Wavelet Scattering Transform and Fourier Representation for Offline Detection of Malicious Clients in Federated Learning

Alessandro Licciardi* Davide Leo* Davide Carbone

arXiv:2506.09674v2 [cs.LG] 10 Oct 2025

***Abstract*—Federated Learning (FL) enables the training of machine learning models across decentralized clients while preserving data privacy. However, the presence of anomalous or corrupted clients—such as those with faulty sensors or non-representative data distributions—can significantly degrade model performance. Detecting such clients without accessing raw data remains a key challenge. We propose `Waffle` (Wavelet and Fourier representations for Federated Learning) a detection algorithm that labels malicious clients *before training*, using locally computed compressed representations derived from either the Wavelet Scattering Transform (WST) or the Fourier Transform. Both approaches provide low-dimensional, task-agnostic embeddings suitable for unsupervised client separation. A lightweight detector, trained on a distillated public dataset, performs the labeling with minimal communication and computational overhead. While both transforms enable effective detection, WST offers theoretical advantages, such as non-invertibility and stability to local deformations, that make it particularly well-suited to federated scenarios. Experiments on benchmark datasets show that our method improves detection accuracy and downstream classification performance compared to existing FL anomaly detection algorithms, validating its effectiveness as a pre-training alternative to online detection strategies. Source code for the paper is publicly available at https://github.com/davedleo/Waffle.**

***Index Terms*—Federated Learning, Wavelet Scattering Transform, Offline Detection, Anomaly Detection, Signal Processing**

I. Introduction

Federated Learning (FL) is a key paradigm for enabling decentralized intelligence in large-scale Internet of Things (IoT) systems, allowing numerous devices to train a shared model without exposing raw data [1], [2]. This approach is vital for privacy-sensitive applications in domains like smart cities, industrial automation, and connected healthcare [3], [4]. However, the success of FL in the IoT is threatened by two intertwined challenges: vast data heterogeneity from diverse devices and the system's vulnerability to malicious or faulty clients [5]–[7].

Consider a smart factory deployment where an Industrial IoT network monitors equipment health [8]. Some sensors may become miscalibrated due to environmental stress, suffer physical damage, or be deliberately compromised, injecting anomalous data into the system. Such data poisoning attacks can degrade the global model's performance, leading to incorrect predictions about machine failure. Detecting and isolating these compromised devices before they participate in training is crucial for maintaining network integrity and operational safety [9]. While defenses like robust aggregation [6], [10] can mitigate the impact of malicious model updates, they are often less effective against subtle data-level perturbations and typically assume a majority of clients are benign—a guarantee that may not hold in a widely distributed and physically accessible IoT network. Furthermore, online detectors that monitor clients during training can introduce significant communication and computational overhead, which is untenable for resource-constrained IoT devices, and may only flag threats after the damage has begun.

In this work, we propose `Waffle` (**Wa**velet and **F**ourier representations for **F**ederated **Le**arning), a lightweight, offline detector designed to identify and exclude clients with malicious data before FL training begins. Its privacy-preserving, pre-training detection is ideal for scalable IoT deployments, as it minimizes computational overhead on both the server and the edge devices. `Waffle` trains a classifier on stable spectral features—extracted via the Fourier Transform (FT) and Wavelet Scattering Transform (WST) [11]—which provide robust representations of client data distributions. The detection is performed using a model pre-trained on a public dataset, ensuring efficiency and privacy. Clients only need to compute low-dimensional spectral statistics and send a secure, non-invertible summary to the server for classification. Unlike existing methods, `Waffle` remains effective even when malicious clients form a large majority. Our experiments demonstrate its high efficacy in diverse settings, including under challenging non-Gaussian data attacks, and we showcase its versatility with a proof-of-concept on a Natural Language Processing (NLP) task.

The structure of the paper is the following: Section II defines the FL setting, the data attacks considered, and the spectral representations (FT and WST). Section III details `Waffle` 's training and detection. Theoretical guarantees are provided in Section IV, showing the benefits of removing malicious clients. Section V reports experimental results validating `Waffle` on benchmark datasets.

Related Work and Contributions.

Malicious Client Detection in FL Detection-based approaches classify clients as benign or malicious based on anomalies in their updates or data distribution [12]. `FLDetector` [13] identifies malicious clients by analyzing the consistency of their updates over time—benign updates follow

A.L. is affiliated to the Department of Mathematical Sciences, Politecnico di Torino, Turin, Italy, and to the Istituto Nazionale di Fisica Nucleare, Sezione di Torino, Turin, Italy

D.L. works for Tynk S.R.L., Via Viberti 4, Turin, Italy

D.C. is affiliated to Laboratoire de Physique de l'Ecole Normale Supérieure, ENS Université PSL; and to the CNRS, Sorbonne Université, Université de Paris, Paris, France

Correspondance to `alessandro.licciardi@polito.it`

* Denotes equal contributions

predictable patterns, while malicious ones are erratic. MuDHog [14] leverages historical update trajectories with model-agnostic meta-learning to detect temporal inconsistencies, though it assumes long-term client participation, which is unrealistic in cross-device settings. VAE [15] uses a variational autoencoder to model the benign update distribution and flags deviations, assuming malicious clients are rare and the VAE is well-trained. These methods rely on multi-round update access, limiting early-stage applicability, and focus on gradients or parameters, making them vulnerable to indirect attacks.

Robust Aggregation in FL. Robust aggregation methods aim to mitigate the influence of malicious clients without explicitly identifying them [6], [16]. KRUM [10] selects the most central update in ℓ_2 distance, but requires fewer than half of the clients to be malicious. TrimmedMean [6] discards extreme values per coordinate, improving robustness to outliers, though it overlooks dependencies across dimensions. FLTrust [7] uses a trusted server-side dataset to normalize and rescale client updates, enhancing robustness but breaking strict decentralization. Secure aggregation protocols like RFLPA [17] and RoFL [18] ensure client privacy via cryptographic techniques, but do not address adversarial robustness. These approaches, unlike detection-based ones, do not label clients, limiting their use when malicious participants must be explicitly excluded.

Spectral Analysis and Frequency-based Defenses. Spectral methods aim to identify or mitigate malicious behavior by analyzing updates in the frequency domain [19]–[21]. FreqFeD [22] applies the Discrete Fourier Transform to client updates, filtering high-frequency components assumed to contain adversarial noise, though this may remove relevant information under data heterogeneity. FedSSP [23] targets backdoor attacks by smoothing and pruning suspicious spectral patterns in model weights, but depends on specific architectures and requires access to full model parameters. Unlike these methods, our approach extracts frequency-based embeddings directly from client-side data before training, enabling model-agnostic detection.

Our *main contributions* are summarized as follows:

- We propose Waffle , a novel offline detector for identifying clients with data attacks, introducing the use of WST for anomaly detection in FL.
- We provide a theoretical framework motivating WST and FT as robust data representations, and mathematically demonstrate that removing malicious clients improves global model estimation.
- We present experiments on benchmark datasets showing that Waffle significantly improves model performance and robustness compared to training with contaminated data or using only robust aggregation.

II. Theoretical Framework

In this section, we introduce the mathematical framework that provides the foundation for our algorithm. Section II-A presents the Federated Learning (FL) setting and defines the class of attacks considered on clients' data. Section II-B introduces the WST and the Fourier Transform FT, recalling their basic properties that are relevant for anomaly detection.

A. Problem Formulation

Consider a standard FL scenario [1] with $K \in \mathbb{N}$ clients and a central server. Each client k possesses n_k data samples $(x_k^i, y_k^i)_{i=1}^{n_k} \sim \mathcal{D}_k$ supported in $\mathcal{X} \times \mathcal{Y}$. The objective of FL is to learn a shared global model θ that generalizes across all clients, by solving the following optimization problem:

$$\theta^* \in \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{k=1}^{K} n_k \mathcal{L}_k(\theta) \tag{1}$$

where Θ denotes the model's parameter space, $N = \sum_{k=1}^{K} n_k$ is the total number of data samples, and $\mathcal{L}_k$ represents the empirical loss function for client k with respect to its local data distribution $\mathcal{D}_k$. In each communication round $t \in \{1, \dots, T\}$, a subset of clients $\mathcal{P}_t$ is randomly selected to participate in training. Each participating client $k \in \mathcal{P}_t$ performs $S \in \mathbb{N}$ local iterations of a stochastic optimizer. Subsequently, clients send their updated parameters to the server, which aggregates these updates to derive a new global model.

