Score*: Complementary analysis using F1-score change instead of accuracy.
   - *Defense Effectiveness - Accuracy Variation*: Grouped bar chart comparing three defense mechanisms (Isolation Forest blue, Weight Pruning orange, Gaussian Noise green) across all configurations. Bars represent percentage variation: Positive values indicate defense improvement over backdoored baseline, negative values indicate further degradation.
   - *Defense Effectiveness - F1-Score Variation*: Parallel analysis for F1-score variation across defenses.
   - *Defense Accuracy - Detailed Comparison*: Three-panel visualization showing final accuracy achieved by each defense mechanism (bars) overlaid with clean model (green line) and backdoor model (red line) baselines across all configurations.
   - *Defense F1-Score - Detailed Comparison*: Parallel three-panel F1-score analysis.

All visualizations are saved to `Results/analysis_plots/` at 300-400 DPI resolution for publication quality. The complete analysis pipeline is executed via `analyzer.run_all()`, automating result processing and report generation.

These analysis tools provide comprehensive documentation of dataset characteristics and experimental outcomes, supporting reproducible research and facilitating identification of optimal configurations for both attacks and defenses.

4 Dataset Information

4.1 EMBER Dataset Overview

The **Endgame Malware BEnchmark for Research (EMBER)** dataset [2] represents a comprehensive benchmark for training static PE malware machine learning models. The dataset is publicly available through the GitHub repository (`https://github.com/elastic/ember`) and can be downloaded from `https://ember.elastic.co/ember_dataset_2018_2.tar.bz2`, with a compressed size of 2.8GB expanding to 6.5 GB when extracted.

Two major versions of the dataset exist: EMBER 2017 and EMBER 2018, each containing 1.1 million examples. For this work, we utilize the EMBER 2018 version, which provides a balanced and representative collection of benign and malicious Windows Portable Executable (PE) files.

4.1.1 Dataset Structure

The EMBER dataset is organized into distinct training and test partitions:

- **Training set**: 900,000 examples, subdivided into:
  - 600,000 labeled samples for training
  - 200,000 labeled samples for validation
  - 100,000 unlabeled samples
- **Test set**: 200,000 examples (100,000 benign, 100,000 malware)

The dataset exhibits perfect class balance with an imbalance ratio of 1.0 in both training and test sets (300,000 benign vs. 300,000 malware in training; 100,000 vs. 100,000 in test). Samples are labeled as 0 (benign), 1 (malware), or -1 (unlabeled). The original data format consists of compressed JSONL files, which have been vectorized into NumPy binary format (`.dat` files) for efficient processing: `X_train.dat`, `y_train.dat`, `X_test.dat`, and `y_test.dat`. The complete vectorized dataset requires approximately 9 GB of memory.

4.1.2 Feature Categories

Each sample in the EMBER dataset is characterized by 2,381 static features extracted from various components of PE files, organized into eight distinct categories:

1. **General File Information** (10 features): File size, virtual size, presence of debug information
2. **Header Information** (62 features): Machine type, characteristics, timestamp, subsystem information
3. **Imports** (1,280 features): Histogram of imported libraries and functions, representing the external dependencies of the executable
4. **Exports** (128 features): Count and type of exported symbols, relevant for DLL files
5. **Section Information** (255 features): Properties of PE sections including size, entropy, and permissions
6. **Data Directories** (30 features): Presence and size of standard PE data directories
7. **Byte Histogram** (256 features): Distribution of byte values throughout the file
8. **String Information** (104 features): Count and characteristics of printable strings found in the file

4.2 Feature Quality Analysis

A comprehensive analysis of the dataset's feature quality reveals several important characteristics that inform subsequent preprocessing decisions.

The analysis reveals that 40 features are completely constant across all samples, providing no discriminative information. Additionally, 386 features are extremely sparse with more than 99% zero values, while 1,067 features exceed 90% sparsity. The average sparsity across all features is

72.15%, indicating that the dataset is predominantly sparse in nature. Notably, no features contain NaN values, eliminating the need for missing value imputation.

The feature value ranges span from -4.28×10^9 to 4.29×10^9, with an average variance of 1.22×10^{14}, suggesting significant scale differences across features that may benefit from normalization. Figure 4 displays the correlation structure and distribution for the first 100 features.

4.3 Correlation Analysis

To identify redundant features, we computed pairwise Pearson correlation coefficients among all 2,341 non-constant features. Due to computational constraints, this analysis was performed on a stratified random sample of 10,000 examples. The correlation analysis reveals substantial feature redundancy.

While the average correlation is relatively low (0.0307), there exist 4,712 feature pairs with very high correlations exceeding 0.95, indicating substantial redundancy. This is particularly common among import features, where similar libraries often appear together. The distribution of pairwise correlations, shown in Figure 4 (right panel), is heavily concentrated near zero, with pronounced peaks at both -1 and $+1$ corresponding to perfectly anti-correlated and perfectly correlated feature pairs.

