ⵜⴰⵖⵡⴰⵍⵜ ⵏ ⵜⵙⴻⵎⵎⵓⵔⵉⵏ
FACULTÉ DES SCIENCES

MARRAKECH
جامعة القاضي عياض
UNIVERSITÉ CADI AYYAD

Semlalia

كليـــــــــة العلوم
FACULTY OF SCIENCE

AI System Security Project

Detecting Data Poisoning in Convolutional Neural Networks Using Probabilistic Feature Modeling

Realized by :

M'KOUKA Btissam

Supervised by :

Dr. Mariya Ouaissa

Session Year : 2025-2026

Acknowledgements

In this project, we investigate the impact of data poisoning on deep neural networks and develop a probabilistic method to detect poisoned samples. We first trained a convolutional neural network (CNN) on the CIFAR-10 dataset and observed its performance under clean and poisoned conditions. Poisoning was introduced by flipping labels for 15% of samples from the airplane and ship classes, totaling 1,500 poisoned points. The introduction of poisoned data led to a noticeable drop in test and training accuracy, particularly for the targeted classes, highlighting the vulnerability of neural networks to label-based attacks.

To identify poisoned samples, we implemented a probabilistic detection pipeline based on penultimate-layer features. For each class, we extracted deep features from a clean subset of samples, modeled them as a multivariate Gaussian, and computed log-likelihoods for all training samples. Samples with log-likelihoods below the 5th percentile of clean features were flagged as suspicious. This approach allows effective identification of poisoned data, and metrics such as precision, recall, and F1-score were computed to quantify detection performance. True positive examples were visualized to demonstrate the method's ability to detect anomalies in the feature space.

Our study demonstrates that modeling class-wise feature distributions provides a principled and interpretable approach to detecting data poisoning in deep learning models, and it emphasizes the importance of robust dataset curation and anomaly detection techniques in machine learning pipelines.

Table of Contents

List of Figures

Chapter 1

Introduction and Background

1.1 Introduction

Artificial Intelligence (AI) has become a transformative technology across multiple domains, including healthcare, finance, transportation, and cybersecurity, due to its ability to analyze large volumes of data and make informed predictions. However, the widespread integration of AI has also introduced new vulnerabilities, particularly the susceptibility of AI systems to data poisoning attacks.

Data poisoning attacks involve the deliberate injection of malicious or manipulated data into the training dataset of a machine learning model. These attacks can degrade model performance, induce incorrect predictions, and compromise the reliability of AI systems. For example, Biggio and la[1] demonstrated that adversaries could poison training data to degrade the performance of Support Vector Machines, causing incorrect classifications and potentially significant financial or reputational damage.

The importance of addressing data poisoning attacks is underscored by the risks they pose to AI system integrity. Such attacks can reduce decision-making accuracy, produce erroneous outputs, and erode trust in AI applications. Organizations relying on AI models risk severe consequences if adversaries successfully manipulate their training data. Korada [2] highlights the critical necessity of securing training datasets to ensure the reliability and robustness of AI systems which are increasingly deployed in critical sectors, ensuring their resilience against adversarial manipulations is essential. Detecting and mitigating data poisoning is crucial not only for maintaining model performance but also for safeguarding sensitive decision-making processes. Research has increasingly focused on developing methods to detect, prevent, and counteract data poisoning attacks.

Despite the effectiveness of modern deep learning models, such as Convolutional Neural Networks (CNNs), they remain vulnerable to subtle manipulations in training data. Small-scale modifications, such as label flipping or the introduction of backdoor triggers, can significantly impact model performance. This project focuses on understanding and mitigating these vulnerabilities in the context of image classification using CIFAR-10.

1.2 Objectives

The primary goals of this project are structured into three phases :

1. **Training and Evaluation of CNNs :**

   — Train a CNN on the **CIFAR-10** dataset, separately using both the **clean** and the **poisoned** versions of the training data.
   — Evaluate the performance of the trained models using key metrics, including overall accuracy and class-wise accuracy.