A critical challenge in realistic FL deployments is the *non-i.i.d.* nature of client data, which can hinder the convergence and performance of the global model. In this work, we specifically address non-i.i.d. settings where the data distribution discrepancies are caused by malicious clients perturbing their original data samples. This differs from typical attack detection scenarios focusing on model poisoning during training.

a) Type of Attacks: We define two types of feature-level attacks that our algorithms aim to address: noisy and blur attackers. Examples of the effect of these attacks are displayed in Figure 1. This focus is motivated by the fact that noise and blur are common consequences of real-world faults [24], [25] –such as sensor degradation, miscalibration, or environmental interference – that can subtly compromise data quality and model performance without exhibiting overtly malicious behavior.

Definition 1. *Let $k \in [K]$ and $\sigma_k > 0$. Client k is a **noisy attacker** if its data samples are perturbed as $\tilde{x}_k^i = x_k^i + \sigma_k \epsilon_k^i$, where x_k^i is the clean sample, and, $(\epsilon_k^i)_{i=1}^{n_k}$ is a family of independent Wiener processes supported in $\mathcal{X}$.*

Let us observe that the severity of the attack is determined by the magnitude of σ_k. Smaller values of σ_k might represent natural noise inherent in data collection or random transformations, requiring careful consideration of what constitutes a 'malicious' level of perturbation.

Another feature-wise attack we formally define is the **blur attacker**. This attack is particularly relevant for image or signal data where x_k^i can be treated as a function over $\mathcal{X}$.

Definition 2. *Let $k \in [K]$ and $\beta_k > 0$. Client k is a **blurred attacker** if it provides samples perturbed according to a convolution operation:*

$$\tilde{x}_k^i = x_k^i \star \zeta_k = \int_{\mathcal{X}} x_k^i(u') \zeta_k(u - u') du' \quad i = 1, \dots, n_k \tag{2}$$

where $\star$ denotes the convolution operation. Typically, ζ_k is a smooth kernel, and the parameter β_k controls its spread or blur radius.

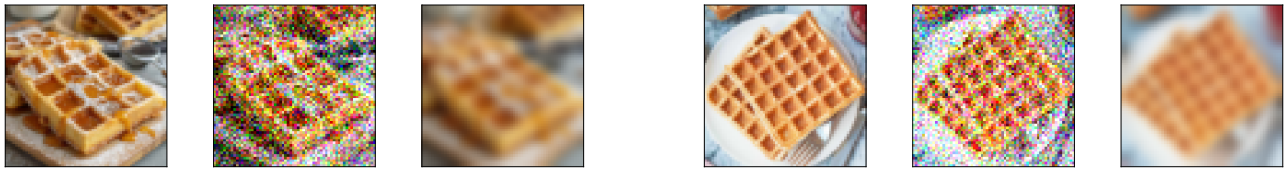


Fig. 1. Examples of attacked data. Two images downloaded from link1 and link2. For each image: *left*: clean client, *center*: noisy attack with magnitude $\sigma = 0.2$, *right*: blur attack with spread $\beta = 11$.

A common choice for the kernel ζ_k falls on Gaussian kernels, and the scalar β_k has a role of controlling the spread of the kernel. Similarly to noisy attacks, in blur attacks the magnitude of the perturbation is controlled by the parameter β_k, the larger it is the higher it perturbs the data.

B. Representation Operators: Wavelet Scattering Transform and Fourier Transform

In this section we recall the notion of a representation operator Φ, which maps a signal x (e.g., an image or a time-series) onto a transformed space. This transformation induces a metric $d(x, x') = \|\Phi[x] - \Phi[x']\|$ in the new space [26]. The core idea is that an effective representation operator Φ should possess properties instrumental for accurately detecting and differentiating between data samples. Specifically, for the purpose of identifying perturbed data, Φ should be able to separate distinct data characteristics while exhibiting robustness to common variations like slight translations or small, non-malicious perturbations. We propose two variants for the representation layer of our detection algorithm: one based on the Fourier Transform (FT) and the other on the Wavelet Scattering Transform (WST) [11], [26]. The Fourier Transform is by far the most widely used tool for spectral analysis in signal processing and data science due to its simplicity and interpretability. However, it has been surprisingly underutilized in the context of Federated Learning (FL). We therefore include it as an internal baseline in our study, allowing us to contrast its performance against the more structured and hierarchical Wavelet Scattering Transform.

a) Fourier Representation: We first formally define the Fourier Transform.

Definition 3. *Let $x \in L^1(\mathcal{X}, du)$, the **Fourier Transform** of x, denoted by $\mathcal{F}[x]$ is a complex valued function defined as*

$$\mathcal{F}[x](\omega) = \int_{\mathcal{X}} x(u) e^{-2\pi i (u \cdot \omega)} du \tag{3}$$

FT can be efficiently computed using the FFT algorithm [27]. Beyond its computational efficiency, the Fourier Transform offers several critical advantages for feature extraction, particularly in the context of analyzing data perturbations. As a linear operator ($\mathcal{F}[ax + bx'] = a\mathcal{F}[x] + b\mathcal{F}[x']$ for scalars a, b and integrable signals x, x'), the FT maps additive perturbations directly to additive components in the frequency domain. For instance, in the case of a *noisy attacker* where $\tilde{x} = x + \epsilon$, we have $\mathcal{F}[\tilde{x}] = \mathcal{F}[x] + \mathcal{F}[\epsilon]$. This linearity simplifies the analysis of such perturbations. Moreover, the convolution theorem [28] states that convolution in the spatial domain corresponds to point-wise multiplication in the frequency domain ($\mathcal{F}[x \star \delta] = \mathcal{F}[x] \cdot \mathcal{F}[\delta]$) . This property is highly advantageous for detecting *blur attacker* perturbations, which are defined as convolutions. By examining the frequency spectrum, different types of data manipulations, like blurring (attenuating high frequencies) or specific noise patterns, reveal distinct signatures. However, FT is an invertible operator: on one side it preserves all information present in the original signal, on the other hand it is possible to reconstruct the original data from the FT.

b) Wavelet Scattering Transform.: WST is a non-linear operator that, alternatively to Fourier based representation, has been designed to be stable to additive perturbations, locally translation invariant and to small continuous deformation. Moreover, the fact that WST is not invertible makes it particularly attractive for privacy-enhancing applications in FL, as reconstructing the original input data from the scattering coefficients is a challenging task. Following the construction in [11], [26] we define the WST and discuss its most relevant properties.

Let $\psi(u) \in L^2(\mathcal{X}, du)$ be a function referred to as the **mother wavelet**, and let $\{a^j\}_{j \in \mathbb{Z}}$ be a family of scale factors defined with respect to a fixed scalar $a > 1$. Let $r \in G$ denote a discrete rotation, where G is the group of discrete rotations acting on the domain $\mathcal{X}$. The j-th **wavelet function** is then defined as $\psi_j(u) = a^{-dj}\psi(a^{-j}r^{-1}u)$. For a fixed maximal depth $J \in \mathbb{Z}$, we define the set of admissible scale-rotation operators as $\Lambda_J = \{\lambda = a^j r : |\lambda| = a^j < 2^J\}$. In most implementations, Morlet wavelets are employed as the mother wavelet, and the scale factor is typically chosen as $a = 2^{1/Q}$ for some $Q \in \mathbb{N}$ [29].

To streamline notation, following [11], we introduce the **propagator operator**, which acts on a signal $x \in L^1(\mathcal{X})$ by cascading modulus and convolution operations. Given a path of scale-rotation operators $p = (\lambda_1, \lambda_2)$, the propagator applied to x is defined as:

$$U[p]x = \big|\, |x \star \psi_{\lambda_1}| \star \psi_{\lambda_2} \,\big| .$$

The definition of the WST naturally follows.

