sciendo

VOLUME 18, N° 3 2024
Journal of Automation, Mobile Robotics and Intelligent Systems

Tackling Non-IID Data and Data Poisoning in Federated Learning Using Adversarial Synthetic Data

Submitted: 27th December 2023; accepted: 11th March 2024

Anastasiya Danilenka

DOI: 10.14313/JAMRIS/3-2024/17

Abstract:
Federated learning (FL) involves joint model training by various devices while preserving the privacy of their data. However, it presents a challenge of dealing with heterogeneous data located on participating devices. This issue can further be complicated by the appearance of malicious clients, aiming to sabotage the training process by poisoning local data. In this context, a problem of differentiating between poisoned and non-identically-independently-distributed (non-IID) data appears. To address it, a technique utilizing data-free synthetic data generation is proposed, using a reverse concept of adversarial attack. Adversarial inputs allow for improving the training process by measuring clients' coherence and favoring trustworthy participants. Experimental results, obtained from the image classification tasks for MNIST, EMNIST, and CIFAR-10 datasets are reported and analyzed.

Keywords: *federated learning, non-IID data, label skew, data poisoning, label flipping*

1. Introduction

Federated learning (FL) [1] focuses on developing a global model by coordinating learning on multiple devices while maintaining the privacy of local data. The typical process of FL consists of training rounds and involves several steps: (1) the global model is initialized on the server; (2) the subset of clients of a specified size is randomly chosen from all available clients; (3) the global model is shared among the selected subset of clients; (4) clients perform local training with the received global model for a limited number of epochs using their private data; (5) clients return their model updates to the server; and (6) model updates are aggregated on the server into a new version of the global model [1].

Each of the default FL steps is open to changes and refinements. Thus, the subset of clients for a training round may be created not randomly, but by following a strategy, clients may not send weight updates to the server, but the results of SGD [1], or communicate full model weights. Moreover, aggregation of the model updates on the server may not simply average all received model updates as proposed in the FedAvg algorithm, but prioritize one client's updates over another's. For instance, by using the size of their local datasets as weights [1], participants with big local datasets are favored as they contribute more to the new global model during aggregation.

Due to the privacy restrictions of FL, clients' local datasets remain on their local devices, making it impossible to perform centralized data analytics and infer properties of both global and local datasets. Moreover, in real-life cases, data may not be identically independently distributed (non-IID) among clients, which was proved to cause problems for FL, as the quality of the global model and its convergence can be negatively impacted by the presence of such data [2, 3]. Non-IID data can be categorized into five types [4], i.e., (1) feature distribution skew (different clients have variations in feature styles for the same label); (2) label distribution skew (clients have varying label distributions but similar features for specific labels); (3) same label, different features (different clients present different feature distributions for the same label); (4) same features, different labels (clients assign different labels to the same features); and (5) quantity skew (differences in the amount of data across clients).

In general, these data-related skews are supposed to be the result of the natural characteristics of the data, highlighting the complexity and diversity of real-life federated datasets. However, another set of data issues can come from malicious actors, which have access to client devices and client data, resulting in a security issue known as a data poisoning attack [5]. In this case, the adversary may perform data poisoning attacks and aim to compromise the training process, reduce the global model performance, and cause incorrect model predictions during the inference stage [6]. Despite poisoned data being different from the non-IID data problem, FL by default equally protects the privacy of non-IID and malicious clients, naturally making the task of distinguishing between them more challenging.

To address the challenges posed by label skew non-IID data, the *Adversarial Federated Learning* (AdFL) algorithm was introduced [7]. This method originated from the concept of adversarial attacks and is mainly applicable to neural networks dealing with image data. The AdFL algorithm allows for gaining valuable insights about clients' local datasets without requesting any additional information from local devices by utilizing synthetic data generated on the server.

The algorithm improves the performance of global models in the presence of label skew data and results in more stable training and more balanced per-class accuracy of the global model.

This work is an extension of the research presented in [7] and explores the possibility of using self-adversarial samples for distinguishing malicious non-IID clients from those that are benign, focusing on untargeted and targeted label-flipping attacks.

Following this, in Section 2, related research on data poisoning attacks in the presence of non-IID data within FL is outlined. Section 3 presents key concepts of adversarial attacks. In Section 4, the AdFL algorithm is described. Section 5 defines the data poisoning attacks adopted in this paper. Section 6 covers the experimental results and their analysis, collected from MNIST [8], EMNIST [9], and CIFAR-10 [10] datasets. This work concludes with a summary of findings and future research suggestions.

2. Related Work

The data poisoning attacks in FL as a standalone issue are being addressed in multiple ways. For instance, malicious clients can be detected and filtered out from the training. Here, methods were proposed to track the consistency of the client's updates to verify its intent [11], apply dimensionality-reduction and clustering techniques (such as kernel principal components analysis and k-means) [12], or using Euclidean distance-measure [13, 14] to distinguish between malicious and benign clients and filter out suspicious clients. Another approach proposed is to maintain a small clean training dataset and a separate model on the server, using them to assess the trustworthiness of clients' updates by comparing the direction of local models updates with the server-side model update obtained from the clean dataset and further using trustworthiness score as aggregation weights for normalized clients' model updates [15]. Another line of research focused on modifying the aggregation strategy towards outlier resistance, for example, is by taking not the mean, but the median for each dimension from the model updates [16] or trimming the updates before averaging [17] to avoid extreme values. However, these methods mainly rely on the assumption that benign clients will have similar model updates, which can not be guaranteed under the non-IID data.

To address the joint problem of possible data poisoning attacks and non-IID data, methods for both malicious clients detection and non-IID data mitigation were proposed. For instance, the algorithm utilizing a cosine similarity measure was presented to assess clients' contribution similarity, assuming that benign clients will have more diverse gradient updates than coordinated malicious clients [18]. Another approach suggested using a small proxy dataset as a tool to perform on-server optimization to find the best model updates fusion and mitigate possible malicious clients effect by naturally assigning them small aggregation weights [19].

A different solution proposed analyzing the critical parameters of the local models to reliably identify malicious clients and use it for weighted updates aggregation [20]. An attack-tolerant FL method was also proposed, presenting local meta updates and global knowledge distillation to mitigate possible malicious clients effect on the global model [21].

Although the research has begun to simultaneously address the problem of both non-IID data and potential data poisoning attacks in FL, the proposed solutions can still rely on proxy datasets available on the server side or complicate the local training process with additional computations. Such assumptions may not be feasible in some FL scenarios. Moreover, the complexity of the non-IID data problem and the variety of data poisoning attack scenarios make it harder to find solutions that can satisfy both performance and a variety of security goals, leaving this challenging area open for further research.

3. Adversarial Federated Learning

Adversarial data are adopted by many FL algorithms. The common idea is utilizing adversarial techniques as data generators in order to (a) defend the model against adversarial attacks [22, 23], or (b) augment the quantity of locally accessible data with synthetic samples [24–26]. This work extends the applicability of the previously proposed alternative method, incorporating adversarial data into FL.

