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

Anonymous Author(s)
Affiliation
Address
`email`

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` (**Wa**velet and **F**ourier representations for **F**ederated **Le**arning) 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.

1 Introduction

Federated Learning (FL) is a distributed learning framework that enables multiple clients to train a global model without sharing raw data [1], making it a promising approach for privacy-sensitive decentralized domains [2, 3, 4]. As deployments scale, a central challenge is data heterogeneity, where non-IID client data can hinder model convergence and degrade performance [5, 6].
Beyond natural data heterogeneity, FL systems are also vulnerable to malicious or faulty clients [7, 8]. Consider a scenario in a large-scale sensor network FL deployment (e.g., environmental monitoring [9] or industrial IoT [10]). Some sensors might be physically damaged, miscalibrated, or even deliberately tampered with, causing them to report highly perturbed, noisy, or statistically anomalous data streams. Being able to quickly detect them and remove them for training or fixing them without accessing raw data is beneficial for monitoring the network, and guaranteeing its correct functioning [11]. Defenses against malicious clients in FL often rely on robust aggregation methods [12, 7] to reduce the influence of outlier updates, particularly in the presence of Byzantine attacks [13]. However, these methods assume most clients are benign and target model-level anomalies, not data-level perturbations that subtly alter local objectives [14, 15]. Online detectors that monitor client behavior during training offer an alternative but add overhead and may act too late. Crucially, neither class of defenses is well-suited to detecting clients with malicious data prior to training, as in

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

compromised sensor networks.

In this work, we propose Waffle (**Wa**velet and **F**ourier representations for **F**ederated **Le**arning), an offline detector designed to identify clients with malicious data before FL training begins. Performing detection offline [16] and in a privacy-preserving manner is particularly desirable, as it limits computational overhead on both the server and client sides, making the approach lightweight and scalable. Waffle trains a classifier on spectral features – extracted via Fourier Transform (FT) and Wavelet Scattering Transform (WST) [17]– which offer stable, invariant representations robust to data perturbations. Detection is performed using a model pre-trained on a distilled public dataset, ensuring efficiency and privacy. Clients locally compute low-dimensional statistics via Principal Component Analysis (PCA) and spectral embeddings, sending an aggregated, secure, and non-invertible information to the server which classifies them as benign or malicious. Malicious clients are excluded prior to training. Unlike many existing methods, Waffle does not assume a benign majority and can be combined with robust aggregation to strengthen FL security.

The structure of the paper is the following: Section 2 defines the FL setting, the data attacks considered, and the spectral representations (FT and WST). Section 3 details Waffle 's training and detection. Theoretical guarantees are provided in Section 4, showing the benefits of removing malicious clients. Section 5 reports experimental results validating Waffle on benchmark datasets.

Related Works 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 [14]. FLDetector[18] identifies malicious clients by analyzing the consistency of their updates over time—benign updates follow predictable patterns, while malicious ones are erratic. MuDHog[19] 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 [20] 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 [21, 7]. KRUM[12] selects the most central update in ℓ_2 distance, but requires fewer than half of the clients to be malicious. TrimmedMean[7] discards extreme values per coordinate, improving robustness to outliers, though it overlooks dependencies across dimensions. FLTrust[8] uses a trusted server-side dataset to normalize and rescale client updates, enhancing robustness but breaking strict decentralization. Secure aggregation protocols like RFLPA[22] and RoFL [23] 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 [24, 25, 26]. FreqFeD[27] 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[28] 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.

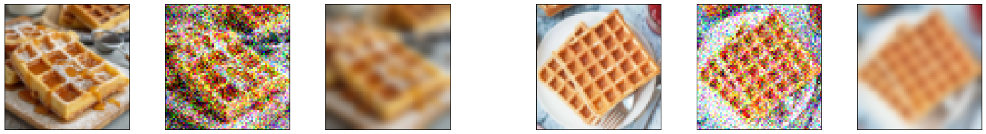

Figure 1: Examples of attacked data. Two images downloaded from link1 and link2 . For each image: *left*: clean client, *center*: noisy attack with magnitude $\sigma = 0.2$, *right*: blur attack with spread $\beta = 11$

2 Theoretical Framework

In this section, we introduce the mathematical framework that provides the foundation for our algorithm. Section 2.1 presents the Federated Learning (FL) setting and defines the class of attacks considered on clients' data. Section 2.2 introduces the WST and the Fourier Transform FT, recalling their basic properties that are relevant for anomaly detection.

2.1 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 $t \in \{1, \ldots, T\}$, 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.

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 [29, 30] –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:*

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

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

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.

2.2 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 [31]. 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) [17, 31]. 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.

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 [32]. 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 [33] 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.

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 [17, 31] 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 $Q \in \mathbb{N}$ [34].

To streamline notation, following [17], 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:

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

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.

3 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.

3.1 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 [35], 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 [36], 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*.

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.

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.

Principal Component Analysis For each simulated client k, PCA [37] 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.

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) [38], between the detector's prediction based on φ_k and the ground-truth label μ_k.