Definition 4. *Let $p = (\lambda_1, \ldots, \lambda_m) \subset \Lambda_J$ be a path of length*

m. *For any signal* $x \in L^1(\mathcal{X})$*, the WST along* p *is defined as:*

$$S_J[p]x = U[p]x \star \phi_J, \tag{4}$$

where ϕ_J *is a low-pass filter rescaled to recover low-frequency content.*

The WST representation shares structural similarities with convolutional neural networks (CNNs), with the key distinction that the wavelet filters are fixed rather than learned. The WST defines a norm with properties desirable for detection and classification. Notably, the operator is **non-expansive**: for any $x, x' \in L^2(\mathcal{X}, du)$, the following inequality holds:

$$\|S_J[p]x - S_J[p]x'\| \leq \|x - x'\|. \tag{5}$$

This implies that small, non-adversarial perturbations do not substantially affect the representation.

Additionally, WST is **translation invariant** in the limit: for a translated signal $x_c(u) = x(u - c)$ with $c \in \mathcal{X}$, we have

$$\lim_{J \to \infty} \|S_J[p]x - S_J[p]x_c\| = 0.$$

Finally, the WST is **Lipschitz continuous** with respect to small C^2-diffeomorphisms. That is, if a signal x undergoes a smooth deformation with small norm, the resulting change in the WST representation remains bounded.

III. Malicious Client Detector: Waffle

This section details the architecture and training of our server-side detector, `Waffle` (**Wa**velet and **F**ourier representations for **F**ederated **Le**arning), designed to identify clients contributing potentially harmful updates based on their data characteristics. `Waffle` is a parametric classification model, trained offline on a generated auxiliary dataset $\mathcal{D}^{\text{aux}}$ to distinguish between benign and malicious clients. It operates by analyzing aggregated, non-privacy-leaking spectral embeddings of client data distributions.

A. Offline Detector Training

The training of the `Waffle` detector is conducted entirely offline, prior to the federated learning process. This approach offers several advantages: it avoids interfering with live FL rounds, allows for controlled generation of diverse malicious scenarios, and ensures the detector is fully trained and ready when FL begins. Coherently with common practices in FL frameworks utilizing auxiliary data [30], the server has access to a representative auxiliary dataset $\mathcal{D}^{\text{aux}}$. Algorithm 1 summarizes the procedure.

The offline training proceeds over E epochs. In each epoch $e \in \{1, \ldots, E\}$, the server simulates a new FL round by generating a set of $\tilde{K}$ fictitious clients with synthetic data and associated ground-truth labels (benign or malicious). This dynamic generation of clients each epoch, similar to methods used for estimating client relationships [31], increases the diversity of simulated scenarios and helps prevent overfitting. Each training iteration within an epoch consists of two main steps: *data simulation/attack* and *feature extraction/labeling*.

a) Step 1: Data Simulation and Attack: For each sample $x \in \mathcal{D}^{\text{aux}}$, the server decides whether to simulate an attack on that sample or keep it clean. This decision is made by drawing from a Bernoulli distribution with probability $p = 1/2$ of being attacked. If selected for attack, the server randomly chooses between two types of data perturbations with equal probability: blur or noise. If a sample is selected for blurring, the server samples a blur severity parameter $\beta \sim \text{Unif}(\beta_0, \beta_1)$ and applies a blurring operation according to Definition 2. This simulates clients whose data might be of lower quality or intentionally blurred to impair model training or target specific vulnerabilities. If a sample is selected for adding noise, the server samples a noise variance $\sigma \sim \text{Unif}(\sigma_0, \sigma_1)$ and applies additive noise according to Definition 1. This simulates clients whose data might be corrupted by sensor noise or intentionally perturbed with adversarial noise patterns. After processing all samples in $\mathcal{D}^{\text{aux}}$ in this manner, the server possesses a modified dataset where each sample is either clean, blurred, or noisy, with the attack type and parameters recorded.

b) Step 2: Fictitious Client Creation and Feature Extraction: The modified dataset from Step 1 is then partitioned to create the data for $\tilde{K}$ fictitious clients. These clients are equally divided into two groups: $\tilde{K}/2$ benign and $\tilde{K}/2$ malicious. Clean data samples are assigned to benign clients, while attacked data samples (either blurred or noisy) are assigned to malicious clients. Let $\{x_k^i\}_{i=1}^{n_k}$ denote the data points assigned to the k-th fictitious client, where n_k is the number of samples for client k.

c) Principal Component Analysis: For each simulated client k, PCA [32] is applied to their local dataset $\{x_k^i\}_{i=1}^{n_k}$ to analyze the covariance structure and extract the top r principal components v_k^i with eigenvalues λ_k^i, capturing dominant directions of variance. A compact representation vector is defined as:

$$\hat{x}_k = \sum_{i=1}^{r} \alpha_k^i v_k^i, \quad \text{with} \quad \alpha_k^i = \frac{\lambda_k^i}{\sum_{j=1}^{r} \lambda_k^j} \tag{6}$$

This PCA-derived vector $\hat{x}_k$ summarizes the client data's intrinsic structure by weighting principal directions by their explained variance. The PCA step supports dimensionality reduction and noise filtering, extracting features sensitive to structural perturbations such as blur or noise. Notably, it is performed offline on simulated data at the server: in real FL deployments, clients neither share raw data nor PCA results. Instead, this offline PCA informs training, while clients transmit only the privacy-preserving spectral embedding φ_k, discussed next.

d) Spectral Embedding: Following this PCA step, the spectral representation φ_k is computed for each fictitious client k. This is achieved by applying a spectral operator Φ (either the WST or FT) to statistics derived from the client's data distribution, such as the PCA-derived representation vector $\hat{x}_k$ or the set of principal eigenvalues λ_k^i. Spectral transforms are particularly sensitive to frequency and texture information, making them effective at capturing the systematic changes introduced by attacks like blur and noise. The output $\varphi_k = |\Phi[\hat{x}_k]|$, where the modulus is taken element-wise, results in a fixed-size vector representation for each client. This φ_k is

designed to be an aggregate statistic that captures characteristics of the data distribution without revealing individual data points, making it suitable as a non-privacy-leaking feature for the detector in a live FL setting.

Finally, for each epoch, we obtain a dataset of client representations and their corresponding labels: $\{(\varphi_k, \mu_k)\}_{k=1}^{\tilde{K}}$, where $\mu_k \in \{\text{B (Benign)}, \text{A (Attacker)}\}$. The detector weights w are updated using a stochastic optimizer (e.g., SGD, Adam) to minimize a binary classification loss, such as Binary Cross-Entropy (BCE) [33], between the detector's prediction based on φ_k and the ground-truth label μ_k.

Algorithm 1 `Waffle` Offline Training

Require: Auxiliary dataset $\mathcal{D}^{\text{aux}}$, Number of epochs E, Number of fictitious clients $\tilde{K}$, Number of top PCs r, Spectral operator Φ, Learning rate η

Ensure: Trained detector weights w

Initialize detector weights w
for $e = 1 \dots E$ **do**
  // Simulate Data and Clients for Epoch e
  $\mathcal{D}_e^{\text{simulated}} \leftarrow \text{SimulateAttackedData}(\mathcal{D}^{\text{aux}})$ ▷ Applies random attacks to $\mathcal{D}^{\text{aux}}$
  $\{(\mathcal{D}_k, \mu_k)\}_{k=1}^{\tilde{K}} \leftarrow \text{PartitionData}(\mathcal{D}_e^{\text{simulated}}, \tilde{K})$ ▷ Creates $\tilde{K}$ clients with labels
  // Extract Features for Each Simulated Client
  Initialize epoch dataset $\mathcal{S}_e = \emptyset$ ▷ Stores (φ_k, μ_k) pairs
  for $k = 1 \dots \tilde{K}$ **do**
    $\{x_k^i\}_{i=1}^{n_k} \leftarrow \mathcal{D}_k$
    Compute PCA-derived representation $\hat{x}_k$ from $\{x_k^i\}$ ▷ Eq. (6)
    Compute spectral embedding $\varphi_k \leftarrow |\Phi[\hat{x}_k]|$ ▷ Apply FT or WST to $\hat{x}_k$
    Add (φ_k, μ_k) to $\mathcal{S}_e$
  end for
  // Update Detector Model
  $w \leftarrow \text{Opt}(\mathcal{L}_{\text{BCE}}(w; \mathcal{S}_e))$ ▷ Optimization step
end for
return w

B. Offline Detection and Filtering

Once the `Waffle` detector model w has been trained offline on the simulated auxiliary dataset $\mathcal{D}^{\text{aux}}$ and prior to the first FL communication round, each client $k \in \{1, \dots, K\}$ in the federation processes its local training data $\{x_k^i\}_{i=1}^{n_k}$ *privately* on their device. This processing involves a sequence of steps performed locally. First, each client computes the PCA of their local training samples to derive the representation vector $\hat{x}_k$, as defined in Equation (6).Then, each client computes its spectral embedding $\varphi_k = \Phi[\hat{x}_k]$, by applying the spectral operator Φ (WST or FT).