3.1. Adversarial Attack

The essence of adversarial attacks lies in the ability to modify a sample from the training data of a neural network in a manner that is imperceptible to humans, yet causes the trained network to incorrectly classify what was once a correctly classified sample [27]. This phenomenon was illustrated to be caused by the ability of the adversary to alter the target data sample in a way that makes it cross the classifier's decision boundary, and, therefore, result in misclassification [28].

The classification of adversarial attacks falls into two main categories: *untargeted* attacks, which simply focus on causing any incorrect classification, and *targeted* attacks, where the goal is to trigger misclassification into a specific class. Attack methodologies are further divided into *white-box* attacks, when the involved adversary has access to the model's architecture and parameters, and *black-box* attacks, which rely solely on the attacker's access to output data. A set of gradient-based algorithms was previously presented that relies on the model's gradients and a loss function to create the necessary changes to the source data in order to perform an attack. For instance, gradient-based algorithms are: one-step Fast Gradient Sign Method (FGSM) [29], its iterative version I-FGSM [30], and its version enhanced with momentum MI-FGSM [31].

In this study, the momentum iterative fast gradient sign method (MI-FGSM) is used to perform targeted adversarial attacks [31] (see Equations 1 and 2).

$$g_{t+1} = \mu * g_t + \frac{\nabla_x J(\theta, x_t^*, y))}{||\nabla_x J(\theta, x_t^*, y))||_1} \quad (1)$$

$$x_{t+1}^* = x_t^* + \alpha * sign(g_{t+1}) \quad (2)$$

Here, g_t represents the accumulated gradients, x_t^* is the perturbed adversarial image in iteration t, y is a target class, J is a loss function, μ is a decay factor introduced for better attack success rate and α is a step size. At each iteration, x_t^* is clipped in the vicinity ϵ, to preserve the resulting adversarial image within L_∞ distance from the source image.

3.2. Transferability of Adversarial Inputs

Adversarial inputs possess the ability to transfer across models, meaning that adversarial inputs designed for one model can also cause mispredictions from other models, with the transferability of adversarial samples being higher between models trained on data and tasks that are similar. This phenomenon is attributed to the fact that models addressing the same task tend to develop similar decision boundaries. In the context of FL, where clients work on the same task with shared model architecture and feature space, this transferability is particularly useful. It was shown, that adversarial samples generated by any client can provide insights about local data distribution [7]. The property of transferability of adversarial samples and its relevance to decision boundaries of trained classifiers in FL formed the basis of the AdFL algorithm.

4. AdFL Algorithm

In the AdFL algorithm, adversarial images are utilized as an additional source of information to improve and guide the training process. The generation of these images is done on the server, using the models that have been updated and a random noise sample image as a starting point for the generation process. This way, the adversarial images are generated in a data-free way, meaning, that no access to actual clients' data is needed. The specific steps performed by the server in the AdFL are outlined in Algorithm 1.

Note, that in the AdFL algorithm, the weights of the model are communicated between clients and server.

In total, six steps summarize the AdFL algorithm:

1) During the first federated training round, all clients receive the initialized global model, perform local training, and return the resulting models back to the server.
2) Updated models returned by clients are used to generate adversarial samples (Section 4.1).
3) The estimation of the distribution of classes across clients is performed using the generated adversarial samples, as discussed in Section 4.2.
4) Each client gets a CS calculated with the help of updated models and the generated adversarial samples (see Section 4.4 for details).
5) The aggregation step utilizes clients' coherence scores as weights to form the next version of the global model. This new global model is then distributed to a new subset of clients, initiating the next training round.
6) Thereafter, the client-picking strategy, guided by the global classes distribution (see Section 4.3), regulates which subset of clients will engage in the next round of training, and the process repeats from step one, omitting the estimation of the distribution of classes across clients.

It should be emphasized, that all steps introduced by the AdFL that expand the classical FL pipeline are performed on the server. The adversarial images creation, client picking strategy, and coherence scores calculation are covered in the next subsections.

Algorithm 1 AdFL algorithm (Server); Cl – client; Cl_e – subset of clients picked for training on epoch e; *global distribution* – distribution of classes during FL training; $distr_{all}$ – estimated classes presence in clients' local datasets

Ensure: global model w_0, *global distribution*, clients ready
  for e in $epochs$ **do**
    if $e == 0$ **then**
      $Cl_e \leftarrow$ *all clients*
    else
      Cl_e, *global distribution* $\leftarrow$ pick clients for training ($distr_{all}$, *global distribution*)
    end if
    for Cl in Cl_e **do**
      $w_e^{Cl} \leftarrow$ run training(w_e)
    end for
    adv data $\leftarrow$ create adv data($[w_e^0, ..., w_e^{Cl_e}]$)
    if $e == 0$ **then**
      $distr_{all} \leftarrow$ estimate distribution(*adv data*)
    end if
    $CS_{[0-Cl_e]} \leftarrow$ calculate CS(*adv data*, $w_e^{[0,..,Cl_e]}$)
    $w_e \leftarrow$ FedAvg($[w_e^{[0,..,Cl_e]}]$, $CS_{[0-Cl_e]}$)
  end for

4.1. Adversarial Inputs Creation

Adversarial data in the AdFL algorithm is created based on the models returned by clients and does not require actual clients' data. Therefore, the adversarial input generation starts from a random noise image and is performed with the MI-FGSM algorithm (see Equations 1 and 2). This attack is parameterized by μ, α, the number of steps, and the clipping boundary. As the adversarial inputs produced by the AdFL algorithm are not used to perform actual adversarial attacks, the constraints on the amount of change applied can be relaxed. For example, the number of steps, α, and the clipping boundary can be viewed as constraints on the algorithm so that the final adversarial image is not far from its source, therefore, they were adjusted according to the objective.

It was experimentally validated that going beyond 30 adversarial steps does not improve transferability, thus, the number of steps was set to 30. The step size α was set to 1, while the clipping boundary and μ were left unchanged, following the original MI-FGSM research.

Algorithm 2 outlines the process for creating adversarial inputs, while the default federated steps are not included.

Algorithm 2 Adversarial data generation

Ensure: $targets \leftarrow [0, ..., C-1]$
Ensure: $w_e^{[0,..,Cl_e]}$ ▷ Updated models at epoch e
  for t in $targets$ **do**
    for w_e^i in $w_e^{[0,..,Cl_e]}$ **do**
      $adv\ img \leftarrow$ random noise$[Ch, H, W]$
      for $step$ in $num\ steps$ **do** ▷ MI-FGSM step
        $adv\ img_t^i \leftarrow \text{step}(w_e^i, adv\ img_t^i, t)$
      end for
    end for
  end for

After the local training, each updated model returned to the server is used to generate C images, i.e., one image is generated per class that is present in the classification task.