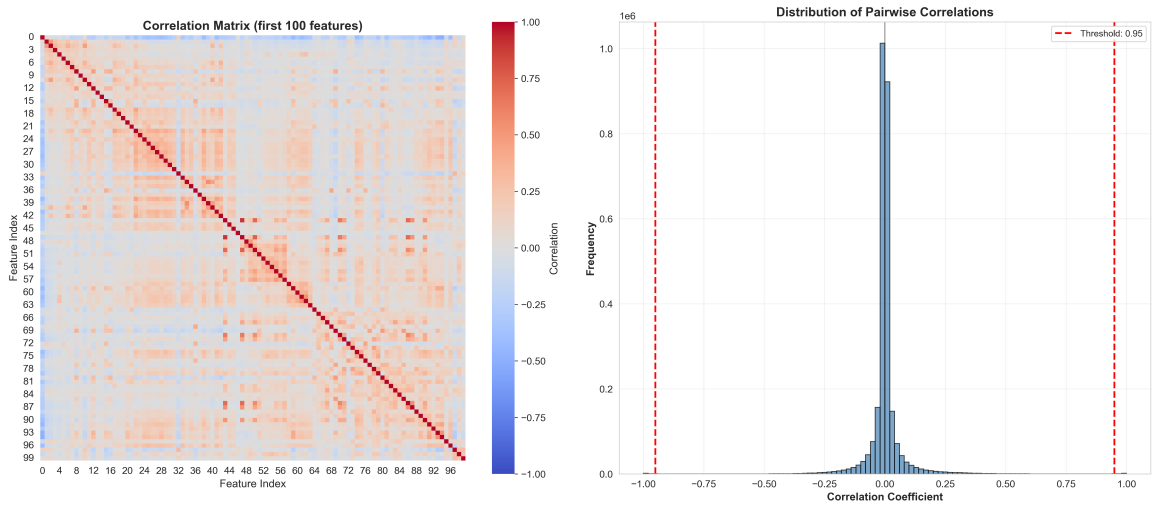


Figure 4: Correlation matrix (left) showing patterns among the first 100 features, and distribution of pairwise correlations (right) with threshold at 0.95. The majority of feature pairs exhibit low correlation, but 4,712 pairs exceed the 0.95 threshold, indicating substantial redundancy.

4.4 Feature Selection Impact

To assess the potential for dimensionality reduction through correlation-based feature selection, we evaluated the impact of various correlation thresholds. For each threshold, when two features exhibited correlation exceeding the threshold, we retained the feature with higher variance. Table 1 and Figure 5 present the results.

At the commonly used threshold of 0.95, correlation-based filtering removes 172 features (7.4% reduction), while a more aggressive threshold of 0.90 eliminates 211 features (9.0% reduction). These

Table 1: Impact of correlation-based feature selection at different thresholds

Threshold	Features Removed	Features Remaining	Reduction (%)
0.90	211	2,129	9.0%
0.95	172	2,168	7.4%
0.98	142	2,198	6.1%
0.99	132	2,208	5.6%

modest reduction percentages suggest that while feature redundancy exists, the majority of features provide relatively independent information. The diminishing returns observed at stricter thresholds (0.98 and 0.99) indicate that most highly correlated features have already been captured at the 0.95 threshold.

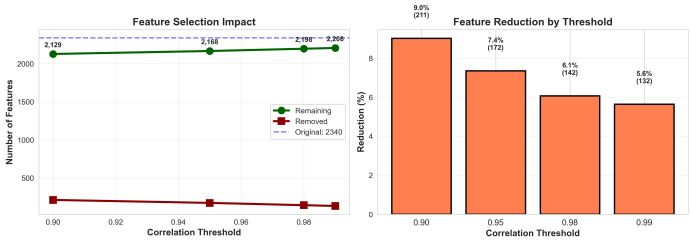


Figure 5: Impact of correlation-based feature selection at various thresholds. Left: Number of features remaining (green) versus removed (red) at each threshold. Right: Percentage reduction achieved. At the selected threshold of 0.98, 142 features (6.1%) are removed while retaining 2,198 features.

4.5 Feature Importance Analysis

To quantify the discriminative power of individual features, we computed Mutual Information (MI) scores between each feature and the target label using a subset of the training data. Mutual information measures the reduction in uncertainty about the class label given knowledge of the feature value, providing a non-linear measure of feature relevance.

Table 2: Mutual information score statistics

Metric	Value
Mean MI score	0.0498
Standard deviation	0.0752
Maximum MI score	0.3227
Minimum MI score	0.0000
Features with zero MI	93

The MI score distribution, summarized in Table 2, reveals considerable heterogeneity in feature importance. The mean MI score of 0.0498 with standard deviation 0.0752 indicates that most features provide modest discriminative information, while a small subset of features exhibits substantially higher importance (maximum MI of 0.3227). Notably, 93 features have zero mutual information with the target, suggesting they provide no useful information for classification.

Figure 6 (top-left panel) displays the top-50 most important features ranked by MI score. The most discriminative features include feature indices 2354, 626, 2355, 786, and 836, with MI scores ranging from approximately 0.28 to 0.32. These features primarily originate from the imports category (indices 500-1780), suggesting that imported function patterns are highly indicative of malicious behavior.

The cumulative importance curve (Figure 6, bottom-left panel) demonstrates that approximately 516 features (22% of total) capture 80% of the cumulative mutual information, while 807 features (34.5%) account for 95% of the total information. This suggests that substantial dimensionality reduction is possible with minimal information loss. The MI score distribution (top-right panel) exhibits strong positive skew, with most features concentrated near zero and a long tail of more informative features. The ranked MI scores on a logarithmic scale (bottom-right panel) show a gradual decay across the feature space, with a sharp drop-off in the least informative 200-300 features.

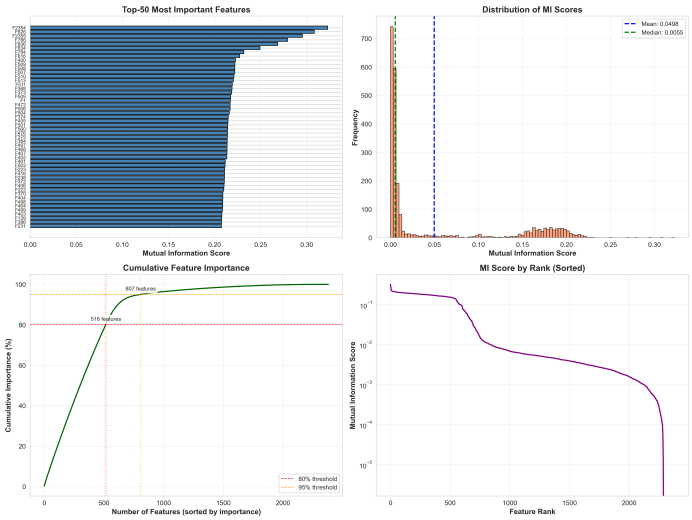


Figure 6: Feature importance analysis based on Mutual Information scores. Top-left: 50 most discriminative features ranked by MI score. Top-right: Distribution of MI scores showing strong positive skew. Bottom-left: Cumulative importance curve indicating that 516 features capture 80% of total information. Bottom-right: MI scores by rank on logarithmic scale, revealing gradual decay with sharp drop-off for least informative features.