After completing these local computations and obtaining φ_k, each client k securely transmits only this resulting spectral embedding vector to the server. The server, upon receiving φ_k from each participating client, inputs it into the pre-trained `Waffle` detector w. Clients that are classified as malicious by the detector are then excluded from participating in the federated training process for the global model θ. This preemptive filtering step enhances the stability and reliability of the global model training process, leading to potentially faster and more robust convergence by ensuring that aggregation occurs over updates from predominantly benign sources.

Moreover, due to its modular nature, `Waffle` operates as an initial defense layer. The set of clients validated as benign by `Waffle` can proceed with any federated learning aggregation methods, allowing `Waffle` to be easily combined with other online robust aggregation techniques to further strengthen the overall defense strategy.

IV. Theoretical Guarantees

In this section, we establish a theoretical foundation for our proposed algorithm, which we refer to as `Waffle` . Our primary focus is to demonstrate the benefits of removing adversarial clients in FL scenarios. We show that by filtering out malicious updates, `Waffle` provides a more accurate estimate of the true global model compared to standard `FedAvg` [1], which is susceptible to adversarial poisoning. We provide general error bounds with detailed proofs presented in Appendix A.

Let $\mathcal{B} \subset \{1, \dots, K\}$ denote the set of benign clients and $\mathcal{M} \subset \{1, \dots, K\}$ the set of malicious clients in a federated system with K total clients. We assume these sets are disjoint and their union covers all clients, i.e., $\mathcal{B} \cap \mathcal{M} = \emptyset$ and $\mathcal{B} \cup \mathcal{M} = \{1, \dots, K\}$. To model the heterogeneity and potential adversarial influence in client updates, we adopt the following statistical framework:

Assumption 1. *For each benign client $k \in \mathcal{B}$, the local model update θ_k is an independent random variable drawn from a distribution $\rho_k(\bar{\theta}^b, \sigma^b)$. This distribution is centered around a common benign mean $\bar{\theta}^b$ with variance $(\sigma^b)^2$, i.e., $\mathbb{E}[\theta_k] = \bar{\theta}^b$ and $\mathbb{V}ar[\theta_k] = (\sigma^b)^2$. Similarly, for malicious clients $k \in \mathcal{M}$, the local updates θ_k are independent random variables drawn from $\rho_k(\bar{\theta}^m, \sigma^m)$ with $\mathbb{E}[\theta_k] = \bar{\theta}^m$ and $\mathbb{V}ar[\theta_k] = (\sigma^m)^2$.*

Assumption 2. *We posit that malicious clients exhibit significantly higher update variance compared to benign clients, reflecting a diverse range of attack strategies and the potential for large, destabilizing updates. Formally, we assume $\sigma^m \gg \sigma^b$.*

The standard federated averaging estimator is defined as a weighted average of client updates: $\theta_{avg} = 1/K \sum_{k=1}^{K} \theta_k$. Our objective is to obtain an estimator that is unbiased with respect to the benign client distribution, meaning $\mathbb{E}[\theta_{avg}] = \bar{\theta}^b$. We demonstrate that removing malicious clients is crucial for achieving this goal. We analyze two scenarios: one where the benign and malicious updates have different means (Lemma 1) and one where they share the same mean but differ in variance (Lemma 2).

Lemma 1. *If the benign and malicious client updates have different mean parameter values, i.e., $\bar{\theta}^m \neq \bar{\theta}^b$, then the standard federated averaging estimator θ_{avg} is a **biased estimator** of $\bar{\theta}^b$, meaning $\mathbb{E}[\theta_{avg}] \neq \bar{\theta}^b$.*

Lemma 2. *Let $\theta^{\mathcal{B}}_{avg} = \frac{1}{|\mathcal{B}|}\sum_{k\in\mathcal{B}}\theta_k$ be the federated averaging estimator computed using only benign client updates. Under Assumption 2, if $(\sigma^m)^2 > \left(2 + \frac{|\mathcal{M}|}{|\mathcal{B}|}\right)(\sigma^b)^2$, then the variance of the standard federated averaging estimator is higher than that of our estimator: $\mathbb{V}ar[\theta_{avg}] \geq \mathbb{V}ar[\theta^{\mathcal{B}}_{avg}]$.*

Lemmas 1 and 2 provide the foundation for the following proposition, which formally establishes the advantage of removing malicious clients from the federated aggregation process.

Proposition 1. *Under Assumptions 1 and 2, removing malicious clients (those in $\mathcal{M}$) from the federation yields a superior estimator of the global model. Specifically, the resulting estimator is unbiased (in the sense of Lemma 1) and exhibits a reduced variance (as shown in Lemma 2), leading to improved model accuracy and robustness.*

V. Experiments

In this section, we present experimental results on widely used federated learning benchmark datasets [34]–[36], comparing the performance of `Waffle` in its two variants—one using the WST representation and the other using FT—with established baselines from the Byzantine-resilient FL literature. Details on implementation settings, datasets, and models are provided in Appendix B.

Section V-A evaluates the detection performance of the two variants of `Waffle` , highlighting the differences between the WST and FT representations. In Section V-B, we compare `Waffle` against standard Byzantine-resilient FL baselines, including `FedAvg` [1], `Krum` and `mKrum` [10], `GeoMed` [37], and `TrimmedMean` [6]. Additionally, we demonstrate that `Waffle` can be applied on top of any aggregation algorithm, improving their performance. Further experiments, comparisons and code release details are reported in Appendix B, and the metrics used for evaluation—both for detection and classification—are detailed in Appendix C.

A. `Waffle` : WST vs Fourier

We compare the detection performance of `Waffle` to assess the differences between the WST and FT representations. As illustrated in Figure 2, both representations yield a clear separation between benign and malicious clients. The visualizations—obtained via two-dimensional PCA embeddings—show that the method effectively distinguishes between the different attacker groups and benign clients, regardless of the chosen representation. However, as shown in Table I, the quantitative results at the client level differ between the two variants. We report standard detection metrics: precision, F1 score, recall, and accuracy [38], setting 40% and 90% of attackers. The WST variant consistently achieves higher precision and F1 scores, while the FT variant tends to yield higher recall. In the context of malicious client detection, higher recall is often desirable, as it reduces the likelihood of overlooking faulty clients. Table I highlights the robustness of `Waffle` : unlike most Byzantine-resilient FL methods, it maintains strong predictive performance even when the vast majority of clients are malicious. Notably, in the extreme case with 90% adversarial clients, `Waffle` with WST achieves 100% precision across all datasets.

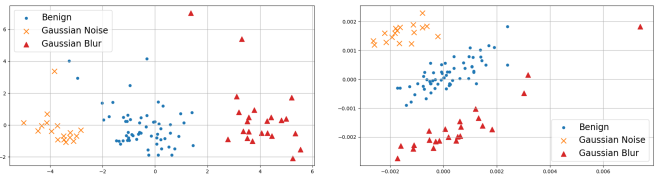


Fig. 2. Client distributions of the φ_k for Cifar10 dataset with $K = 100$ clients on a 2-dimensional space, for `Waffle` + FT (left), and `Waffle` + WST (right). There is a total of 60 benign clients (dots), and 40 attackers: 20 noisy (crosses) and 20 blurred (triangles). Both methods provide a noticeable separation between the clients.