4.2. Local Distribution Estimation

As specified in the Algorithm 1, the first training round in the AdFL algorithm involves all clients in performing the local training. These models are then used to create adversarial samples (Section 4.1). During the research, it was determined that when one client's updated model makes predictions on adversarial samples created by another client's updated model at the end of the first epoch, these predictions are indicative of the specific classes present in the local dataset of the client that performed the predictions [7]. Therefore, at the end of the first round, it is possible to estimate the label distribution among all clients, that participated in the training round.

Detecting labels' presence in local data by inspecting the adversarial data predictions presents the opportunity for further improvements in the federated training process, based on the insights gathered. However, it is worth emphasizing, that the discovered label distribution is still an estimation.

4.3. Client-picking Strategy

Once the classes in the clients' local datasets are estimated, a client-picking strategy can be used to reduce the effects of label skew in the local datasets. In the AdFL algorithm, the influence of label skew on the training process is being addressed with a balanced client-picking.

The balanced client-picking is performed by utilizing the information retrieved during the local distribution estimation step described in the previous subsection and aims at having the clients with diverse local data label distributions picked for each training round.

This strategy ensures equal representation of common and rare classes in each training round, therefore, continuously exposing the model to all possible classes in the classification task, leading to a more balanced performance across all classes.

To track which classes were present on the clients that participated in the training process, a global label frequency vector of size C is maintained on the server, accumulating the number of clients that participated in training epochs up to now and had a certain class c in their local dataset. As a new subset of clients is being formed for a training round, the vector is updated with the label distribution information of each client added to the subset for this federated training round.

To maintain the balanced FL training and consistent involvement of all classes in the training, the clients for each new federated round are picked in such a way as to bring the values in the global frequency vector closer to a uniform distribution. To do so, a Kullback–Leibler divergence is used (Equation 3).

$$D_{kl}(P||Q) = \sum_{x \in X} P(x) log(\frac{P(x)}{Q(x)}) \tag{3}$$

Therefore, prior to adding a certain client to a subset of clients for the training round, the KL-divergence is calculated with respect to the uniform distribution and the global label frequency vector assuming that this client is added to the training, i.e., its classes are admitted to the global classes frequency. This technique ensures that clients who possess rare labels in their data are consistently included in the training.

4.4. Clients Coherence Measurement

Transferability of adversarial samples is not guaranteed by default for all federated clients, as it relies on the internal properties of the model and the data it was trained on. As presented in Section 4.2, examining the predictions of the models that only completed their first federated training round can help identify their local distribution, since this is what can be seen in the predictions the models make based on adversarial samples. Consequently, these predictions can identify which classes are not in the local distribution, locating nodes with rare data. However, this property can be used not only for label distribution estimation but also for assessing how close to each other the updated clients' models are. This assessment in the AdFL algorithm is called a coherence score (CS) and is employed to find clients with a high ability to correctly predict adversarial samples as well as produce those that are correctly predicted by other models.

Thus, the CS consists of two parts, i.e., the model's ability to (1) produce samples transferable to other models and (2) predict adversarial samples from other models. The calculation of these metrics is performed each training round after the updated client models return to the server after performing local training. Each updated model generates C adversarial samples and makes predictions for all adversarial samples generated by other updated models returned by clients participating in the current training round.

After the predictions are done, the score calculation proceeds with calculating the model's ability to predict adversarial images produced by other models according to Equation 4. For each multiplication, there is a binary indicator determining if the prediction for the adversarial sample for class c from model k was accurate and the class probability given by the model. The results obtained for the model predicting its own adversarial inputs are not included.

$$\text{predicted others} = \sum_{k=1}^{K} \sum_{c=0}^{C-1} \text{is correct}_{k,c} * \text{prob.}_{k,c} \tag{4}$$

The same formula is used to assess the model's ability to produce transferable samples that are recognized by other models, where the correct predictions are analyzed across the models that made predictions of the samples produced by the currently evaluated model.

The final CS is a summation of the two assessment results and is calculated as:

$$\textit{coh. score} = \textit{predicted others} + \textit{was predicted} \tag{5}$$

The resulting normalized coherence scores are used to favor models with good transferability and are employed as weights during the updated models averaging process, therefore, have a direct influence on the global model aggregation.

The AdFL algorithm can utilize coherence scores to identify clients which cannot reliably classify adversarial inputs or create such inputs. This property of the algorithm can be useful when dealing with data poisoning attacks, that interfere with the client's local data during the training process. Moreover, in conformity with the literature overview, it provides a weighting scheme for potentially assigning more importance to benign clients over malicious ones. Therefore, this property of CSs inspired this research, extending the application of the AdFL algorithm beyond non-IID label-skew scenarios.

5. Data Poisoning Attacks in FL

Data poisoning attacks can be classified into two categories based on the target of adversarial manipulation: clean data attacks and dirty-label attacks [32]. The first type of attacks does not change the labels of the data, on the contrary, it injects changes to the subset of training data [33] and does not require access to data labeling, while the second attack type, changes the labels of the samples inside the dataset, according to the adversary's goal and leaving data features unchanged [34].

As the non-IID scenarios considered in this work are represented by the label skew, the natural type of attack to consider as its "companion" is a dirty label label-flipping attack. Meaning, that in addition to the limited classes being present inside the local data, it can further be a subject of local data suffering from labels being flipped.

In these terms, data poisoning attacks can be performed by federated clients. Here, the attack can be described from the perspective of the number of clients participating in the attack — whether there is only a limited number of adversaries or if there are many — as well as from the way the source data labels are being affected, whether the adversaries do not have a specific strategy and the labels are flipped randomly [35], or they have a specific objective and flip labels according to some rule [36].

In the current research, two label-flipping strategies are being studied: untargeted(random) label flipping and targeted label flipping, meaning that labels for one class are consistently substituted by labels from another class. In both scenarios, adversaries do not have a way to see benign clients data distributions, however, in the target label-flipping scenarios malicious clients have a joint pair of source and target labels for the attack. This pair is known to all adversaries. The detailed description of attack scenarios employed in this work is given in Section 6.4.

The random label-flipping attack primarily focuses on the overall performance degradation of the global model, while targeted attacks have a target class whose performance they aim to damage. In order to assess whether the targeted attacks were successful, the Attack Success Rate (ASR) measure is employed (Equation 6) with respect to the label whose performance is targeted.

$$ASR = \frac{\text{number of successful attacks}}{\text{total number of attacks}} \tag{6}$$

It is also worth saying that in the presence of highly skewed data partition with equal class probabilities inside local data, random label flipping results in a softer attack scenario. For example, with 2 classes being present on a local node, there are around 50% of correctly assigned labels inside every class, as random assignment is not prohibited from picking the actual class.