2. **Poisoned Data Detection :**

   — Develop a **probabilistic detection pipeline** that utilizes the **penultimate-layer features** extracted from the trained CNN.
   — Model the distribution of **clean features** for each class by fitting **multivariate Gaussian distributions** (MGDs).
   — Compute **log-likelihoods** for all training samples based on their respective class MGD and flag samples that fall below a determined threshold as **anomalies**.

3. **Analysis and Visualization :**
   — Quantitatively assess the effectiveness of the detection pipeline using metrics such as **precision**, **recall**, and the **F1-score**.
   — Provide interpretability by **visualizing true positive detections**.
   — Compare the observed degradation in **model accuracy** due to data poisoning against the effectiveness of the proposed **detection mechanism**.

1.3 Contribution

By rigorously addressing the outlined objectives, this project makes significant contributions to the field of AI system security by quantifying the exact performance degradation of Convolutional Neural Networks (CNNs) on CIFAR-10 due to data poisoning. It implements a novel and practical probabilistic detection method that leverages the statistical properties of penultimate-layer features in deep networks to flag malicious inputs. Ultimately, this study offers AI practitioners actionable insights and a clear methodological framework for implementing practical mitigation strategies, thereby bridging the gap between standard Aritificial Intelligence performance evaluation and cybersecurity requirements to enhance the reliability and resilience of AI systems under adversarial conditions. [colorlinks=true, linkcolor=blue, urlcolor=blue]hyperref xcolor hyperref parskip

Chapter 2

Methodology

This study investigates the impact of data poisoning on machine learning models using CIFAR-10 Dataset for image classification. The methodology involves systematically introducing poisoning attacks, training models on both clean and poisoned data, and evaluating performance degradation. The CIFAR-10 data set is susceptible to poisoning due to similarities among image classes, making it suitable for studying label-flipping attacks.

This chapter explains exactly how the experiment is structured, from dataset preparation to poisoning, CNN training, and statistical detection of anomalies.

2.1 Dataset and Poisoning Setup

2.1.1 CIFAR-10 Dataset

The **CIFAR-10 dataset** is a widely recognized benchmark in computer vision for image classification tasks. It comprises $\mathbf{60,000}$ color images, each of size $\mathbf{32 \times 32}$ pixels. The data is distributed evenly across **10 distinct classes**, with each class containing $\mathbf{5,000}$ training images and $\mathbf{1,000}$ test images.

The **10** classes included in the CIFAR-10 dataset (indexed 0 to 9) are :
0 : Airplane, **1** : Automobile, **2** : Bird, **3** : Cat, **4** : Deer, **5** : Dog, **6** : Frog, **7** : Horse, **8** : Ship, and **9** : Truck.

FIGURE 2.1 – CIFAR-10 Subset.

The dataset is split into a $\mathbf{50,000}$ image **Training Set** and a $\mathbf{10,000}$ image **Test Set**.

2.1.1.1 Poisoning Procedure

To study the impact of data poisoning, we selectively manipulated a fraction of the training dataset :

- **Targeted Poisoned Classes : Airplane** (Class 0) and **Ship** (Class 8).
- **Number of Poisoned Samples : 750** samples were selected from each targeted class, constituting approximately **15**% of the $5,000$ training images per class to keep the data balanced.
- **Poisoning Method : Label Flipping** was employed as a form of non-targeted integrity attack.
  - Samples originally labeled **Airplane** were flipped to the **Ship** label (Airplane → Ship).
  - Samples originally labeled **Ship** were flipped to the **Airplane** label (Ship → Airplane).
- **Total Poisoned Samples :** This setup results in a total of $\mathbf{1,500}$ poisoned samples being injected into the training set.
- **Clean Reference Set :** A dedicated clean reference we trust set of **500** samples per class was selected from the unpoisoned portion of the training data. This set is exclusively used for modeling typical class feature distributions and establishing thresholds for anomaly detection.