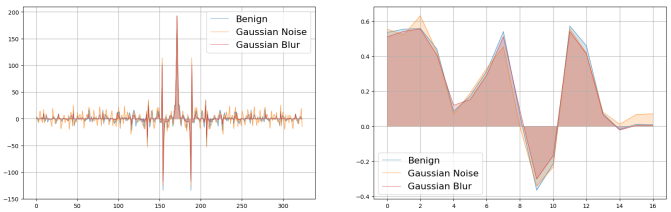


Fig. 3. Embeddings φ_k produced by `Waffle` for three clients (blur attacker, noise attacker, and benign) on CIFAR-10. The left panel shows the embeddings obtained using FT, while the right panel shows those obtained using WST.

B. Comparison with Baselines and Orthogonality of `Waffle`

In this section, we compare `Waffle` with established Byzantine-resilient FL methods, highlighting its advantages in two complementary settings: (1) we evaluate the impact of applying the two `Waffle` variants to `FedAvg`, compared to using different aggregation rules without detection; and (2) we assess the effect of applying `Waffle` on top of robust aggregation algorithms. As shown in Table II, the WST variant of `Waffle` combined with `FedAvg` consistently outperforms all baselines across all datasets. Furthermore, `Waffle` improves the performance of each aggregation method it is applied to, demonstrating its orthogonality to the choice of aggregator. These results indicate that `Waffle` is effective in identifying and removing malicious clients without compromising benign contributions. In contrast, the FT variant exhibits more variable performance, further confirming the suitability of WST representations for this detection task. For reference, we also report the test accuracy of `FedAvg` trained on a clean federation (i.e., without malicious clients, corresponding to $\theta^{\mathcal{B}}_{avg}$ in the notation of Lemma 2): FashionMNIST 75.08%, CIFAR-10 50.24%, CIFAR-100 17.72%. These values demonstrate that `Waffle` enables recovery of near-optimal performance, effectively neutralizing the impact of adversarial clients.

C. Comparison with Anomaly Detection Baselines

To provide a comprehensive comparison, we also benchmark our approach against other recent detection methods from the literature, namely `FLDetector` [13]and `VAEDetector` [15]. These methods operate as online techniques, analyzing model updates across multiple training rounds to identify malicious behavior. We evaluate their performance in a scenario with 60% benign clients under the same random block attack. The accuracy achieved by `FedAvg` when integrated with

TABLE I

CLIENT DETECTION. COMPARISON BETWEEN VARIANTS OF WAFFLE USING WST AND FT REPRESENTATIONS, UNDER TWO ATTACK SCENARIOS (40% TOP, 90% BOTTOM). METRICS (F1 SCORE, PRECISION, RECALL, ACCURACY [38]) REFER TO THE DETECTION OF MALICIOUS CLIENTS.

	Method	FashionMNIST				CIFAR-10				CIFAR-100			
		F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.	Acc.
40%	Waffle - FT	65.1 ±3.1	59.9±3.1	**69.1** ±3.1	**69.2**±3.1	80.2±2.6	69.1±2.6	**96.1**±2.6	67.0±2.6	55.1 ±3.2	40.5±2.6	**89.7**±2.6	44.1±2.6
	Waffle - WST	**72.7** ±1.1	**96.3**±1.1	58.2±1.1	**82.4**±2.6	**95.2**±1.0	**97.6**±1.0	92.9±1.0	**96.1** ±1.0	**83.0** ±1.2	**93.1**±1.2	75.1±1.2	**87.0**±1.2
90%	Waffle - FT	**80.9**±2.6	94.2 ±2.6	**70.7**±2.6	**71.2**±2.6	**93.3**±1.6	89.2±1.6	**95.7**±1.6	**86.2**±1.6	89.0 ±1.6	88.2±1.6	**88.4**±1.6	**81.1**±1.6
	Waffle - WST	65.6 ±0.2	**100.0**±0.0	49.1±0.2	54.0±0.2	91.1±0.5	**100.0**±0.0	83.8 ±0.5	87.0±0.5	**88.1**±0.3	**100.0**±0.0	68.3±0.3	72.2±0.3

TABLE II

COMPARISON BETWEEN BASELINES FOR DETECTING MALICIOUS CLIENTS AND WAFFLE (WITH BOTH WST AND FT) WITH 2σ ERROR BARS. WE CONSIDER AS UPPER-BOUND FOR ALL METHODS FEDAVG TRAINED ON THE WHOLE BENIGN FEDERATION WITHOUT MALICIOUS CLIENTS — FASHIONMNIST $\mathbf{75.5 \pm 1.7}$, CIFAR-10 $\mathbf{50.3 \pm 0.5}$, AND CIFAR-100 $\mathbf{17.0 \pm 1.3}$.

Dataset	Setting	FedAvg	Krum	mKrum	GeoMed	TrimmedMean
FashionMNIST	w/o detector	73.7 ± 1.3	73.8 ± 1.1	72.5 ± 4.0	73.4 ± 1.7	74.6 ± 0.4
	Waffle - WST	**74.9 ± 1.9**	70.2 ± 0.4	74.2 ± 0.9	74.6 ± 1.6	74.7 ± 1.8
	Waffle - FT	73.8 ± 1.1	71.4 ± 2.0	74.6 ± 0.4	74.7 ± 1.0	**74.9 ± 0.5**
CIFAR-10	w/o detector	48.7 ± 1.3	44.8 ± 2.2	46.2 ± 5.9	48.3 ± 0.5	48.1 ± 0.4
	Waffle - WST	**49.6 ± 0.3**	46.2 ± 0.6	49.5 ± 0.6	49.0 ± 1.4	49.5 ± 0.8
	Waffle - FT	47.1 ± 0.4	43.8 ± 1.8	46.7 ± 1.3	47.2 ± 0.3	46.8 ± 1.1
CIFAR-100	w/o detector	16.4 ± 0.1	10.1 ± 0.8	14.6 ± 0.7	16.4 ± 0.7	16.5 ± 1.1
	Waffle - WST	**16.5 ± 1.0**	8.8 ± 2.2	14.5 ± 0.7	16.3 ± 0.3	16.2 ± 0.5
	Waffle - FT	11.6 ± 0.2	7.6 ± 0.6	10.6 ± 0.7	12.1 ± 0.3	10.6 ± 0.5

these detectors is reported in Table III. The results show that under this challenging, non-Gaussian attack scenario, these benchmarks were unable to reliably detect the attacks, leading to a significant drop in performance compared to our method.

TABLE III

PERFORMANCE COMPARISON WITH OTHER DETECTION METHODS UNDER A RANDOM BLOCK ATTACK WITH 40% MALICIOUS CLIENTS.

Dataset	FedAvg	Waffle -WST	Waffle -FT	FLDetector	VAEDetector
FashionMNIST	73.7 ± 1.3	$\mathbf{74.9 \pm 1.9}$	73.8 ± 1.1	71.4 ± 0.9	73.1 ± 0.8
CIFAR-10	48.7 ± 1.3	$\mathbf{49.6 \pm 0.3}$	47.1 ± 0.4	45.4 ± 0.6	47.0 ± 0.5
CIFAR-100	16.4 ± 0.1	$\mathbf{16.5 \pm 1.0}$	11.6 ± 0.2	16.2 ± 0.1	15.5 ± 0.7

D. Waffle with Non-Gaussian Attacks

In the theoretical section of the paper, we formalize two scenarios of attacks, i.e., clients providing images that have been perturbed with additive Gaussian noise – *noisy clients* – and those that have blurred data, where the blur mathematically is a convolution with a Gaussian kernel. In this section, we provide an experimental evaluation of our method in the presence of non-Gaussian attacks. The experimental framework that we analyze consists of an attack in which a random subset of pixels of each client's data is perturbed; in this case, 50% are substituted with black pixels.

However, the Waffle framework is not limited to detecting these types of attacks. To validate its robustness against more complex, non-Gaussian structural attacks, we conducted further experiments. In this new scenario, malicious clients apply a random dropout attack on part of the image, where 50% of the image pixels, grouped into small random blocks, are set to zero. This introduces sharp, non-Gaussian artifacts that are structurally different from simple noise. Waffle-WST obtained an almost perfect detection performance, as it is summarized here.

We report in Table IV the results of the random block attack across CIFAR-10, CIFAR-100, and Fashion-MNIST, including standard robust aggregation baselines.