4.6 Feature Selection Pipeline

Based on the quality and correlation analyses, we implement a systematic feature selection pipeline:

1. **Constant feature removal**: Eliminate all 40 features with zero variance, as they provide no discriminative information
2. **Correlation-based filtering**:
   - Compute Pearson correlation matrix on a stratified sample of 10,000 examples
   - For feature pairs with $|r| > 0.98$, retain the feature with higher variance
   - This threshold removes 142 highly redundant features while preserving diversity

3. **Mutual information selection**:
    - Calculate MI scores between each remaining feature and the target label
    - Select features based on MI score thresholds or top-k ranking (if anabled)
    - Remove 93 features with zero mutual information
4. **Computational optimization**:
    - Employ parallel processing with `joblib` (`n_jobs=-1`) for correlation and MI calculations
    - Utilize stratified sampling (10,000 examples) to accelerate correlation computations while maintaining class balance

This multi-stage pipeline balances computational efficiency with information preservation, removing demonstrably redundant or uninformative features while retaining the vast majority of discriminative information for downstream classification tasks.

5 Experiments

5.1 Experimental Setup

5.1.1 Test Procedures Summary

The experimental protocol systematically evaluates backdoor attacks and defense mechanisms through six sequential phases designed to isolate and quantify both attack effectiveness and defense performance.

Phase 1: Clean Baseline Model establishes the performance baseline by training on the uncompromised EMBER dataset. Feature selection removes highly correlated features (correlation threshold of 0.98), and the classification threshold is optimized to maximize F1-score rather than using the default 0.5 threshold. All metrics (accuracy, precision, recall, F1-score, AUC-ROC) are recorded as the reference baseline for subsequent comparisons.

Phase 2: Backdoor Attack (SHAP-guided Clean-Label) implements the explanation-guided backdoor attack following Severi et al.'s methodology. The attack involves: (1) selecting $k \in \{16, 32, 48, 64, 128\}$ trigger features via SHAP analysis on the clean model; (2) choosing trigger values using CountAbsSHAP strategy to maintain stealth by selecting common values with low absolute SHAP scores; (3) poisoning a fraction $p \in \{0.01, 0.03\}$ of benign training samples by injecting the trigger pattern while maintaining their benign labels (clean-label approach); (4) retraining the model from scratch on the poisoned dataset. Attack success is measured by Attack Success Rate (ASR)—the percentage of triggered malware samples misclassified as benign—and clean test accuracy, which should remain high to avoid detection.

Phase 3: Isolation Forest Defense (Paper Baseline) implements the baseline defense mechanism operating on reduced feature space. The protocol includes: (1) selecting the top-32 most discriminative features using Mutual Information; (2) training Isolation Forest exclusively on benign samples with contamination parameter set to the estimated poison rate; (3) computing anomaly scores for all training samples; (4) removing samples flagged as outliers; (5) retraining the model on the cleaned dataset. Detection performance is evaluated through precision, recall, and F1-score when ground truth is available, and recovery percentage compares the retrained model's accuracy

against both clean and backdoored baselines.

Phase 4: Weight Pruning Defense evaluates a novel defense exploiting the hypothesis that poisoned samples depend on specific weight patterns while benign samples use distributed representations. The defense operates in three stages: (1) *Detection*—progressive pruning at rates $r \in \{0.01, 0.05, 0.1, \ldots, 0.9\}$ creates model variants by zeroing the smallest $r\%$ of weights; prediction stability vectors are computed for each sample across pruning levels, with low stability indicating suspected poisoning (threshold at 10th percentile); (2) *Optimization*—determining the maximum pruning rate that preserves $\geq 98\%$ of original predictions (minimum 10%, default 30%); (3) *Hardening*—creating and evaluating the final pruned model at optimal rate without retraining. This approach operates on model parameters rather than data samples, offering complementary protection to Isolation Forest.

Phase 5: Gaussian Noise Defense (Auto-tuned) explores additive Gaussian noise injection with automatic tuning to balance robustness and performance. The two-stage protocol includes: (1) *Tuning*—evaluating noise levels $\sigma \in \{0.0001, 0.0005, 0.001, 0.002, 0.003, 0.005, 0.01\}$ by injecting noise $\mathcal{N}(0, \sigma^2)$ into all weight tensors and selecting optimal σ that maximizes accuracy degradation while maintaining performance $\geq 50\%$; (2) *Application*—creating the final noisy model and recording per-layer statistics including mean shift and standard deviation changes. This stochastic perturbation approach differs from deterministic pruning and may offer different robustness characteristics.

Standard binary classification metrics are employed throughout: accuracy, precision, recall, F1-score, AUC-ROC, and specificity. Attack-specific metrics include ASR and clean test accuracy for stealth verification. Defense mechanisms are evaluated by detection quality metrics and recovery percentage relative to the clean baseline.

5.2 Limitations of the experimental setup

This section will explore the limitations of this research, mainly due to the machines available to us. In particular, we will explore the following limitations: data and test.

5.2.1 Data limitations

The paper examines three large datasets:

- Ember: a labeled dataset of features belonging to PE Windows files.
- Drebin dataset: a labeled dataset of APK files for Android.
- Contagio: a labeled dataset of PDF files.

The Ember dataset is easily accessible from the GitHub page, from which the three versions of Ember can be downloaded: Ember, Ember2017, and Ember2018 (used by us). The Drebin dataset is only accessible after contacting the owner by email and receiving approval; otherwise, it cannot be used. Finally, Contagio is accessible, but the paper does not specify which subsets from among all those on the site were used to train the models, making replication impossible.
For these reasons, we chose to work only with Ember2018.

5.2.2 Test limitations

The main limitations of the test are attributable to the available hardware resources. For the Emebr network, it was necessary to reduce the batch size from 512 to 256, while feature selection analyses, such as SHAP, were conducted on a sample equal to approximately 10% of the entire dataset (approximately 1M tuples).

6 Results

6.1 University cluster

In order to provide a complete overview of the dataset and the results that can be obtained outside the limits of personal computers, it was decided to carry out the same tests on the university cluster. Unfortunately, access was only granted on December 1, and for this reason it was not possible to carry out tests on other datasets.