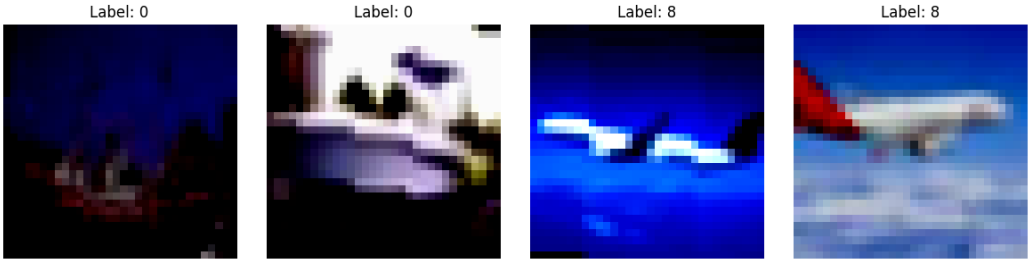

FIGURE 2.2 – Example of our Poisoned Data(Label flipped).

This setup allows us to evaluate both the degradation in model performance due to poisoning and the effectiveness of detection methods.

2.2 CNN Architecture and Training

2.2.1 Network Architecture

We implemented a Convolutional Neural Network (CNN) for CIFAR-10 classification. The architecture consists of the following components :

Convolutional Layers : The model contains 5 convolutional layers with ReLU activations and batch normalization. Each convolutional layer is followed by a 2×2 max pooling layer :

- **Conv1 :** $3 \to 64$ filters, kernel size 3×3
- **Conv2 :** $64 \to 128$ filters, kernel size 3×3
- **Conv3 :** $128 \to 256$ filters, kernel size 3×3
- **Conv4 :** $256 \to 128$ filters, kernel size 3×3
- **Conv5 :** $128 \to 128$ filters, kernel size 3×3

Fully Connected Layers : After flattening, the network includes 5 fully connected layers :

- **FC1 :** $128 \to 512$
- **FC2 :** $512 \to 256$
- **FC3 :** $256 \to 128$
- **FC4 :** $128 \to 64$
- **FC5 :** $64 \to 10$ (output layer)

Flattening Layer A flattening layer converts the output of the final convolutional layer into a vector before passing it to the fully connected layers.

This deep architecture allows the model to capture hierarchical features essential for image classification.

2.2.2 Training Setup

The model was trained with the following hyperparameters :
— **Optimizer :** Adam
— **Learning Rate :** 0.001
— **Batch Size :** 128
— **Epochs :** 50
— **Loss Function :** Cross-entropy loss
— **Device :** GPU (GoogleColab)

2.3 Probabilistic Detection Pipeline

To detect poisoned samples, we implemented a class-wise statistical anomaly detection approach :

1. **Feature Extraction :**
   Extract features from the penultimate layer(before output) of the CNN for all training samples. This layer captures high-level representations before classification.
2. **Build Clean Reference Features :**
   Select 500 clean samples we trust per class (excluding known poisoned indices) to model the typical feature distribution of each class.
3. **Fit Multivariate Gaussian :**
   Compute the mean vector μ_k and covariance matrix Σ_k for clean features of class k.
   Regularize covariance for numerical stability :

$$\Sigma_{\text{reg}} = \Sigma_k + \epsilon I, \quad \epsilon = 1 \times 10^{-6}$$

4. **Compute Log-Likelihoods :**
   For each sample x of class k, compute the log-likelihood :

$$\log\text{-likelihood}(x) = -\frac{1}{2}(f(x) - \mu_k)^T \Sigma_k^{-1} (f(x) - \mu_k)$$

   This measures how typical the sample is relative to clean features.
5. **Threshold and Flag Suspected Poisoned Samples :**
   Set a threshold at the 5th percentile(– since we 100% trust the subset) of clean log-likelihoods. Samples below this threshold are marked as suspected poisoned. Detection performance is evaluated using precision, recall, and F1-score.

2.3.1 Implementation Details

— **Frameworks :** PyTorch for CNN training and feature extraction, NumPy for statistical computations.
— **Indexed Dataset :** Maintains original dataset indices to compare flagged samples with ground truth.
— **Sequential DataLoader :** Ensures reproducibility during feature extraction.
— **Covariance Regularization :** Prevents numerical instability when inverting Gaussian covariance matrices.