As expected from our theoretical results, if the detector is perfect, we reach a performance of the federated training that is comparable with the situation without malicious clients. The WST variant is particularly effective, as it is designed to capture local structural information and textures. The random dropout attack fundamentally disrupts these local patterns, creating a strong and detectable signal for our framework. The benefit of Waffle, especially the WST variant, is particularly prominent on color images (CIFAR-10/100), where the attack disrupts chromatic and texture patterns that WST is well-suited to detect. In contrast, Fashion-MNIST consists of grayscale images, where the attack is more subtle and less disruptive to local statistics. Nonetheless, Waffle-WST still achieves performance very close to the clean-case baseline. This displays that Waffle is a robust solution capable of identifying a broader class of feature-level data integrity attacks beyond simple Gaussian perturbations.

E. Waffle in NLP tasks

As a proof of concept for tasks beyond computer vision, we extend our evaluation to a Natural Language Processing (NLP) task. We do not compare against other baselines here, as this is intended to demonstrate the versatility of our framework. For this experiment, we implemented a composite Shift-and-Noise Attack on the 50-dimensional GloVe embeddings [39] in its most recent version [40] for 40% of the 100 clients in the federation. The attack consists of two components: (1) applying

TABLE IV

PERFORMANCE UNDER RANDOM BLOCK ATTACK. WE REPORT MEAN TEST ACCURACY AND 2-SIGMA ERROR BARS OVER MULTIPLE RUNS. WE CONSIDER AS UPPER-BOUND FOR ALL METHODS FEDAVG TRAINED ON THE WHOLE BENIGN FEDERATION WITHOUT MALICIOUS CLIENTS — FASHIONMNIST **75.5** $\pm$ **1.7**, CIFAR-10 **50.3** $\pm$ **0.5**, AND CIFAR-100 **17.0** $\pm$ **1.3**.

Dataset	FedAvg	Krum	MultiKrum	TrimmedMean	GeoMed	Waffle-WST	Waffle-FT
CIFAR-10	48.7 ± 1.3	44.5 ± 0.2	47.8 ± 0.2	47.9 ± 0.3	48.2 ± 0.2	**49.8** $\pm$ **0.2**	48.5 ± 0.2
CIFAR-100	16.4 ± 0.1	9.4 ± 0.1	15.3 ± 0.1	16.2 ± 0.1	15.1 ± 0.1	**16.9** $\pm$ **0.1**	16.3 ± 0.1
Fashion-MNIST	73.7 ± 1.3	74.3 ± 0.3	71.7 ± 0.3	75.0 ± 0.2	71.4 ± 0.3	**75.4** $\pm$ **0.5**	75.3 ± 0.6

random permutations to the embedding vectors and (2) adding Gaussian noise. Our Waffle-WST method demonstrated strong detection capabilities against this attack, achieving an F1-score of 0.73 and, notably, a perfect precision of 1.0, ensuring no honest clients were penalized. This successful detection directly translated to a significant performance recovery in the global model. As shown in Table V, the Waffle detector is able to raise the final test accuracy from a compromised 38.53% (without our detector) to 42.71%. This result brings the model's performance remarkably close to the ideal, attack-free scenario of 44.81%. These outcomes are entirely consistent with the extensive experiments in our main paper, and we will provide a full and detailed analysis of these NLP results in the camera-ready submission.

TABLE V

MODEL ACCURACY ON THE NLP TASK UNDER A COMPOSITE SHIFT-AND-NOISE ATTACK. THE TASK IS CLASSIFICATION OF SENTIMENTS, THEREFORE THE EVALUATION METRIC IS STILL ACCURACY.

Scenario and method	**Test Accuracy (%)**
FedAvg w/o malicious clients (No Attack)	44.81 ± 2.1
FedAvg w/ Waffle -WST (Under Attack)	42.71 ± 1.9
FedAvg w/ Waffle -FT (Under Attack)	41.91 ± 1.6
FedAvg w/o Detector (Under Attack)	38.53 ± 2.0

VI. CONCLUSION

We propose Waffle , a novel offline algorithm to detect malicious client data in FL before training commences. Exploiting stable spectral features extracted via the WST and FT, our method enables robust anomaly detection from private, low-dimensional client-side summaries built on publicly distilled data. By filtering out compromised clients prior to the aggregation process, Waffle significantly improves convergence speed, final model accuracy, and robustness to data contamination. Our benchmarks show it achieves near-perfect precision, even in extreme scenarios with up to 90% malicious clients, outperforming strategies that rely solely on robust aggregation.

A key advantage of Waffle is its role as a proactive and complementary security layer. By specializing in the detection of data-level attacks, it acts as an essential first line of defense, sanitizing the client pool before resource-intensive training begins. This model-agnostic approach is not intended to replace in-training defenses but rather to fortify them. It can be seamlessly integrated with existing FL defenses, such as robust aggregation mechanisms that target model-level threats, to create a more comprehensive, multi-layered security architecture against a wider spectrum of attacks.

This early-detection mechanism also yields substantial practical benefits by reducing training time, communication overhead, and energy consumption—factors that are crucial in large-scale and resource-constrained deployments, like IoT. By enhancing the robustness, trustworthiness, and efficiency of the training pipeline, our method helps pave the way for secure FL deployments in sensitive domains like connected healthcare, autonomous systems, and smart infrastructure.

Future work will focus on extending Waffle to defend against more sophisticated threats, including backdoor attacks, model poisoning, and sybil-based infiltration. In parallel, we plan to adapt the approach to support wider neural architectures capable of handling more complex and high-dimensional datasets, such as CIFAR-100 or even ImageNet-scale benchmarks. These directions aim to broaden the applicability and impact of Waffle in advancing secure and efficient decentralized machine learning.

ACKNOWLEDGEMENTS

A.L. and D.C. worked under the auspices of Italian National Group of Mathematical Physics (GNFM) of INdAM. A.L. was supported by the Project Piano Nazionale di Ripresa e Resilienza - Next Generation EU (PNRR-NGEU) from Italian Ministry of University and Research (MUR) under Grant DM 117/2023. D.C. expresses their gratitude to Marylou Gabrié for the support.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[2] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[3] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.

[4] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," in *Federated learning: privacy and incentive*, pp. 240–254, Springer, 2020.

[5] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[6] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International conference on machine learning*, pp. 5650–5659, Pmlr, 2018.

[7] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[8] M. Dzaferagic, N. Marchetti, and I. Macaluso, "Fault detection and classification in industrial iot in case of missing sensor data," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8892–8900, 2021.

[9] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, vol. 195, pp. 346–361, 2022.

[10] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[11] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[12] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[13] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2545–2555, 2022.

[14] A. Gupta, T. Luo, M. V. Ngo, and S. K. Das, "Long-short history of gradients is all you need: Detecting malicious and unreliable clients in federated learning," in *European Symposium on Research in Computer Security*, pp. 445–465, Springer, 2022.

[15] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[16] R. Guerraoui, S. Rouault, *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International conference on machine learning*, pp. 3521–3530, PMLR, 2018.

[17] P. Mai, R. Yan, and Y. Pang, "Rflpa: A robust federated learning framework against poisoning attacks with secure aggregation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 104329–104356, 2024.

[18] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 453–476, IEEE, 2023.

[19] Z. Wang, C. Pei, M. Ma, X. Wang, Z. Li, D. Pei, S. Rajmohan, D. Zhang, Q. Lin, H. Zhang, *et al.*, "Revisiting vae for unsupervised time series anomaly detection: A frequency perspective," in *Proceedings of the ACM Web Conference 2024*, pp. 3096–3105, 2024.

[20] C.-h. Chan and G. K. Pang, "Fabric defect detection by fourier analysis," *IEEE transactions on Industry Applications*, vol. 36, no. 5, pp. 1267–1276, 2000.

[21] R. Tao, X. Zhao, W. Li, H.-C. Li, and Q. Du, "Hyperspectral anomaly detection by fractional fourier entropy," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 12, pp. 4920–4929, 2019.

[22] H. Fereidooni, A. Pegoraro, P. Rieger, A. Dmitrienko, and A.-R. Sadeghi, "Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning," *arXiv preprint arXiv:2312.04432*, 2023.

[23] Y. Chen and Z. Tan, "Fedssp: Federated graph learning with spectral knowledge and personalized preference," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[24] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 3, pp. 1–39, 2010.