6. Experimental Setup, Results, and Analysis

6.1. Datasets

For experiments, three image datasets were used – MNIST [8], EMNIST [9], and CIFAR-10 [10]. The datasets represent tasks of varying difficulty for the algorithms and are commonly utilized as benchmark datasets in FL research. MNIST offers 10-class, 28x28 grayscale images, and is often used as a basic image classification task. EMNIST expands the task with hand-written letters, increasing the number of classes to 62, adding complexity to label-skewed data, and making targeted label attacks harder to spot and counter. CIFAR-10 further escalates the challenge, introducing 10 classes, and 32x32 pixels RGB images with more complex features.

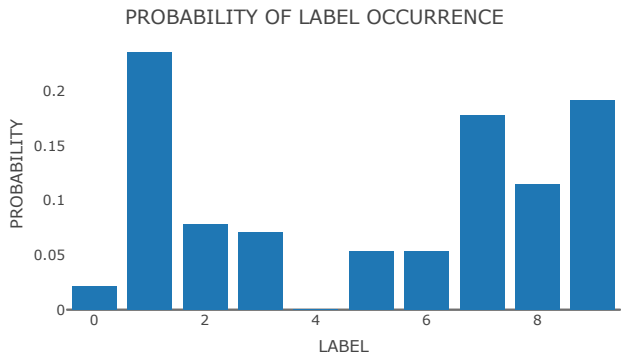

Figure 1. Probability of occurrence for 10 classes

6.2. Experimental Setup

The project was implemented in Python (version 3.7.9), using PyTorch machine-learning framework (version 1.10.0 [37]). Datasets and custom model architectures were provided by the torchvision package (version 0.11.1 [38]). All experiments were run on GPU hardware, specifically, NVIDIA GTX 1070 and NVIDIA A100.

6.3. Data Partitioning

Non-IID label skew was emulated on local devices according to the following procedure: (1) in the parameters of the experiment, the number of unique classes L and the total number of data samples N are defined and applied for all clients, (2) random seed is set for random operations reproducibility, (3) the probability of each class appearing on the local model is defined by taking a sample from a normal distribution, (4) for each client a set of classes in their respective local dataset is determined by drawing a sample of size L from the class probability distribution, (5) a unique subset of total N data samples of the selected classes is assigned to the client, with each label having N/L samples in the local dataset.

As the number of unique label-skew distributions that can be generated with this approach is immense and the results obtained across different distributions cannot be simply aggregated due to these differences in statistical properties of the datasets, the experiments were performed on a fixed data distribution. This helped prevent data-dependent features from interfering with performance metrics and made it more reliable to attribute differences in model performance to specific algorithms and data poisoning attacks used.

The probability of label occurrence used for classification tasks with 10 classes (MNIST and CIFAR-10) is illustrated in Figure 1.

Due to the normal distribution being used to create the label probability distribution for the whole experiment, some classes naturally appear more often in the local dataset than others. It additionally introduces the global class imbalance to the FL pipeline. In a case when the number of federated clients is low or the number of unique classes in the local datasets is low, some classes may not appear at all.

6.4. Assumptions and Data Poisoning Attack Model

In the considered attack scenarios it is assumed that the server is fair and not compromised – only a set of malicious clients are threatening the FL pipeline. Moreover, the attackers are present in the FL pipeline from the beginning and stay till the end of training – no attacker leaves or joins the training in the process. The design of the FL experiment is adapted from the FoolsGold algorithm non-IID scenario [18] and features 15 federated clients: 10 honest clients and 5 malicious clients. However, changes were made to the data partition scenario compared to the reference experiment according to the data partition strategy presented in the previous section. These changes modify the data distribution strategy and introduce the label skew to the clients' local datasets as adopted in the experiments designed for the algorithms focused on mitigating the effects of non-IID data.

According to the scenarios presented in Section 5, three attacks were designed for experiments: (1) untargeted random flipping attack, (2) targeted attack on a common label, (3) targeted attack on an uncommon label.

During the untargeted attack, every malicious client randomly assigns labels to local data samples based on the available local labels set. The targeted attacks include malicious clients jointly picking their target. First, malicious clients reveal the set of labels present in their local data. Then, malicious clients estimate label distribution based on the observed local distributions. The attack rule is defined as a pair of labels c_{source}, c_{target}, where c_{source} is a label that will be flipped with c_{target}, therefore, c_{source} performance will be targeted. After setting the attack pair, every malicious client inspects its data and looks for c_{source}. If found, the flipping is performed according to the attack rule. Moreover, the targeted attacks utilize non-IID data properties, defined in the previous section, by targeting either common or uncommon labels based on the empirical label distribution collected by the malicious clients. Common labels are defined as labels, whose probability of occurrence exceeds the 66% quantile of the probability vector, while uncommon are defined as those under the 66% quantile.

This way, targeted attack schemes naturally limit the active number of active attackers, as they are based on the empirical label probability distribution estimated by attackers before training starts. Common labels are not guaranteed to be present in every attacker's local data due to the 66% quantile threshold, making some potentially malicious clients benign during training, however, still contributing to the overall distribution estimation.

In the presented scenarios with 5 malicious federated clients, an untargeted attack results in all 5 clients being malicious during the training, a targeted attack on common labels results in around 3 clients performing the joint attack on a certain label, while the attack on an uncommon label is done by one malicious client, therefore, reflecting the adaptation of the attacking party to the observed non-IID data distribution.

Table 1. Hyperparameters used in the experiments

Parameter	MNIST	EMNIST	CIFAR-10
Total labels	10	62	10
Total clients	15	15	15
Labels on client	3	19	3
Local data size	600	1900	600
Learning rate	0.001	0.001	0.001
Batch size	10	10	10
Fed epochs	10	10	2
Adv. steps	30	30	30

6.5. Models and Hyperparameters

Convolutional neural networks (CNNs) were chosen for the image classification tasks represented in the datasets. The basic LeNet5 [39] architecture was adopted for both MNIST and EMNIST tasks. For CIFAR-10 a more sophisticated architecture was chosen, namely, mobilenetv2 [40]. The pre-trained version of the model was provided by the torchvision package with weights coming from the Imagenet [41] dataset.

Each experiment used the cross-entropy loss function and the Stochastic Gradient Descent optimizer. The full list of parameters for experiments with respect to datasets is given in Table 1.

For each algorithm, dataset, and attack type, the training was performed 5 times with different model initializations controlled by a set of seeds, and mean results were used for further analysis.

6.6. Experimental Results and Analysis

To evaluate the AdFL algorithm's ability to identify malicious clients and mitigate their effect in the presence of non-IID data, two well-known algorithms were chosen as baselines for evaluation. The first one is Multi-Krum [13] which uses the Euclidean distance metric to find m closest models to use for global model aggregation, rejecting the rest of the updates collected on the server during the FL training round. This approach favors model updates that are similar to each other and treats unusual updates as malicious. The second approach taken into comparison is the FoolsGold (implementation for the experiments was based on the source code of the algorithm provided by the authors [18]). This approach employs a different strategy and uses cosine similarity measure to identify clients with similar gradient updates and assigns them smaller importance during global model update. This algorithm serves as an example of a weighting approach that was initially evaluated on non-IID data. Moreover, it is an example of a defense that is not suited for untargeted attacks [42]. Therefore, the two selected baseline algorithms present two different approaches to defense against label-flipping attacks.