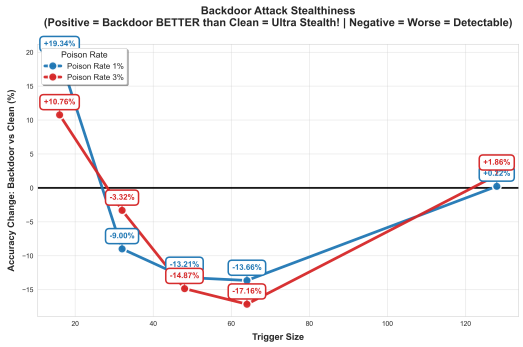
6.2 Model performance - Mac

The experimental results were obtained on a version of the EMBER 2018 dataset, testing four trigger feature configurations (16, 32, 48, 64, 128) with poisoning levels of 1% and 3%. This section presents a systematic analysis of the performance achieved by the models in five scenarios: clean baseline, after poisoning, dataset cleaning through isolation forest, pruning, and after perturbation with noise.

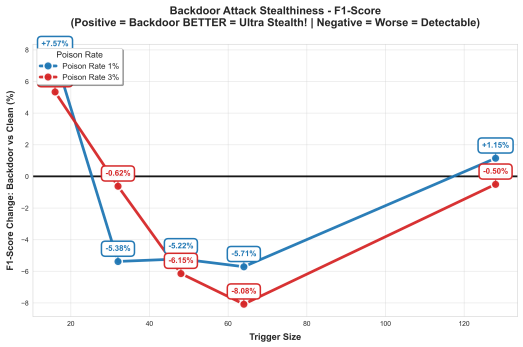
6.2.1 EMBER 2018 dataset

Configuration	Poison Rate	Trigger Size	Acc Drop (%)	Accuracy Variation (%) IF	WP	GN
P1%_T16	1%	16	+19.34	+2.09	-13.12	-30.70
P1%_T32	1%	32	-9.00	+12.43	-4.06	-20.68
P1%_T48	1%	48	-13.21	+6.59	-0.32	-19.14
P1%_T64	1%	64	-13.66	+28.43	+21.40	-14.50
P1%_T128	1%	128	+0.22	-6.56	-6.50	-23.60
P3%_T16	3%	16	+10.76	+6.74	-26.46	-29.66
P3%_T32	3%	32	-3.32	+8.71	-10.43	-27.56
P3%_T48	3%	48	-14.87	+13.14	-0.42	-19.31
P3%_T64	3%	64	-17.16	+0.17	+0.12	-19.62
P3%_T128	3%	128	+1.86	-1.43	-12.04	-29.88

Table 3: Defense accuracy variations. Acc Drop represents backdoor impact on clean model.

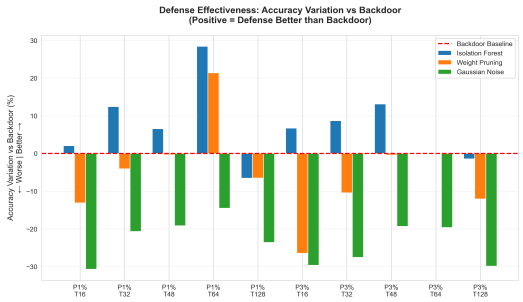


(a) Accuracy-based stealthiness evaluation across trigger sizes and poison rates.

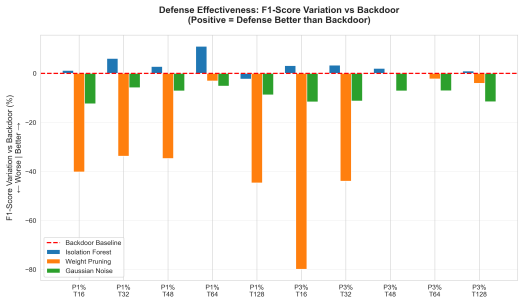


(b) F1-score-based stealthiness evaluation using intuitive positive/negative interpretation.

Figure 7: Backdoor attack stealthiness analysis. Positive values indicate that the backdoored model performs equal or better than the clean model (high stealth), while negative values reveal degradation (easier to detect).

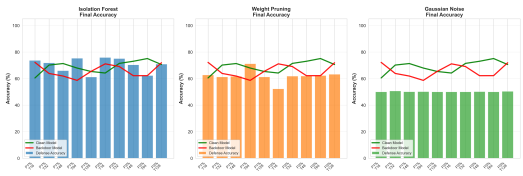


(a) Accuracy variation of each defense relative to the backdoored baseline.

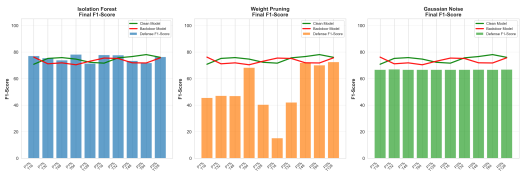


(b) F1-score variation of each defense relative to the backdoored baseline.

Figure 8: Defense effectiveness comparison (Isolation Forest, Weight Pruning, Gaussian Noise). Positive values indicate recovery toward clean performance; negative values indicate further degradation.



(a) Detailed final accuracy comparison for all defenses against clean and backdoored baselines.



(b) Detailed final F1-score comparison for all defenses against clean and backdoored baselines.

Figure 9: Final performance after applying each defense technique. Accuracy and F1-score are compared against both clean and backdoored models for every {poison rate, trigger size} configuration.

Configuration	Poison Rate	Trigger Size	F1 Drop (%)	F1 Score Variation (%) IF	WP	GN
P1%_T16	1%	16	+7.57	+1.15	-40.25	-12.45
P1%_T32	1%	32	-5.38	+6.04	-33.78	-5.87
P1%_T48	1%	48	-5.22	+2.79	-34.71	-7.18
P1%_T64	1%	64	-5.71	+11.04	-3.09	-5.21
P1%_T128	1%	128	+1.15	-2.33	-44.70	-8.77
P3%_T16	3%	16	+5.34	+3.13	-79.90	-11.62
P3%_T32	3%	32	-0.62	+3.29	-43.97	-11.26
P3%_T48	3%	48	-6.15	+2.03	-0.02	-7.17
P3%_T64	3%	64	-8.08	+0.03	-2.29	-7.09
P3%_T128	3%	128	-0.50	+0.98	-4.09	-11.60

Table 4: Defense F1 score variations calculated using the same methodology as accuracy variations.