3.2 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 $k \in \{1, \ldots, K\}$ 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.

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

4 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.*

5 Experiments

In this section, we present experimental results on widely used federated learning benchmark datasets [39, 40, 41], 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 5.1 evaluates the detection performance of the two variants of `Waffle` , highlighting the differences between the WST and FT representations. In Section 5.2, we compare `Waffle` against standard Byzantine-resilient FL baselines, including `FedAvg` [1], `Krum` and `mKrum` [12], `GeoMed` [42], and `TrimmedMean` [7]. 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.

Table 1: **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 [43]) 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	65.11	60.9	**70.0**	70.0	79.75	68.42	**95.59**	67.0	56.69	41.38	**90.0**	45.0
	`Waffle` - WST	**71.88**	**95.83**	57.5	**82.0**	**94.87**	**97.36**	92.5	**96.0**	**83.33**	**93.75**	75.0	**88.0**
90%	`Waffle` - FT	**81.81**	95.45	**71.59**	**72.0**	**93.05**	89.69	**96.67**	**87.0**	88.1	88.1	**88.1**	**80.0**
	`Waffle` - WST	65.65	**100.0**	48.86	55.0	90.91	**100.0**	83.33	86.0	**88.85**	**100.0**	67.86	73.0

5.1 `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 1, the quantitative results at the client level differ between the two variants. We report standard detection metrics: precision, F1 score, recall, and accuracy [43], 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 1 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.

5.2 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`

Table 2: Comparison between baselines for detecting malicious clients and `Waffle` (with both WST and FT). `Waffle` -WST combined with `FedAvg` achieves the highest test accuracy across all datasets, outperforming baselines designed to mitigate Byzantine attacks. Results also highlight the orthogonality of `Waffle` to aggregation methods, consistently improving their performance. For reference, the test accuracy of `FedAvg` without malicious clients is: FashionMNIST 75.08%, CIFAR-10 50.24%, CIFAR-100 17.72%.

Dataset	**Setting**	`FedAvg`	`Krum`	`mKrum`	`GeoMed`	`TrimmedMean`
FashionMNIST	w/o detector	73.33	73.85	70.56	72.75	74.84
	`Waffle` - WST	**76.18**	70.26	74.75	74.18	75.21
	`Waffle` - FT	73.38	72.10	74.40	75.35	74.98
CIFAR-10	w/o detector	48.75	45.2	47.4	48.51	48.22
	`Waffle` - WST	**49.70**	46.28	49.08	49.41	49.0
	`Waffle` - FT	46.95	44.13	47.58	47.14	46.86
CIFAR-100	w/o detector	16.35	9.61	14.73	16.83	16.85
	`Waffle` - WST	**17.12**	8.50	14.85	16.32	15.89
	`Waffle` - FT	11.58	7.24	10.15	12.25	10.29

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 2, 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.

6 Conclusion

We propose `Waffle` , a novel offline algorithm to detect malicious client data in Federated Learning (FL) before training. Exploiting stable spectral features extracted via the Wavelet Scattering Transform (WST) and Fourier Transform (FT), it enables robust anomaly detection from private, low-dimensional client-side summaries built on publicly distilled data. By filtering out compromised clients prior to training, `Waffle` significantly improves convergence speed, final model accuracy, and robustness to data contamination. It achieves near-perfect precision (100% in our benchmarks) even in extreme scenarios with up to 90% malicious clients, outperforming strategies that rely solely on robust aggregation. This early detection mechanism also reduces training time, communication overhead, and energy consumption—factors crucial in large-scale deployments. Furthermore, `Waffle` is model-agnostic and can be seamlessly integrated with existing FL defenses to enhance overall system security.

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 of `Waffle` to realistic FL scenarios in vision, healthcare, and IoT.

Limitations. `Waffle` targets data-level attacks altering client input features, not model-level attacks (e.g., gradient manipulation, backdoors), which necessitate different defense strategies. However, combining `Waffle` with robust aggregation can help mitigate such hybrid threats.

Broader Impact. Our method enhances FL robustness and trustworthiness, crucial for deployments in sensitive domains (e.g., healthcare, finance), and reduces resource consumption. Potential misuse (e.g., unfairly excluding outlier populations) warrants careful auditing and fairness-aware deployments, though `Waffle` itself introduces no new privacy or fairness risks beyond those inherent in existing FL pipelines.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[3] Rodolfo Stoffel Antunes, Cristiano André da Costa, Arne Küderle, Imrana Abdullahi Yari, and Björn Eskofier. Federated learning for healthcare: Systematic review and architecture proposal. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–23, 2022.

[4] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated learning: privacy and incentive*, pages 240–254. Springer, 2020.

[5] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[6] Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8397–8406, 2022.

[7] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International conference on machine learning*, pages 5650–5659. Pmlr, 2018.

[8] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[9] Daniel Leiria, Hicham Johra, Justus Anoruo, Imants Praulins, Marco Savino Piscitelli, Alfonso Capozzoli, Anna Marszal-Pomianowska, and Michal Zbigniew Pomianowski. Is it returning too hot? time series segmentation and feature clustering of end-user substation faults in district heating systems. *Applied Energy*, 381:125122, 2025.