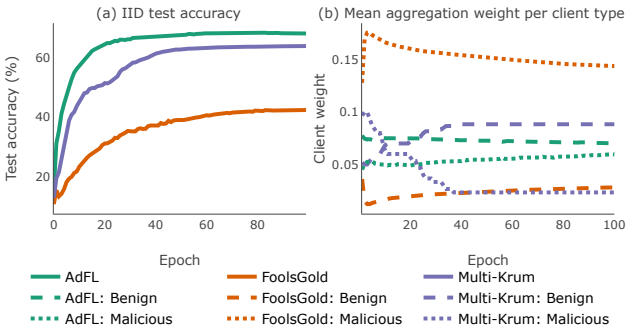

Figure 2. MNIST untargeted attack

As the AdFL algorithm was not designed as a defense against data poisoning and the current research aims at extending the usage of generated self-adversarial samples to the FL security domain, these baseline methods serve more as representatives of the algorithms designed to defend FL training rather than competitors in terms of defense efficiency.

It is important to note, that the Multi-Krum algorithm expects the server to know beforehand the number of potential malicious clients taking part in each FL training round, as this parameter controls the number of updates to be eliminated from the aggregation process. This condition was fulfilled in the experiments, and the Multi-Krum algorithm was empowered with the knowledge of the actual number of attackers who performed the label flipping in their local data. As this parameter is set for the whole learning process and does not adapt depending on the subset of clients picked for a certain FL iteration, it was decided to eliminate the client-picking step from the experiments, making all 15 clients always participate in each training round. In such cases, the Multi-Krum algorithm always has a chance to eliminate all malicious clients and for weighting algorithms (FoolsGold and AdFL), the aggregation weights for each client can be tracked throughout the whole training process uninterruptedly.

During all experiments, the aggregation weights for the client are tracked for each epoch reported by re-weighting algorithms (AdFL and FoolsGold), while for the Multi-Krum algorithm, the aggregation weights are assigned equally for the client updates chosen for aggregation, e.g. if there are m chosen clients for aggregation, each of the clients receives aggregation weight of $\frac{1}{m}$.

Each experiment was analyzed with respect to the mean accuracy reached by the algorithm in a given scenario and the mean aggregation weights that each algorithm gave to the malicious/benign clients. Mean values are calculated across all 5 repetitions performed for each unique algorithm, dataset, and attack type.

The first dataset analyzed was MNIST. The results for the untargeted attack for model accuracy and mean benign/malicious client aggregation weights are shown in Figure 2.

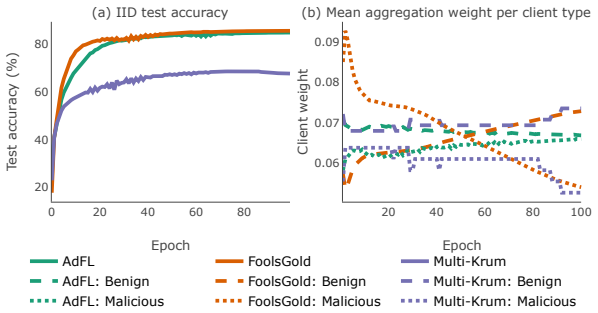

Figure 3. MNIST targeted attack on the common label

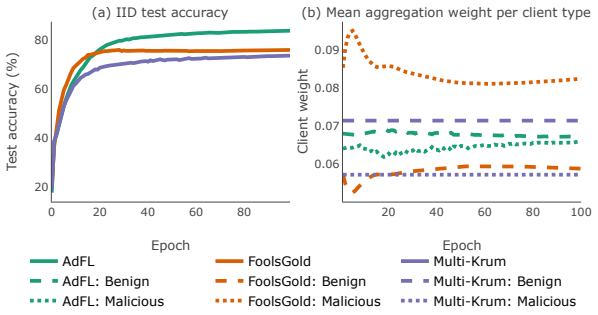

Figure 4. MNIST targeted attack on uncommon label

Here it is visible, that the AdFL algorithm scores first in accuracy, while the FoolsGold algorithm reaches far lower accuracy (68% and 42.5% for the AdFL and FoolsGold algorithms respectively). Among the presented algorithms, Multi-Krum gives the highest weight to benign clients, however, at the beginning of training, for the first 10 epochs the weight of malicious clients was higher. The FoolsGold algorithm continuously favors malicious clients, while the AdFL algorithm manages to assign higher weights to benign clients.

The comparison of three algorithms for the targeted attack on the common label on the MNIST dataset is presented in Figure 3.

It can be seen, that both AdFL and FoolsGold algorithms manage to reach accuracy around 85%, while Multi-Krum scores significantly lower, despite properly favoring benign clients during model aggregation. Here, FoolsGold shows a notable change in the weighting dynamic, with malicious clients first scoring highest and then, after epoch 53, switching with benign clients.

The comparison of three algorithms for the targeted attack on the uncommon label on the MNIST dataset is presented in Figure 4.

The plot illustrates the AdFL algorithm reaching higher accuracy, and it can be seen, that the weight of the only malicious client was also different from the benign, although the preference towards benign clients is smaller than those of the Multi-Krum algorithm. What is more, in the presented scenario, the mechanism of the FoolsGold algorithm favoring the unique updates can be seen in action, assigning the highest weights to the malicious client.

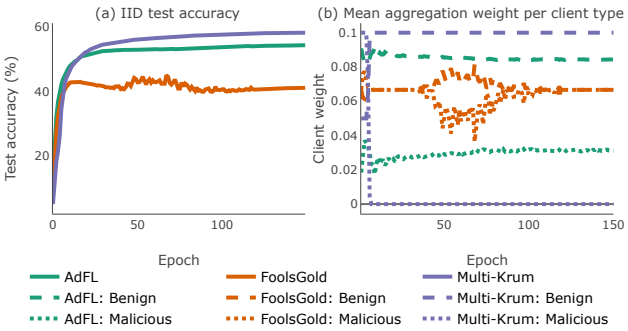

Figure 5. EMNIST untargeted attack

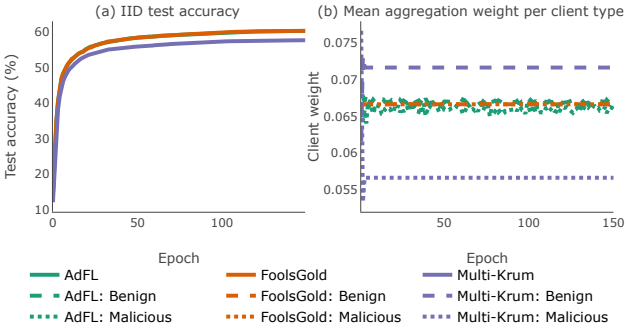

Figure 6. EMNIST targeted attack on the common label

The comparison of three algorithms for the untargeted attack on the EMNIST dataset is illustrated in Figure 5.