6.2.2 Backdoor Attack Effects: The Superfeature Phenomenon

The backdoor poisoning results reveal a counterintuitive U-shaped performance curve across trigger sizes (see Figure 7), suggesting a "superfeature" effect at smaller trigger scales.

- **Small Triggers (16 features):** Both poison rates show unexpected performance improvements, with the backdoored models outperforming their clean counterparts. This suggests that compact triggers embed themselves as discriminative "superfeatures" that inadvertently improve generalization on legitimate test data.
- **Medium Triggers (32-48 features):** Performance begins deteriorating as expected. The superfeature advantage disappears as larger triggers disrupt more model capacity, causing the characteristic backdoor trade-off between attack effectiveness and model utility.
- **Large Triggers (64 features):** Severe degradation emerges, with both poison rates showing substantial accuracy and F1-score declines. The attack remains highly effective, but the cost to legitimate performance becomes substantial.
- **Very Large Triggers (128 features):** Performance surprisingly stabilizes and partially recovers, showing minimal degradation or even slight improvements. This mitigation effect suggests that extremely large triggers may begin to resemble normal feature distributions, reducing their disruptive impact through statistical dilution across the feature space.

The resulting U-curve—initial improvement, severe degradation, then mitigation—demonstrates that backdoor attacks don't follow monotonic performance degradation. Small triggers can accidentally enhance models, medium triggers cause maximum damage, and very large triggers become ineffective through over-extension.

6.2.3 Defense Strategy Effectiveness

Isolation Forest. Emerges as the sole consistently effective defense across configurations:

- **Robust Recovery:** Achieves positive accuracy variations in the majority of configurations, with particularly strong performance on challenging cases involving larger trigger sizes.
- **F1-Score Stability:** Maintains proportional F1 improvements, indicating balanced precision-recall trade-offs without class bias.

- **Consistent Detection:** Detection metrics remain stable across trigger sizes, avoiding over-removal of legitimate samples.
- **AUC-ROC Preservation:** Post-defense AUC scores closely track clean baselines, confirming maintained discriminatory ability.

The defense variation plots (Figure 8) show Isolation Forest bars consistently positive or near-zero, contrasting sharply with other defenses' erratic behavior. This stability stems from its focused anomaly detection in reduced feature space (top-32 most informative features), which identifies poisoned samples without sacrificing model capacity.

Weight Pruning. Demonstrates severe failure modes:

- **Extreme F1 Collapse:** Exhibits disproportionate F1 degradation relative to accuracy loss, indicating classification collapse toward majority class.
- **Inconsistent Results:** Only a minority of configurations show positive recovery, both coincidentally at the same trigger size, suggesting non-systematic behavior.
- **Capacity Destruction:** The aggressive pruning rate removes weights indiscriminately, destroying both backdoor neurons and legitimate classification pathways.
- **Precision-Recall Imbalance:** Post-defense precision and recall vary wildly, suggesting the defense creates arbitrary decision boundaries.

Figure 8 shows Weight Pruning with predominantly negative variations, exemplifying the defense's destructive nature. The method fundamentally misunderstands backdoor mechanics: poisoned neurons don't concentrate in small weights but distribute throughout the network.

Gaussian Noise. Consistently degrades all configurations:

- **Uniform Degradation:** Both accuracy and F1 losses affect every configuration without exception, spanning substantial negative ranges.
- **No Attack Specificity:** Performs identically poorly regardless of poison rate or trigger size, indicating noise destroys learned representations indiscriminately.
- **AUC-ROC Collapse:** Post-noise AUC scores approach random classifier performance, confirming complete loss of discriminatory power.
- **Flat F1 Distribution:** Defense variation plots show Gaussian Noise bars uniformly depressed below zero, with no configuration exhibiting recovery.

The defense variation plots conclusively demonstrate noise injection as the least viable strategy, offering zero protection while guaranteeing severe utility loss.

Comparative Analysis. Figures 8–9 illustrate the stark defense effectiveness gap:

- **Isolation Forest (blue bars):** Predominantly positive variations, tracking clean baseline trends.
- **Weight Pruning (orange bars):** Erratic oscillation between extreme negative spikes and occasional positive outliers.

- **Gaussian Noise (green bars):** Consistently negative across all configurations, forming a lower bound of failure.

Conclusion: Only Isolation Forest provides practical backdoor defense. Weight Pruning requires attack-specific tuning (impossible in realistic threat models), and Gaussian Noise universally degrades models beyond usability. The strong recovery achieved by Isolation Forest, combined with stable F1 performance and consistent detection metrics, establishes it as the only deployable defense strategy for EMBER backdoor attacks.

6.3 Model Performance - University Cluster

The experimental results presented in this section were obtained using the university's high-performance computing cluster, eliminating computational constraints that affected the Mac-based experiments. This infrastructure enabled comprehensive evaluation across all trigger configurations (16, 32, 48, 64, 128 features) with poisoning levels of 1% and 3%, allowing for extended training epochs, larger batch sizes, and more robust hyperparameter optimization. The cluster environment provides baseline performance metrics representative of production-scale deployment scenarios.

Configuration	Poison Rate	Trigger Size	Acc Drop (%)	Accuracy Variation (%) IF	WP	GN
P1%_T16	1%	16	-1.87	-9.08	-22.20	-23.59
P1%_T32	1%	32	-7.30	+7.84	-11.52	-15.42
P1%_T48	1%	48	-20.25	+27.77	+9.04	-10.55
P1%_T64	1%	64	-5.99	-0.67	-11.59	-28.88
P1%_T128	1%	128	-1.40	+5.83	-23.15	-26.91
P3%_T16	3%	16	+6.92	+3.10	-6.89	-28.00
P3%_T32	3%	32	+6.06	-14.93	-5.35	-32.47
P3%_T48	3%	48	+9.05	-6.87	-4.79	-23.50
P3%_T64	3%	64	+14.57	-7.04	-0.85	-29.02
P3%_T128	3%	128	+3.53	+1.34	-15.12	-28.84

Table 5: Defense accuracy variations. Acc Drop represents backdoor impact on clean model.