2.3.1.1 Pipeline Summary

1. Train CNN on clean and poisoned datasets.
2. Extract features from the penultimate layer for all samples.
3. Build clean reference features for each class.
4. Fit class-wise Gaussian distributions to clean features.
5. Compute log-likelihoods and threshold at the 5th percentile.
6. Flag suspected poisoned samples and compute detection metrics (precision, recall, F1-score).
7. Visualize true positives for interpretability.

Chapter 3

Results and Analysis

3.1 Impact of Poisoning on Model Performance

To assess the impact of data poisoning, we compare the model's performance on clean vs. poisoned datasets. The poisoning targeted the *airplane* and *ship* classes, with 750 samples ($\approx 15\%$) from each class altered via label flipping.

3.1.1 Overall Accuracy

Dataset	**Test Accuracy**	**Training Accuracy**
Original (clean)	86.75%	92.79%
Poisoned (15%)	85.42%	90.09%

TABLE 3.1 – Overall accuracy comparison between clean and poisoned datasets.

The introduction of poisoned samples caused a drop in overall test accuracy by **1.33%** and training accuracy by **2.7%**, indicating that even moderate poisoning can degrade model performance.

3.1.2 Per-Class Accuracy

This table illustrates how poisoning reduces model accuracy, especially for the targeted classes (airplane and ship).

Class	**Test Accuracy (original)**	**Training Accuracy (original)**	**Test Accuracy (poisoned)**	**Training Accuracy (poisoned)**
Airplane	**92.10%**	**96.22%**	**90.20%**	**84.72%**
Automobile	95.40%	98.00%	92.50%	94.32%
Bird	77.70%	88.42%	86.80%	94.54%
Cat	70.10%	84.00%	63.90%	80.94%
Deer	89.70%	95.50%	87.10%	94.80%
Dog	80.70%	90.44%	72.30%	86.16%
Frog	91.40%	93.98%	88.30%	91.68%
Horse	88.00%	92.02%	92.20%	97.14%
Ship	**90.60%**	**94.60%**	**88.20%**	**80.44%**
Truck	91.80%	94.68%	92.70%	96.18%

TABLE 3.2 – Observed impact of poisoning on per-class accuracy.

Observations :
— **Airplane class :** Test accuracy dropped by **1.9%**, training accuracy dropped by **11.5%** .
— **Ship class :** Test accuracy dropped by **2.4%**, training accuracy dropped by **14.2%**.
These results illustrate that poisoning disproportionately affects the targeted classes in both training and testing, confirming the vulnerability of the model to label-flipping attacks.

3.2 Detection of Poisoned Samples

We applied the probabilistic detection pipeline to flag poisoned samples for airplane and ship classes using the penultimate-layer feature distributions :

-Extract features from the penultimate layer of the CNN.
-Build clean reference feature sets per class (500 clean samples).
-Fit multivariate Gaussian distributions (mean covariance) for clean features.
-Compute log-likelihoods of all class samples.
-Flag samples below the 5th percentile of clean log-likelihoods as suspected poisoned.
-Evaluate precision, recall, and F1-score against ground truth.

Any sample whose log-likelihood is below this 5th percentile is considered statistically unlikely to belong to the clean distribution of its class, so it's flagged as potentially poisoned.

3.2.1 Airplane Class Poisoned Detection Metrics

To evaluate the effectiveness of our probabilistic detection pipeline, we first saved the indices of poisoned samples during the poisoning process (label flipping). Using penultimate-layer features of the CNN, we fit a multivariate Gaussian on 500 clean airplane samples and computed log-likelihoods for all airplane-labeled samples.
We then set the detection threshold at the 5th percentile of clean sample log-likelihoods (-25.7362) and flagged samples with log-likelihoods below this threshold as suspected poisoned.