Both Multi-Krum and AdFL algorithms successfully identify malicious clients, while for the FoolsGold algorithm, it takes time to start correctly re-weighting clients and the positive benign client weighting dynamic vanishes as the training progresses after epoch 100. Still, Multi-Krum scores higher in both accuracy and benign clients' weight as it manages to successfully filter out all malicious clients, while the AdFL algorithm only lowers their weights.

The targeted attack on the common label on the EMNIST dataset is presented in Figure 6.

It is seen, that the Multi-Krum algorithm manages to filter out some of the malicious clients, but scores lower in accuracy, while the FoolsGold algorithm is not able to reliably identify the malicious clients. However, together with the AdFL algorithm it reaches an accuracy of 60%, while the Multi-Krum algorithm – only 57.5%. The AdFL algorithm shows a slight preference for benign clients, with both malicious and benign client weights changing in the narrow range. Therefore, the mean and standard deviation values were computed for the difference (not absolute) between aggregated weights assigned to benign and malicious clients and are presented in Table 2.

The targeted attack on the uncommon label on the EMNIST dataset is presented in Figure 7.

Table 2. Mean and standard deviation of the difference between aggregation weights for targeted attack on the common label for the EMNIST dataset

Algorithm	Mean	SD
AdFL	0.0010	0.0015
FoolsGold	0.0000	0.0000
Multi-Krum	0.0148	0.0125

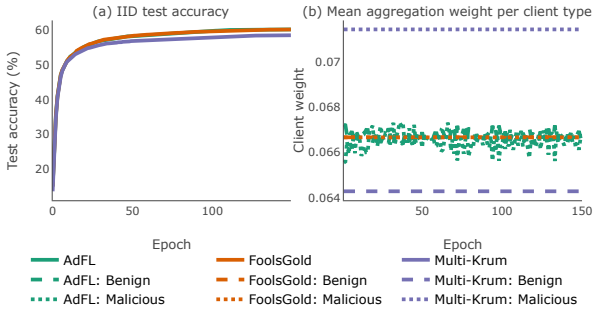

Figure 7. EMNIST targeted attack on the uncommon label

Table 3. Mean and standard deviation of the difference between aggregation weights for targeted attack on the uncommon label for the EMNIST datase

Algorithm	Mean	SD
AdFL	0.0002	0.0015
FoolsGold	0.0000	0.0000
Multi-Krum	-0.0071	0.0000

This scenario was the most complex for all three algorithms to deal with. As the classification task, in this case, consists of 62 unique labels, and one of the rarest labels was poisoned by only one adversary, detecting the malicious client was not trivial. Therefore, it can be seen that the Multi-Krum algorithm failed to filter out the malicious client, while the FoolsGold algorithm, as in the scenario with the targeted attack on the common label, fails to perform re-weighting at all. As for the AdFL algorithm, the fluctuations of weights can be observed, however, the weights are changing within a small range (Table 3 states the mean and standard deviation for the difference between the aggregation weights of benign and malicious clients), moreover, the benign clients are being continuously favored only after epoch 70.

The comparison of three algorithms for the untargeted attack on the CIFAR-10 dataset is presented in Figure 8.

In the observed scenario, the Multi-Krum algorithm manages to correctly identify the malicious clients and scores first in accuracy, while both the FoolsGold and the AdFL algorithms show similar lower accuracy of 32% compared to 66% for the Multi-Krum algorithm. However, despite lower accuracy, the AdFL algorithm still properly re-weights clients, favoring benign clients from the beginning of the training, when the FoolsGold algorithm prefers malicious clients.

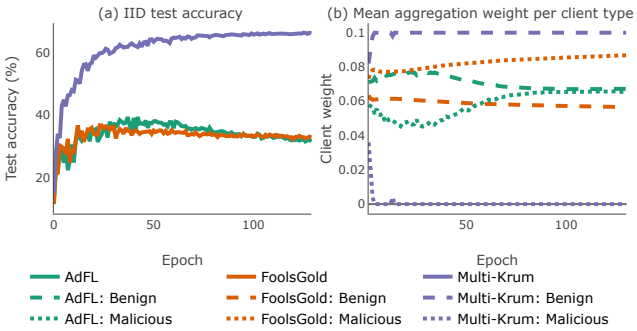

Figure 8. CIFAR-10 untargeted attack

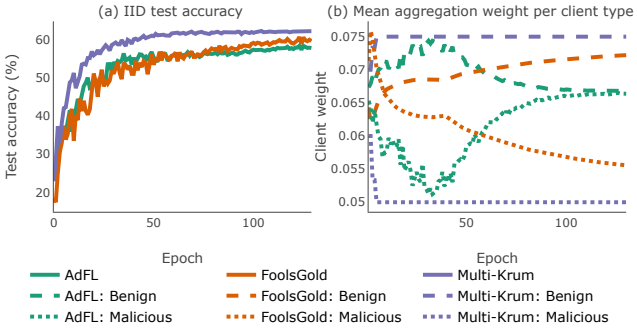

Figure 9. CIFAR-10 targeted attack on the common label

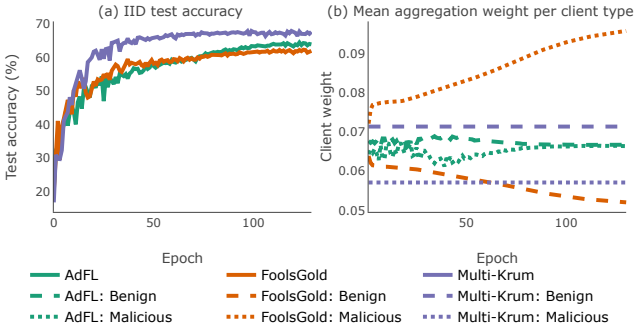

Figure 10. CIFAR-10 targeted attack on the uncommon label

The comparison of three algorithms for the targeted attack on the common label on the CIFAR-10 dataset is presented in Figure 9.

In the presented case, it can be observed, that all three algorithms manage to correctly favor benign clients and reach the accuracy of 62%, 59.5%, and 58% respectively for the Multi-Krum, FoolsGold, and AdFL algorithms.

The comparison of three algorithms for the targeted attack on the uncommon label on the CIFAR-10 dataset is presented in Figure 10.

Here, the Multi-Krum algorithm manages to filter out the malicious client in some experiments, while the AdFL algorithm only decreases the weight of the malicious client until epoch 90 and the FoolsGold algorithm continuously favors the malicious client.