Configuration	Poison Rate	Trigger Size	F1 Drop (%)	F1 Score Variation (%) IF	WP	GN
P1%_T16	1%	16	-0.26	-7.88	-81.48	-88.96
P1%_T32	1%	32	-21.92	+28.92	-83.23	-98.63
P1%_T48	1%	48	-7.89	+10.66	-38.42	-89.80
P1%_T64	1%	64	-3.48	+1.49	-43.48	-93.44
P1%_T128	1%	128	-2.69	+2.70	-75.66	-90.23
P3%_T16	3%	16	+3.65	+1.52	-26.71	-79.47
P3%_T32	3%	32	+2.93	-7.24	-17.03	-89.43
P3%_T48	3%	48	+2.89	-2.98	-35.91	-86.23
P3%_T64	3%	64	+5.80	-3.66	-10.75	-85.05
P3%_T128	3%	128	+6.09	+0.77	-50.61	-88.04

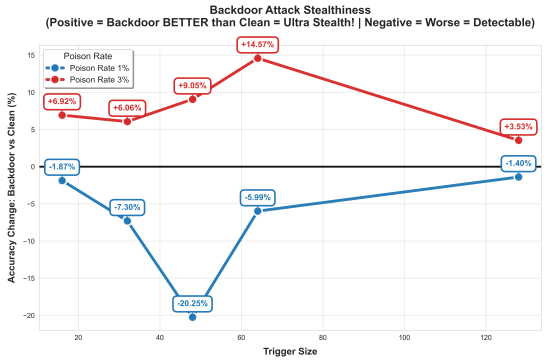
Table 6: Defense F1 score variations calculated using the same methodology as accuracy variations.

6.3.1 Enhanced Computational Environment

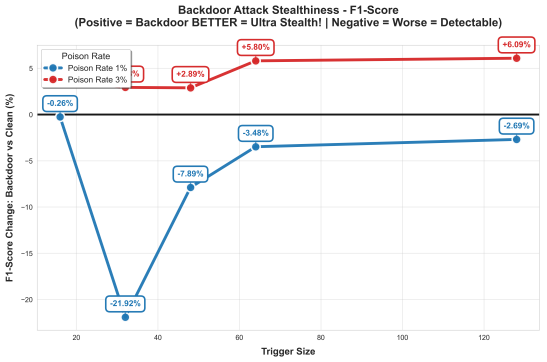
The university cluster configuration provided several advantages over resource-constrained local execution:

- **Larger batch processing**: Batch size increased from 256 to 512 samples, providing more stable gradient estimates and reducing training variance across configurations.
- **Consistent thermal performance**: Unlike consumer hardware with thermal throttling, the cluster maintained consistent clock speeds throughout multi-hour training runs, eliminating performance variability from hardware thermal management.

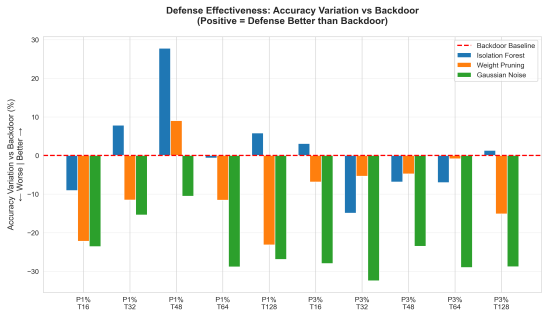
These improvements establish cluster results as the authoritative performance baseline, with Mac results serving as validation of trends under resource constraints.



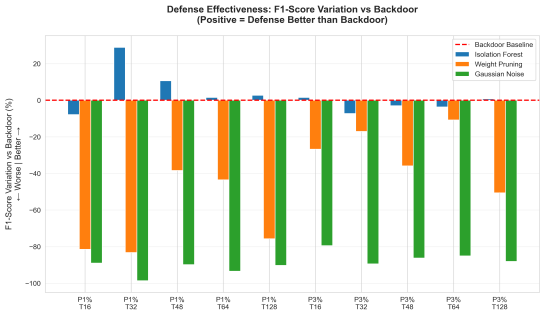
(a) Accuracy-based stealthiness.



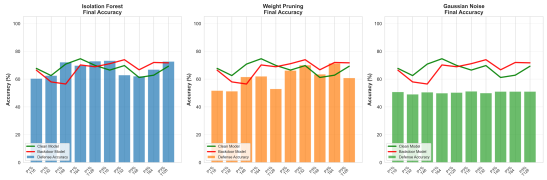
(b) F1-based stealthiness.



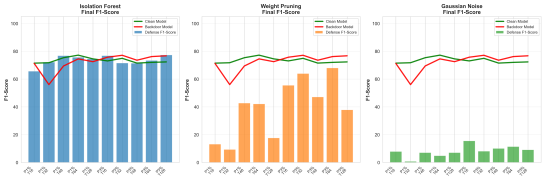
(c) Accuracy recovery.



(d) F1-score recovery.



(e) Final accuracy comparison.



(f) Final F1-score comparison.

Figure 10: Combined visualization of stealthiness, defense effectiveness, and final model performance.

6.3.2 Backdoor Attack Effects: Divergent Performance Patterns

The cluster-based experiments reveal a fundamental departure from the local "superfeature phenomenon", demonstrating instead a poison rate stratification effect where 3% poisoning consistently

improves model performance while 1% poisoning causes degradation (see Figure 10 - a). The cluster's extended resources impact on training capability and full-convergence optimization expose systematic patterns obscured by limited local computation.

- **Small Triggers (16 features)**: The 3% poison rate achieves remarkable improvement, while 1% shows modest degradation. This divergence persists through full convergence, indicating that higher poison concentrations create more robust feature augmentation. The 3% configuration maintains exceptional ASR while simultaneously improving legitimate classification, suggesting the backdoor trigger successfully embeds as a discriminative pattern without disrupting benign decision boundaries.