[25] Y. Peng, Z. Tang, G. Zhao, G. Cao, and C. Wu, "Motion blur removal for uav-based wind turbine blade images using synthetic datasets," *Remote Sensing*, vol. 14, no. 1, p. 87, 2021.

[26] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[27] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[28] R. N. Bracewell, "The fourier transform," *Scientific American*, vol. 260, no. 6, pp. 86–95, 1989.

[29] M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, J. Andén, E. Belilovsky, *et al.*, "Kymatio: Scattering transforms in python," *Journal of Machine Learning Research*, vol. 21, no. 60, pp. 1–6, 2020.

[30] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.

[31] W. Bao, H. Wang, J. Wu, and J. He, "Optimizing the collaboration structure in cross-silo federated learning," in *International Conference on Machine Learning*, pp. 1718–1736, PMLR, 2023.

[32] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[33] U. Ruby, V. Yendapalli, *et al.*, "Binary cross entropy with deep learning technique for image classification," *Int. J. Adv. Trends Comput. Sci. Eng*, vol. 9, no. 10, 2020.

[34] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[35] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[36] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)."

[37] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[38] J. Lever, "Classification evaluation: It is important to understand both what a classification metric expresses and what it hides," *Nature methods*, vol. 13, no. 8, pp. 603–605, 2016.

[39] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[40] R. Carlson, J. Bauer, and C. D. Manning, "A new pair of gloves," *arXiv preprint arXiv:2507.18103*, 2025.

[41] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, vol. 60, pp. 53–60, Perth, Australia, 1995.

[42] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, J. AndÃ©n, E. Belilovsky, J. Bruna, V. Lostanlen, M. Chaudhary, M. J. Hirn, E. Oyallon, S. Zhang, C. Cella, and M. Eickenberg, "Kymatio: Scattering transforms in python," *Journal of Machine Learning Research*, vol. 21, no. 60, pp. 1–6, 2020.

[44] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[45] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

APPENDIX

A. Theoretical Guarantes

In this Appendix, we present comprehensive proofs and assumptions pertaining to the results delineated in Section IV. Specifically, we focus on demonstrating that in the context of the federated averaging estimator, the exclusion of malicious clients prior to the training process facilitates the attainment of an unbiased estimator of the true mean θ^b, consequently resulting in a less noisy estimate, as stated in Proposition 1. This outcome is supported through the introduction of two lemmas, namely Lemmas 1 and 2. Without loss of generality, we propose two general assumptions, which reflect the distinct behaviors of benign and malicious clients concerning model distribution. Specifically, we denote the set of benign clients as $\mathcal{B} \subset \{1, \dots, K\}$ and the set of malicious clients as $\mathcal{M} \subset \{1, \dots, K\}$, within a federated system comprising K clients in total. We assume these sets are mutually exclusive and collectively exhaustive, meaning $\mathcal{B} \cap \mathcal{M} = \emptyset$ and $\mathcal{B} \cup \mathcal{M} = \{1, \dots, K\}$. To adequately address the heterogeneity and the potential adversarial impact on client updates, we employ the following statistical framework.

Assumption A1. *For each benign client $k \in \mathcal{B}$, the local model update θ_k is an independent random variable drawn from a distribution $\rho_k(\bar{\theta}^b, \sigma^b)$. This distribution is centered around a common benign mean $\bar{\theta}^b$ with variance $(\sigma^b)^2$, i.e., $\mathbb{E}[\theta_k] = \bar{\theta}^b$ and $\mathbb{V}ar[\theta_k] = (\sigma^b)^2$. Similarly, for malicious clients $k \in \mathcal{M}$, the local updates θ_k are independent random variables drawn from $\rho_k(\bar{\theta}^m, \sigma^m)$ with $\mathbb{E}[\theta_k] = \bar{\theta}^m$ and $\mathbb{V}ar[\theta_k] = (\sigma^m)^2$.*

Assumption A2. *We posit that malicious clients exhibit significantly higher update variance compared to benign clients, reflecting a diverse range of attack strategies and the potential for large, destabilizing updates. Formally, we assume that there exists $C > 0$ such that $(\sigma^m)^2 > C(\sigma^b)^2 > \left(2 + \frac{|\mathcal{M}|}{|\mathcal{B}|}\right)(\sigma^b)^2$.*

Assumption A1 characterizes the statistical distributions corresponding to the two distinct groups of clients. In contrast, Assumption A2 provides that the variance associated with the malicious models is significantly greater than that of the benign client updates. The standard federated averaging estimator is defined as a weighted average of client updates, i.e.

$$\theta_{avg} = \frac{1}{K} \sum_{k=1}^{K} \theta_k \quad . \tag{7}$$

Our objective is to obtain an estimator that is unbiased with respect to the benign client distribution, meaning $\mathbb{E}[\theta_{avg}] = \bar{\theta}^b$. We demonstrate that removing malicious clients is crucial for achieving this goal. We analyze two scenarios: one where the benign and malicious updates have different means (Lemma A1) and one where they share the same mean but differ in variance (Lemma A2).

Lemma A1. *If the benign and malicious client updates have different mean parameter values, i.e., $\bar{\theta}^m \neq \bar{\theta}^b$, then the standard federated averaging estimator θ_{avg} is a **biased estimator** of $\bar{\theta}^b$, meaning $\mathbb{E}[\theta_{avg}] \neq \bar{\theta}^b$.*

Proof. Let us first recall that a random variable $\hat{X}$ is an unbiased estimator of μ, if its expectation equals the parameter that we aim to estimate, i.e. if $\mathbb{E}[\hat{X}] = \mu$. In case $\mathbb{E}[\hat{X}] \neq \mu$, we say that $\hat{X}$ is a biased estimator of μ.

If we compute the expectation of the estimator θ_{avg}, defined in Equation 7, using the fact that malicious and benign client $\{\mathcal{B}, \mathcal{M}\}$ form a partition of $\{1, \dots, K\}$, we get

$$\mathbb{E}[\theta_{avg}] = \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\theta_k\right] = \mathbb{E}\left[\frac{1}{K}\left(\sum_{k\in\mathcal{B}}\theta_k + \sum_{k\in\mathcal{M}}\theta_k\right)\right] \quad . \tag{8}$$

Let us denote with $M = |\mathcal{M}|$ and $B = |\mathcal{B}|$ the number of malicious and benign clients in the federation, by exploiting linearity of the expectation operator, we obtain

$$\begin{aligned}\mathbb{E}[\theta_{avg}] &= \frac{1}{K}\left(\sum_{k\in\mathcal{B}}\mathbb{E}[\theta_k] + \sum_{k\in\mathcal{M}}\mathbb{E}[\theta_k]\right) = \frac{B\bar{\theta}^b + M\bar{\theta}^m}{K} \\ &= \frac{B\bar{\theta}^b + M\bar{\theta}^b - M\bar{\theta}^b + M\bar{\theta}^m}{K} \\ &= \bar{\theta}^b + \frac{M}{K}(\bar{\theta}^m - \bar{\theta}^b) \neq \bar{\theta}^b\end{aligned} \tag{9}$$

where $\bar{\theta}^b$ and $\bar{\theta}^m$ denote the expectation of the model updates for benign and malicious clients, respectively. Since we obtained that $\mathbb{E}[\theta_{avg}] \neq \bar{\theta}^b$, we conclude that the estimator is biased. □

Furthermore, we observe that the drift in the estimate away from the benign model is controlled by the ratio of malicious clients M with respect to the number of total clients K.

Lemma A2. *Let*

$$\theta^{\mathcal{B}}_{avg} = \frac{1}{|\mathcal{B}|}\sum_{k\in\mathcal{B}}\theta_k \tag{10}$$

be the federated averaging estimator computed using only benign client updates. Under Assumption 2 the variance of the standard federated averaging estimator is higher than that of our estimator: $\mathbb{V}ar[\theta_{avg}] \geq \mathbb{V}ar[\theta^{\mathcal{B}}_{avg}]$.