Table 4. Mean ASR(%) at the final training epoch for a targeted attack on the common label

Algorithm	Dataset		
	MNIST	EMNIST	CIFAR-10
AdFL	3	11	93
Fools Gold	1	12	63
Multi-Krum	2	10	0.25

Table 5. Mean ASR(%) at the final training epoch for a targeted attack on uncommon label

Algorithm	Dataset		
	MNIST	EMNIST	CIFAR-10
AdFL	9	2	25
Fools Gold	84	3	58
Multi-Krum	12	3	8

For targeted attacks, ASRs for each of the algorithms were also tracked with respect to the hold-out test dataset according to Equation 6. Thus, the mean ASR on common and uncommon labels at the end of the training for the MNIST, EMNIST, and CIFAR-10 datasets are shown in Table 4 and Table 5, respectively.

There is a notable difference in the ASR between the Multi-Krum algorithm and the AdFL and FoolsGold algorithms when it comes to the CIFAR-10 dataset. For targeted attacks on both common and uncommon labels, the Multi-Krum algorithm reaches significantly lower ASR (under 10%), while the other two algorithms reach ASR between 25 and 93%. For the EMNIST dataset, all three algorithms show similar ASR regardless of the attack target. However, for the MNIST dataset, the targeted attack on the uncommon label shows an exceptionally high ASR for the FoolsGold.

The dynamic of the ASR change for the attack on the common label is presented in Figure 11.

Here, the differences in the range of ASR values is further specified with the dynamic throughout all epochs, highlighting that for the MNIST dataset, the end of the training aligned with the lowest ASR, while for both EMNIST and CIFAR-10 datasets, the end of the training yielded a higher ASR, with the exception of Multi-Krum algorithm on the CIFAR-10 dataset.

The changes in ASRs during the scenarios with the targeted attack on the uncommon label are presented in Figure 12.

It is clearly visible how the FoolsGold algorithm's tendency to favor unique updates impacts the ASR on all three datasets. Another finding here illustrates that although the EMNIST dataset has a relatively low absolute ASR, the ASR grows dynamically as training progresses, showcasing how all three algorithms fail to defend the model from targeted attacks regardless of the target label.

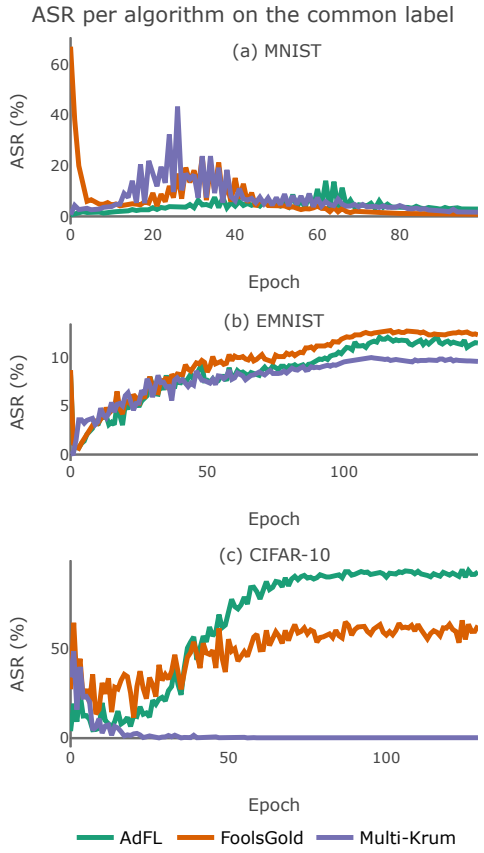

Figure 11. ASR on the common label per algorithm per dataset

To sum up, the experiments show that the label skew combined with different label-flipping attacks presents a challenging task for all three algorithms when compared to one another. However, it can be seen, that the aggregation weights given by the AdFL algorithm to malicious and benign clients differ in all experiments conducted, with benign clients being favored by the algorithm. Still, the range of the difference between these weights varies depending on the dataset and attack type. Moreover, experiments on the MNIST and CIFAR-10 datasets show that the weights of malicious and benign clients, reported by the AdFL algorithm, tend to become even as the training progresses, as more model aggregation happens, and the accuracy of the global model increases.

Therefore, the synthetic samples generated by both malicious and benign clients become similar and receive similar coherence scores. Another observation highlights that the ASRs for algorithms differ depending on the dataset and attack type, with the Multi-Krum having the most stable mean ASR across all datasets and attack targets, while both the AdFL and FoolsGold algorithms showed high ASRs for the CIFAR-10 dataset, while EMNIST dataset was the most challenging dataset to protect from the targeted attack regardless of the target label being common or uncommon among federated clients. Still, the AdFL algorithm showed an ASR comparable with the selected defense algorithms, despite not being designed with protection from data poisoning attacks in mind.

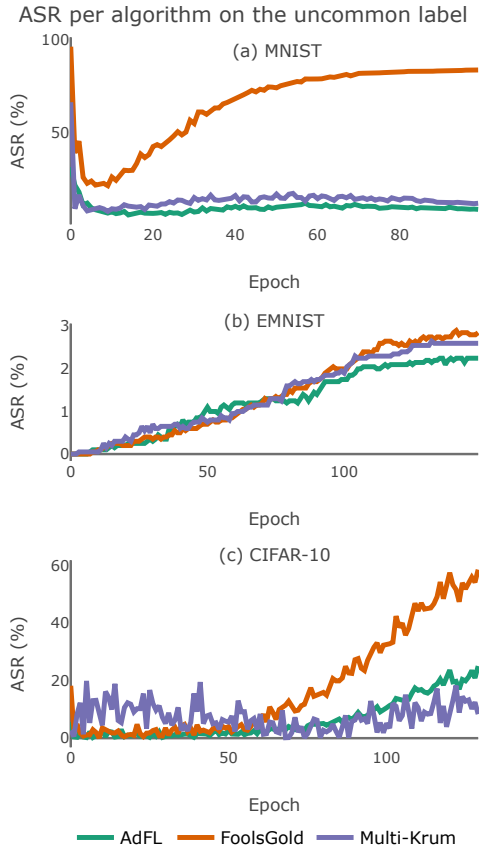

Figure 12. ASR on the uncommon label per algorithm per dataset

Table 6. Wilcoxon signed-rank test p-value and significance

Dataset	Attack type	P-value
All	All	1.08×10^{-12}
All	Untargeted	6.1×10^{-5}
All	Targeted (common)	6.1×10^{-5}
All	Targeted (uncommon)	0.00061
MNIST	All	6.1×10^{-5}
EMNIST	All	0.00116
CIFAR-10	All	6.1×10^{-5}

The significance of the aggregation weights difference provided by the AdFL algorithm was additionally assessed with the help of the Wilcoxon signed-rank test [43] based on the mean aggregation weights for malicious/benign clients inside each trial, i.e, for each repetition inside the experiment, the mean aggregation weights were calculated for malicious and benign clients across all epochs and used as a pair for the Wilcoxon signed-rank test. A separate test was performed for all trials performed, for each dataset (regardless of the attack type), and for each attack type (regardless of the dataset). The results are presented in Table 6.