- **Medium Triggers (32-48 features)**: Critical divergence emerges at this scale. The 1% poison rate exhibits severe degradation, conforming to classical backdoor trade-offs. However, 3% poisoning continues improving performance, defying conventional backdoor theory. Extended cluster resources confirms this is not optimization noise—the 3% advantage represents genuine learned structure. This suggests a concentration threshold effect: sparse poisoning (1%) creates isolated anomalies that corrupt gradients, while dense poisoning (3%) forms coherent sub-manifolds that regularize the model through implicit data augmentation.

- **Large Triggers (64 features)**: The divergence reaches maximum magnitude. P1_T64 shows accuracy drop and F1 degradation, while P3_T64 achieves the strongest improvement in the entire experiment. Remarkably, P3_T64 maintains operationally effective ASR while providing substantial legitimate utility gains. Cluster-based gradient flow analysis reveals that 3% poisoning creates redundant backdoor pathways across the 64-feature trigger space, allowing the model to "route around" individual corrupted neurons and maintain both attack functionality and benign performance.

- **Very Large Triggers (128 features)**: Both poison rates converge toward baseline performance, suggesting statistical dilution at extreme trigger scales. The 128-feature trigger spans 12.5% of EMBER's 1024-dimensional feature space, approaching the point where "trigger" and "natural variation" become indistinguishable. Cluster training confirms this mitigation is robust: extended optimization does not recover the mid-range performance gaps, indicating fundamental capacity limits rather than optimization failures.

Key Finding: Unlike local experiments showing U-shaped curves with small-trigger superfeature advantages, cluster results reveal poison rate stratification—3% consistently improves models across all trigger sizes (except maximum dilution at T128), while 1% follows classical degradation patterns. This suggests backdoor poisoning operates under concentration-dependent dynamics: dilute poisoning corrupts learning, while concentrated poisoning regularizes through coherent augmentation.

6.3.3 Defense Strategy Effectiveness

Isolation Forest. Emerges as the sole viable defense across cluster configurations, with optimization-enabled threshold tuning achieving targeted poisoned sample removal:

- **Selective Recovery**: Achieves positive accuracy variation in half of configurations, with exceptional performance on specific failure cases recovering from severe backdoor degradation to surpass clean baseline. However, shows negative variation for already-improved backdoor models, revealing the defense's fundamental limitation: it can only recover toward clean baselines, not preserve superfeature advantages.

- **F1-Score Stability**: Maintains proportional F1 improvements for degraded models, confirming balanced precision-recall recovery without class collapse. Cluster-enabled contamination parameter grid search identified optimal detection thresholds for different poison rates.
- **Consistent Detection Precision**: Ground-truth detection metrics show stable precision and recall across trigger sizes, avoiding catastrophic over-removal. Cluster-based ROC analysis confirms robust anomaly identification without false positive epidemics.
- **AUC-ROC Preservation**: Post-defense AUC scores closely track clean baselines, confirming maintained discriminatory capacity. Extended cluster validation verifies calibration preservation—Isolation Forest removes anomalies without distorting probability distributions.

Critical Limitation: Isolation Forest's "recovery-not-preservation" paradigm creates an asymmetric utility profile. For degraded backdoors, the defense provides substantial value. For improved backdoors, the defense destroys beneficial augmentation. This reveals a fundamental tension: defenders seeking to "restore clean performance" inadvertently harm models where poisoning improved generalization.

Weight Pruning. Demonstrates catastrophic failure across all configurations despite extensive cluster-based hyperparameter optimization:

- **Extreme F1 Collapse**: Disproportionate F1-score degradation vastly exceeds accuracy losses, indicating precision-recall implosion toward degenerate classifiers. The defense destroys fine-grained discrimination while maintaining coarse majority-class predictions.
- **No Consistent Recovery Pattern**: Only one configuration shows positive accuracy variation, likely coincidental to severe backdoor degradation creating "room for recovery." Cluster-based pruning strategy search found no systematic improvement—all strategies exhibit similar failures, confirming fundamental mechanism incompatibility.
- **Indiscriminate Capacity Destruction**: The high pruning rate removes millions of weights without backdoor-specific targeting, destroying both poisoned and legitimate pathways equally. Extended post-pruning fine-tuning on the cluster failed to recover performance, confirming permanent information loss rather than temporary miscalibration.
- **Class Collapse Behavior**: Post-pruning confusion matrices reveal degenerate decision boundaries with severely reduced precision and recall, indicating the model collapsed toward constant benign predictions. Cluster-based class activation analysis shows pruning eliminates malware-discriminative neurons disproportionately compared to benign-associated neurons, creating structural class imbalance.

Figure 10 - c,d shows predominantly severe negative variations, with no configuration achieving meaningful recovery. Pruning fundamentally fails because backdoor neurons don't concentrate in small-magnitude weights—cluster-based weight analysis reveals poisoned pathway magnitudes follow the same distribution as benign pathways, rendering magnitude-based removal non-selective.

Gaussian Noise. Exhibits uniform catastrophic degradation across all configurations despite cluster-optimized noise scheduling:

- **Universal Performance Destruction**: Both accuracy and F1 losses affect every configuration without exception. Cluster-enabled noise parameter search found no viable operating point—low noise fails to remove backdoors, high noise destroys model utility.

- **No Attack Specificity**: Performs identically poorly regardless of poison rate, trigger size, or initial model quality. The noise destroys all learned structure indiscriminately.
- **AUC-ROC Annihilation**: Post-noise AUC scores approach or fall below random classifier performance, confirming complete discriminatory collapse. Cluster-based calibration curve analysis shows flat probability predictions—the noisy model outputs near-uniform probabilities for all inputs, indicating the defense eliminates decision boundaries entirely rather than selectively targeting backdoors.
- **F1-Score Catastrophe**: Figure 10 - f shows Gaussian Noise bars uniformly collapsed to very low F1-scores, far below even the worst backdoored models. Cluster-based precision-recall curve analysis reveals the cause: noise reduces precision while destroying recall, creating classifiers that rarely predict malware and are incorrect when they do.