[10] Merim Dzaferagic, Nicola Marchetti, and Irene Macaluso. Fault detection and classification in industrial iot in case of missing sensor data. *IEEE Internet of Things Journal*, 9(11):8892–8900, 2021.

[11] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Mamoun Alazab, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*, 195:346–361, 2022.

[12] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[13] Stefano Marano, Vincenzo Matta, and Lang Tong. Distributed detection in the presence of byzantine attacks. *IEEE Transactions on Signal Processing*, 57(1):16–29, 2008.

[14] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[15] Francesco Colosimo and Floriano De Rango. Median-krum: A joint distance-statistical based byzantine-robust algorithm in federated learning. In *Proceedings of the Int'l ACM Symposium on Mobility Management and Wireless Access*, pages 61–68, 2023.

[16] Hangyu Zhu, Haoyu Zhang, and Yaochu Jin. From federated learning to federated neural architecture search: a survey. *Complex & Intelligent Systems*, 7(2):639–657, 2021.

[17] Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[18] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang 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*, pages 2545–2555, 2022.

[19] Ashish Gupta, Tie Luo, Mao V Ngo, and Sajal 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*, pages 445–465. Springer, 2022.

[20] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[21] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International conference on machine learning*, pages 3521–3530. PMLR, 2018.

[22] Peihua Mai, Ran Yan, and Yan Pang. Rflpa: A robust federated learning framework against poisoning attacks with secure aggregation. *Advances in Neural Information Processing Systems*, 37:104329–104356, 2024.

[23] Hidde Lycklama, Lukas Burkhalter, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Robustness of secure federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 453–476. IEEE, 2023.

[24] Zexin Wang, Changhua Pei, Minghua Ma, Xin Wang, Zhihan Li, Dan Pei, Saravan Rajmohan, Dongmei Zhang, Qingwei Lin, Haiming Zhang, et al. Revisiting vae for unsupervised time series anomaly detection: A frequency perspective. In *Proceedings of the ACM Web Conference 2024*, pages 3096–3105, 2024.

[25] Chi-ho Chan and Grantham KH Pang. Fabric defect detection by fourier analysis. *IEEE transactions on Industry Applications*, 36(5):1267–1276, 2000.

[26] Ran Tao, Xudong Zhao, Wei Li, Heng-Chao Li, and Qian Du. Hyperspectral anomaly detection by fractional fourier entropy. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(12):4920–4929, 2019.

[27] Hossein Fereidooni, Alessandro Pegoraro, Phillip Rieger, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning. *arXiv preprint arXiv:2312.04432*, 2023.

[28] Yu Chen and Zihan 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.

[29] Abhishek B Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Transactions on Sensor Networks (TOSN)*, 6(3):1–39, 2010.

[30] Yeping Peng, Zhen Tang, Genping Zhao, Guangzhong Cao, and Chao Wu. Motion blur removal for uav-based wind turbine blade images using synthetic datasets. *Remote Sensing*, 14(1):87, 2021.

[31] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[32] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[33] Ronald N Bracewell. The fourier transform. *Scientific American*, 260(6):86–95, 1989.

[34] Mathieu Andreux, Tomás Angles, Georgios Exarchakis, Roberto Leonarduzzi, Gaspar Rochette, Louis Thiry, John Zarka, Stéphane Mallat, Joakim Andén, Eugene Belilovsky, et al. Kymatio: Scattering transforms in python. *Journal of Machine Learning Research*, 21(60):1–6, 2020.

[35] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[36] Wenxuan Bao, Haohan Wang, Jun Wu, and Jingrui He. Optimizing the collaboration structure in cross-silo federated learning. In *International Conference on Machine Learning*, pages 1718–1736. PMLR, 2023.

[37] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[38] Usha Ruby, Vamsidhar Yendapalli, et al. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.

[39] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[41] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[42] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

[43] Jake Lever. Classification evaluation: It is important to understand both what a classification metric expresses and what it hides. *Nature methods*, 13(8):603–605, 2016.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As clarified by experiments, our proposed method not only outperform existing FL methods, but also have the critical advantage to work offline and not during training.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in a paragraph of the Conclusions section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide full and revised proofs in Appendix A

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Indeed, all relevant information is thoroughly addressed within the paper. Moreover, the implementation details are meticulously elaborated upon in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Indeed, we offer an anonymized GitHub repository containing all necessary information to accurately reproduce the results as well as the algorithms detailed and presented within the paper. The datasets used are public and easily downloadable, and splits are reproducible.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All implementation details on splits, hyperparameters, optimizers and architectures are detailed in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix B contains the results obtained from experiments conducted with three distinct random seeds, thereby ensuring statistical significance for our findings and experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We thoroughly describe computational resources in Appendix B, specifying the characteristics of the machines on which experiments have been conducted.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms in every respect, with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impacts are discussed in the Conclusions

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Code and data sources are acknowledged through explicit citations.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.