The Wilcoxon signed-rank test revealed a significant difference across all considered combinations of datasets and attack types. However, it is seen, that the difference between malicious and benign clients aggregation weights for the EMNIST dataset and for targeted attacks on uncommon labels is less significant than in the rest of the cases, which highlights that for the AdFL algorithm, it is harder to operate in presence of attacks aimed at uncommon labels and within the classification tasks with a bigger set of unique labels.

7. Conclusion

In this work, the applicability of synthetic adversarial samples was explored in the context of non-IID data and data poisoning attacks. Three types of attacks were performed on three benchmark image classification datasets and the results were compared with respect to global model accuracy, the ability of the algorithms to distinguish malicious clients from benign, and the ASR of the targeted attacks.

The results revealed that utilizing adversarial data on the server side during FL training can successfully re-weight malicious clients and give them less importance during model aggregation for all untargeted and targeted attacks. However, the magnitude of the weight difference is not sufficient to fully mitigate the damage performed by the malicious clients in comparison with the security methods specifically crafted to battle data poisoning attacks of certain types. Still, as the AdFL algorithm showed the ability to favor benign clients over malicious ones during the experiments conducted, future research can further improve the results by ensuring a more powerful weighting scheme to promote a greater influence of the AdFL coherence measure step on the model aggregation and verify the AdFL algorithm performance in more populated FL scenarios that include client picking and introduce more diverse data distributions.

AUTHOR

Anastasiya Danilenka* – Faculty of Mathematics and Information Science, Warsaw University of Technology,
Koszykowa 75, 00-662 Warsaw, Poland, e-mail: anastasiya.danilenka.dokt@pw.edu.pl, www: orcid.org/0000-0002-3080-0303.

*Corresponding author

ACKNOWLEDGEMENTS

This work was supported by the Centre for Priority Research Area Artificial Intelligence and Robotics of Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme and by the Laboratory of Bioinformatics and Computational Genomics and the High-Performance Computing Center of the Faculty of Mathematics and Information Science at Warsaw University of Technology.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. "Communication-efficient learning of deep networks from decentralized data", 2017.

[2] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. "On the convergence of fedavg on non-iid data", 2020.

[3] T.-M. H. Hsu, H. Qi, and M. Brown. "Measuring the effects of non-identical data distribution for federated visual classification", 2019.

[4] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning", *Future Generation Computer Systems*, vol. 135, 2022, 244–258, https://doi.org/10.1016/j.future.2022.05.003.

[5] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey", *Applied Sciences*, vol. 12, no. 19, 2022.

[6] P. Erbil and M. E. Gursoy, "Detection and mitigation of targeted data poisoning attacks in federated learning". In: *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2022, 1–8, 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927914.

[7] A. Danilenka, "Mitigating the effects of non-iid data in federated learning with a self-adversarial balancing method", *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2023, 925–930.

[8] Y. LeCun and C. Cortes, "MNIST handwritten digit database", 2010.

[9] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. "Emnist: an extension of mnist to handwritten letters", 2017.

[10] A. Krizhevsky. "Learning multiple layers of features from tiny images", 2009.

[11] 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*, New York, NY, USA, 2022, 2545–2555, 10.1145/3534678.3539231.

[12] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means". In: *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, 2021, 551–559, 10.1109/DSA52907.2021.00081.

[13] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent". In: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[14] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning". In: *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, 233–239, 10.1109/ICPADS47876.2019.00042.

[15] X. Cao, M. Fang, J. Liu, and N. Z. Gong. "Fltrust: Byzantine-robust federated learning via trust bootstrapping", 2022.

[16] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. "Byzantine-robust distributed learning: Towards optimal statistical rates", 2021.

[17] C. Xie, O. Koyejo, and I. Gupta. "Generalized byzantine-tolerant sgd", 2018.

[18] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings". In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, San Sebastian, 2020, 301–316.

[19] Y. Xie, W. Zhang, R. Pi, F. Wu, Q. Chen, X. Xie, and S. Kim. "Robust federated learning against both data heterogeneity and poisoning attack via aggregation optimization", 2022.

[20] S. Han, S. Park, F. Wu, S. Kim, B. Zhu, X. Xie, and M. Cha, "Towards attack-tolerant federated learning via critical parameter analysis". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, 4999–5008.

[21] S. Park, S. Han, F. Wu, S. Kim, B. Zhu, X. Xie, and M. Cha, "Feddefender: Client-side attack-tolerant federated learning". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2023, 1850–1861, 10.1145/3580305.3599346.

[22] C. Chen, Y. Liu, X. Ma, and L. Lyu. "Calfat: Calibrated federated adversarial training with label skewness", 2023.

[23] G. Zizzo, A. Rawat, M. Sinn, and B. Buesser. "Fat: Federated adversarial training", 2020.

[24] Z. Li, J. Shao, Y. Mao, J. H. Wang, and J. Zhang. "Federated learning with gan-based data synthesis for non-iid clients", 2022.

[25] Y. Lu, P. Qian, G. Huang, and H. Wang. "Personalized federated learning on long-tailed data via adversarial feature augmentation", 2023.

[26] X. Li, Z. Song, and J. Yang. "Federated adversarial learning: A framework with convergence analysis", 2022.

[27] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. "Intriguing properties of neural networks", 2014.

[28] O. Suciu, R. Marginean, Y. Kaya, H. D. III, and T. Dumitras, "When does machine learning FAIL? generalized transferability for evasion and poisoning attacks". In: *27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, 2018, 1299–1316.

[29] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples", 2015.

[30] A. Kurakin, I. Goodfellow, and S. Bengio. "Adversarial examples in the physical world", 2017.

[31] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. "Boosting adversarial attacks with momentum", 2018.

[32] G. Xia, J. Chen, C. Yu, and J. Ma, "Poisoning attacks in federated learning: A survey", *IEEE Access*, vol. 11, 2023, 10708–10722, 10.1109/ACCESS.2023.3238823.

[33] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks". In: *Neural Information Processing Systems*, 2018.

[34] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on federated learning", *ArXiv*, vol. abs/2108.10241, 2021.

[35] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines". In: *Proceedings of the 20th European Conference on Artificial Intelligence*, NLD, 2012, 870–875.

[36] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu. "Data poisoning attacks against federated learning systems", 2020.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc., 2019.

[38] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch". In: *Proceedings of the 18th ACM International Conference on Multimedia*, New York, NY, USA, 2010, 1485–1488, 10.1145/1873951.1874254.

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, 1998, 2278–2324, 10.1109/5.726791.

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks", 2019.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*, 2009, 248–255.

[42] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses", *IEEE Transactions on Neural Networks and Learning Systems*, 2022, 1–21, 10.1109/TNNLS.2022.3216981.

[43] F. Wilcoxon. *Individual comparisons by ranking methods*, 196–202. Springer, 1992.