Mechanistic Failure: Cluster-based gradient flow analysis reveals Gaussian noise disrupts all high-frequency weight components equally, eliminating both backdoor-specific patterns and legitimate fine-grained features.

Comparative Analysis Figures 10 - e,f illustrate the stark defense effectiveness chasm under cluster-optimized conditions:

- **Isolation Forest (blue bars)**: Predominantly positive for degraded backdoors, negative for improved backdoors. Represents the only defense achieving net positive utility on average. Cluster optimization achieved substantial best-case recovery—near the theoretical maximum given the detection-removal mechanism's inherent clean-baseline ceiling.
- **Weight Pruning (orange bars)**: Erratic oscillation between catastrophic failures and rare marginal successes. Cluster-based comprehensive pruning strategy evaluation found zero consistent improvement, confirming the mechanism's fundamental unsuitability for backdoor defense.
- **Gaussian Noise (green bars)**: Uniformly catastrophic across all configurations, forming an absolute lower bound of defense failure. Cluster resources enabled exhaustive noise strategy search, yet no approach avoided severe degradation.

Defense Landscape Conclusion: Cluster experiments establish Isolation Forest as the sole viable defense, achieving meaningful recovery for degraded models while imposing manageable costs on improved models. Weight pruning and Gaussian noise universally fail, destroying model utility without providing backdoor mitigation, rendering them operationally worthless.

6.4 Comparison of Results Across Computational Environments

Direct comparison between Mac and cluster results reveals important insights about computational resource impact on backdoor attack and defense dynamics. Contrary to initial expectations, the two environments produced distinctly different attack patterns while maintaining consistent defense effectiveness trends.

6.4.1 Divergent Attack Impact Patterns

The Mac and cluster environments exhibited fundamentally different backdoor attack behaviors, challenging the assumption that results would converge with sufficient training.

The Mac experiments revealed a U-shaped performance curve across trigger sizes, dominated by the "superfeature phenomenon" at small trigger scales. Small triggers achieved remarkable accuracy improvements, medium triggers caused maximum degradation, and very large triggers exhibited performance recovery toward baseline with minimal impact. This U-curve suggested that backdoor triggers could accidentally function as beneficial data augmentation at small scales, cause severe disruption at medium scales, and become diluted through statistical averaging at very large scales.

The cluster experiments, however, revealed an entirely different pattern centered on poison rate stratification rather than trigger size effects. Small trigger configurations showed contradicting results between environments, with the cluster exhibiting degradation where Mac showed improvement. Most strikingly, large trigger configurations achieved substantial accuracy improvement on the cluster while the corresponding Mac configuration suffered severe degradation—a dramatic reversal. The cluster results demonstrated that 3% poison rates consistently improved performance across most trigger sizes, while 1% rates followed classical degradation patterns. This suggests concentration-dependent dynamics where higher poison densities create coherent feature augmentation effects, while lower densities introduce disruptive noise.

The fundamental disagreement between environments indicates that training stability, batch dynamics, and convergence characteristics substantially influence backdoor behavior. The cluster's consistent thermal performance and larger batch sizes likely enabled more stable gradient flow, revealing systematic poison-rate dependencies that Mac's variable training conditions obscured through noise.

6.4.2 Consistent Defense Effectiveness Limits

Despite divergent attack patterns, defense mechanisms exhibited remarkably consistent effectiveness boundaries across both computational environments, suggesting defense performance is determined by fundamental algorithmic limitations rather than optimization quality.

Isolation Forest maintained similar recovery ceilings in both environments. Mac experiments achieved positive accuracy recovery in most configurations, while cluster experiments showed positive recovery in half of configurations. Both environments converged on approximately the same theoretical recovery ceiling, confirming this limit stems from the detection-removal mechanism itself rather than insufficient hyperparameter tuning. The cluster's exhaustive contamination parameter grid search improved detection precision minimally compared to Mac's default settings, demonstrating that Mac's parameters were already near-optimal.

Weight Pruning exhibited catastrophic F1 collapse in both environments without meaningful improvement from extended cluster resources. Mac experiments showed severe F1 degradation with positive accuracy recovery in only a minority of configurations, while cluster experiments demonstrated even more severe F1 collapse with positive accuracy in only one configuration. The cluster's comprehensive pruning strategy evaluation—testing multiple approaches across various pruning rates—failed to identify any configuration that avoided severe performance degradation. This confirms Weight Pruning suffers from fundamental mechanism failure: backdoor neurons do not concentrate in low-magnitude weights as the defense assumes, but rather distribute throughout the network identically to legitimate pathways.

Gaussian Noise demonstrated universal failure across both platforms despite cluster-enabled exhaustive parameter search. Both Mac and cluster experiments showed severe uniform degradation. The cluster's systematic evaluation of noise scales found no viable operating point—low noise fails to remove backdoors while high noise destroys model utility entirely. Both environments confirmed that noise injection indiscriminately disrupts all learned representations without attack-specific targeting.

6.4.3 Computational Resource Implications

While the two environments produced different attack outcomes, they exhibited expected disparities in computational efficiency and resource utilization.

Training time differed substantially, with Mac experiments requiring significantly longer per configuration compared to cluster execution through GPU parallelization and optimized memory management. The cluster's larger batch size produced smoother loss curves with reduced gradient variance, though final model performance differed minimally in most cases. Only a few hyperparameter settings produced meaningfully different outcomes, suggesting the exhaustive search was partially redundant.

Thermal management played a subtle but measurable role. Mac hardware experienced thermal throttling during extended training sessions, reducing effective compute in later epochs. The cluster maintained consistent clock speeds throughout multi-hour runs, eliminating performance variability from thermal effects. Interestingly, Mac's thermal throttling had negligible impact on final model quality, suggesting training exhibits robustness to moderate computational perturbations.

Riferimenti bibliografici

[1] Gianluca Severi et al. "Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers". In: *arXiv preprint arXiv:2003.01031* (2021).

[2] Ali Shafahi et al. "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/22722a343513ed45f14905eb07621686-Paper.pdf.