Metric	**Value**
Total airplane samples	5,000
True poisoned (ground truth)	750
Suspected poisoned	783
True Positives (TP)	552
False Positives (FP)	231
False Negatives (FN)	198
True Negatives (TN)	4,019
Precision	70,5%
Recall	73,6%
F1-score	72,0%

TABLE 3.3 – Detection metrics for poisoned samples in the airplane class.

The pipeline successfully flagged 552 of 750 poisoned samples. Some clean samples were incorrectly marked as poisoned (231 false positives), but overall detection is strong.

The image bellow Illustrate the effectiveness of the probabilistic detection pipeline by showing true positives for the airplane class it shows that true positives generally correspond to poisoned data, validating the method.

FIGURE 3.1 – Images flagged as poisoned by the detection method on airplane class.

3.2.2 Ship Class Poisoned Detection Metrics

Similarly, we applied the detection method for the ship class. Clean reference features were extracted from 500 clean ship samples. Thresholding at the 5th percentile of clean log-likelihoods (-25.3460) identified suspected poisoned samples.

Metric	Value
Total ship samples	5,000
True poisoned (ground truth)	750
Suspected poisoned	669
True Positives (TP)	408
False Positives (FP)	261
False Negatives (FN)	342
True Negatives (TN)	3,989
Precision	61%
Recall	54,4 %
F1-score	57,5%

TABLE 3.4 – Detection metrics for poisoned samples in the ship class.

Detection for ships is slightly less effective than airplanes, possibly due to feature overlap or more subtle distribution differences in penultimate-layer features.
The pipeline still correctly identifies a significant portion (54.4%) of poisoned samples.

The image bellow Illustrate the effectiveness of the probabilistic detection pipeline by showing true positives for the ship class.

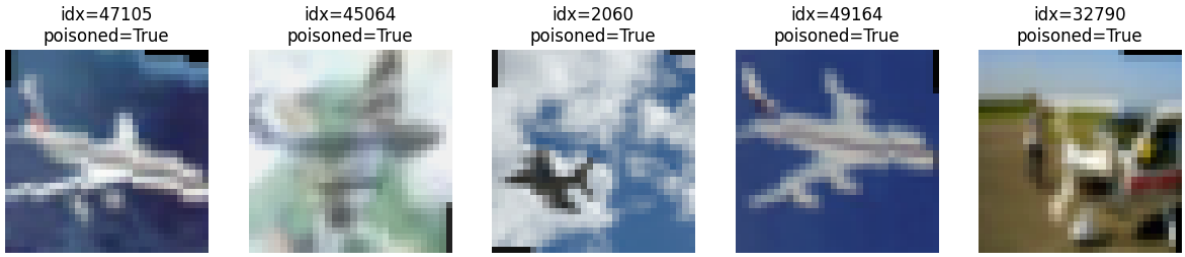

FIGURE 3.2 – Images flagged as poisoned by the detection method on ship class.

3.3 Summary of Detection Results

To evaluate the effectiveness of our poisoned sample detection pipeline, we compare detection metrics for the two targeted classes, airplane and ship, highlighting differences in performance and interpretability.

Class	True Poisoned	True Positives	False Positives	False Negatives	Precision	Recall	F1-score
Airplane	750	552	231	198	0.705	0.736	0.720
Ship	750	408	261	342	0.610	0.544	0.575

TABLE 3.5 – Comparison of poisoned sample detection metrics for the airplane and ship classes.

Key Points :
— The pipeline is more effective for the airplane class, likely due to cleaner feature separation in the penultimate layer.

— Thresholding at the 5th percentile of clean samples provides a practical and interpretable detection mechanism.
— Most true positives can be visually verified by displaying the flagged poisoned images.

Chapter 4

Discussion

4.1 Effectiveness of the Probabilistic Detection Pipeline

The probabilistic detection method we applied demonstrated promising results in identifying poisoned samples in the CIFAR-10 dataset. These results indicate that the approach is capable of detecting the majority of poisoned samples, especially for classes where poisoned samples exhibit strong deviations from the clean feature distribution.