Proof. First, we compute the variance for the two estimators θ_{avg} and $\theta^{\mathcal{B}}_{avg}$ exploiting the indipendece between model distributions, posit in Assumption A1. In particular

$$\begin{aligned}\mathbb{V}ar[\theta_{avg}] &= \mathbb{V}ar\left[\frac{1}{K}\sum_{k=1}^{K}\theta_k\right] \\ &= \frac{1}{K^2}\left(\sum_{k\in\mathcal{B}}\mathbb{V}ar[\theta_k] + \sum_{k\in\mathcal{M}}\mathbb{V}ar[\theta_k]\right) \\ &= \frac{B(\sigma^b)^2 + M(\sigma^m)^2}{K^2} \quad .\end{aligned} \tag{11}$$

Similarly, we get that

$$\mathbb{V}ar[\theta^{\mathcal{B}}_{avg}] = \frac{(\sigma^b)^2}{B} \quad . \tag{12}$$

If we consider the difference between the variances $\mathbb{V}ar[\theta_{avg}]$ and $\mathbb{V}ar[\theta^{\mathcal{B}}_{avg}]$, and we impose that this quantity is positive qwe obtain the following inequality

$$\mathbb{V}ar[\theta_{avg}] - \mathbb{V}ar[\theta^{\mathcal{B}}_{avg}] = \frac{B(\sigma^b)^2 + M(\sigma^m)^2}{K^2} - \frac{(\sigma^b)^2}{B} > 0 \quad . \tag{13}$$

Recalling that $K = B + M$, we get

$$\begin{aligned} &\frac{B^2(\sigma^b)^2 + MB(\sigma^m)^2 - (B+M)^2(\sigma^b)^2}{B(B+M)^2} > 0 \\ &\iff MB(\sigma^m)^2 - M(2B+M)(\sigma^b)^2 > 0 \end{aligned} \tag{14}$$

that is, since $M > 0$,

$$(\sigma^m)^2 > \frac{1}{B}(2B+M)(\sigma^b)^2 \iff (\sigma^m)^2 > \left(2 + \frac{M}{B}\right)(\sigma^b)^2 \tag{15}$$

which together with Assumption A1 concludes the proof. □

Lemma A2 provides a definitive bound on the variance of the model, thereby resolving the question *how much larger should the variance of malicious models should be with respect to the benign models' variance.* Nonetheless, given the assumption in A2 that the variance of malicious models σ^m may exceed that of benign models σ^b to an arbitrary extent, the hypothesis of Lemma A2 proves to be non-restrictive and readily achievable.

Proposition A1. *Under Assumptions 1 and 2, removing malicious clients (those in $\mathcal{M}$) from the federation yields a superior estimator of the global model. Specifically, the resulting estimator is unbiased (in the sense of Lemma 1) and exhibits a reduced variance (as shown in Lemma 2), leading to improved model accuracy and robustness.*

Proof. The proof is immediately derived from Lemmas A1 and A2. This is due to the fact that upon the exclusion of malicious clients, the federated averaging estimator reduces to the form presented in $\theta^{\mathcal{B}}_{avg}$, which is not only unbiased but also exhibits reduced variance—thereby diminishing noise in the global model's estimation. □

B. Datasets and Implementation Details

We conducted experiments on common FL benchmark datasets [34], namely FashionMNIST [35], CIFAR-10, and CIFAR-100 [36]. Since our goal was to detect malicious clients based on data characteristics, we sampled benign clients using a Dirichlet distribution with parameter $\alpha = 1000$ to ensure near-i.i.d. conditions. This setting ensures a fair comparison with the baselines, as `Waffle` neither relies on model updates nor is affected by class imbalance. Moreover, the shuffled training on a distillated dataset already exposes `Waffle` to synthetic heterogeneity. Introducing additional data imbalance would therefore not yield further insights into its performance.

For classification, we used LeNet-5 [41]. The standard version was applied to FashionMNIST, while we adjusted the input channels to three and modified the number of output classes for CIFAR-10 and CIFAR-100. The federation included $K = 100$ clients, with $|\mathcal{P}_t| = 10$ clients participating per round. Training was carried out over $T = 500$ communication rounds, using $S = 1$ local epoch per round and a batch size of 64. We employed the cross-entropy loss optimized with the ADAM optimizer [42], using an initial learning rate $\eta = 0.001$.

For `Waffle` , we used WST parameters $J = 3$, $L = 6$, and first-order coefficients [43]. The FT baseline employed a window size of 0.5. The `Waffle` detector was a multilayer perceptron with three hidden layers and hyperbolic tangent activations, trained for 100 epochs using ADAM. The attack parameters β and σ were randomly sampled from Unif$\{3, 5, \ldots, 19\}$ and Unif$(0.5, 2.0)$, respectively. For baselines, we used `mKrum` with $k = 5$, and `TrimmedMean` with a cut-off parameter of 0.2.

All computations were performed using the CPU of a MacBook Pro equipped with an Apple M3 Pro chip. No additional computational resources were employed.

C. Detection Metrics

In evaluating the performance of `Waffle` , we employed several detection metrics [38], [44], each offering a different perspective on the detector's effectiveness in binary classification task. Let TP, TN, FP, and FN represent True Positives (correctly identified malicious clients), True Negatives (correctly identified benign clients), False Positives (benign clients incorrectly flagged as malicious), and False Negatives (malicious clients incorrectly flagged as benign), respectively.

a) Accuracy: Accuracy is one of the most straightforward metrics, representing the overall correctness of the classifier. It is calculated as the ratio of correctly classified instances (both malicious and benign) to the total number of instances.

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

While intuitive, accuracy can be misleading, especially in scenarios with imbalanced datasets. For instance, if 90% of clients are benign, a detector that classifies all clients as benign would achieve 90% accuracy, despite failing to identify any malicious clients. Therefore, while providing a general overview, accuracy alone is often insufficient for evaluating a malicious client detector.

b) Precision: Precision measures the proportion of correctly identified malicious clients among all clients classified as malicious by the detector.

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

High precision indicates a low false positive rate, meaning that when the detector flags a client as malicious, it is highly likely to be correct. This is crucial in scenarios where incorrectly blocking a benign client (a false positive) has significant negative consequences, such as denying service to legitimate users. A low precision score suggests the detector raises many false alarms.

c) Recall: Recall measures the proportion of actual malicious clients that are correctly identified by the detector.

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

High recall indicates a low false negative rate, meaning the detector successfully identifies a large fraction of the malicious

clients present. This is critical in security applications where failing to detect a malicious client (a false negative) can lead to significant damage or compromise. A low recall score suggests the detector misses many malicious clients.

d) F1-Score: The F1-Score is the harmonic mean of Precision and Recall, providing a single metric that balances both concerns.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}$$

The F1-Score is particularly useful when there is an uneven class distribution, as it punishes extreme values of precision or recall. A high F1-Score indicates that the detector has both good precision and good recall, meaning it is both accurate in its positive predictions and captures a majority of the actual positive instances. It is often preferred over accuracy in imbalanced malicious client detection scenarios where both minimizing false alarms and maximizing detection of actual threats are important.

D. Privacy of `Waffle`

`Waffle` detector architecture prioritizes client privacy throughout its operation. Throughout the learning process, **individual raw data $\{x_k^i\}$ remains strictly on the client's device**. Each client k *privately* computes its PCA-derived representation $\hat{x}_k$ and subsequently its spectral embedding φ_k locally on its own hardware. Clients only transmit the resulting spectral embedding vector φ_k to the server, ideally over a secure communication channel to protect these embeddings while in transit. This φ_k is explicitly designed to be an aggregate statistic that captures characteristics of the data distribution without revealing individual data points, thereby serving as a non-privacy-leaking feature. Furthermore, since WST is non-invertible, it is also impossible to reconstruct the PCA representant. Our methodology is consistent with approaches in privacy-preserving machine learning where transformed or aggregated representations of data are used instead of raw sensitive information to train models or make inferences [45]. Furthermore, the offline training of the `Waffle` detector and Algorithm 1) is conducted on a distinct auxiliary dataset $\mathcal{D}^{\text{aux}}$, coherently with common practices [30], ensuring that no actual client data from the federation is used or exposed during the detector's training phase. The combination of local feature extraction by clients, the transmission of only these specialized spectral embeddings, and offline training using auxiliary data ensures that `Waffle` functions as a privacy-conscious safeguard within the FL ecosystem.