4.2 Advantages

— **Interpretable :** The method uses class-wise Gaussian models and log-likelihoods, making it easy to understand why a sample is flagged. Low log-likelihood directly indicates deviation from the clean feature distribution.
— **Class-wise analysis :** Each class is modeled separately, allowing differentiated detection thresholds and accounting for natural variability between classes.
— **Minimal clean data needed :** Only a small subset of clean samples (e.g., 500–1000 per class) is required to build the reference Gaussian distribution, making the method data-efficient.

4.3 Limitations

— **Dependent on clean reference subset :** The detection relies heavily on the quality and representativeness of the clean samples. If clean data is not truly clean or too small, detection accuracy drops.
— **Gaussian assumption :** We assume that clean features follow a multivariate Gaussian distribution. Real-world features may not be perfectly Gaussian, which could affect detection, especially for complex or high-dimensional features.
— **Computational cost :** Inverting the covariance matrix and computing log-likelihoods for high-dimensional features can be computationally expensive, particularly for very deep networks or large datasets.

4.4 Insights for Improving AI System Security

— **Early detection of poisoned samples :** Using feature-based anomaly detection allows organizations to clean training data before training or fine-tuning models.
— **Class-specific thresholds :** Setting thresholds per class (like the 5th percentile) improves detection accuracy compared to a global threshold.
— **Potential for hybrid approaches :** Combining probabilistic methods with other techniques (e.g., clustering, gradient-based detection) could improve robustness and catch poisoned samples missed by one method alone.
— **Model monitoring :** This pipeline can be part of a continuous monitoring system to check for anomalies when new data is added or models are updated, enhancing long-term AI security.

Chapter 5

Conclusion

In this study, we investigated the vulnerability of deep learning models to data poisoning attacks and evaluated a probabilistic detection pipeline for identifying poisoned samples. We demonstrated that even a well-performing CNN can be significantly impacted when a relatively small fraction of training data is poisoned. Specifically, introducing 750 poisoned samples in both the airplane and ship classes (15% of each class) led to noticeable drops in per-class accuracy and overall model performance, highlighting the real-world risks of such attacks.

To counter this, we developed a feature-based anomaly detection method, leveraging the penultimate-layer activations of the CNN. By fitting a class-wise multivariate Gaussian on a small subset of clean samples, we modeled the natural distribution of each class's features. Computing log-likelihoods for all samples enabled us to flag low-likelihood points as potentially poisoned, with thresholds determined by the 5th percentile of clean likelihoods.

Our results showed that this approach could successfully detect a substantial proportion of poisoned samples :
Airplane class : 552 out of 750 poisoned samples detected (precision 0.705, recall 0.736, F1 0.720).
Ship class : 408 out of 750 poisoned samples detected (precision 0.610, recall 0.544, F1 0.575).

These findings confirm that statistical modeling of clean features is an effective tool for identifying anomalous samples introduced by adversaries. The method is interpretable, class-specific, and data-efficient, requiring only a modest clean reference set to build robust detection models. Visual inspection of top anomalous samples further confirmed the accuracy of the detection pipeline, validating its practical applicability.

However, limitations remain. The approach assumes that clean features follow a Gaussian distribution and that the clean subset is representative. Detection performance may decrease if these assumptions do not hold, especially in more complex datasets or high-dimensional feature spaces. Additionally, the computational cost of covariance inversion and log-likelihood computation grows with feature dimensionality, which should be considered for large-scale deployments.

Overall, this study underscores the importance of securing AI systems against poisoning attacks. It provides a practical, statistically grounded methodology to detect poisoned data before it compromises model performance. By integrating such detection pipelines into the AI development lifecycle, organizations can enhance model reliability, trustworthiness, and resilience against adversarial manipulations. Future work could explore hybrid detection methods, real-time monitoring, and extending this framework to more complex architectures and datasets, further strengthening AI security in operational environments.

Bibliography

[1] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. arXiv preprint arXiv :1206.6389, 2013.

[2] Data poisoning : what it is and how it is being addressed by leading gen ai providers.