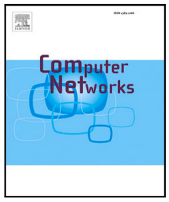
Computer Networks 254 (2024) 110798

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



An Adversarial Machine Learning Based Approach for Privacy Preserving Face Recognition in Distributed Smart City Surveillance

Farah Wahida [a,*], M.A.P. Chamikara [b], Ibrahim Khalil [a], Mohammed Atiquzzaman [c]

[a] School of Computing Technologies, RMIT University, Melbourne, Australia
[b] CSIRO's Data61, Melbourne, Australia
[c] School of Computer Science, University of Oklahoma, Norman, USA

ARTICLE INFO

Keywords:
Federated Learning
Smart cities surveillance
Data perturbation
IoT data privacy
Privacy-preserving Federated Learning

ABSTRACT

Smart cities rely heavily on surveillance cameras for urban management and security. However, the extensive use of these cameras also raises significant concerns regarding data privacy. Unauthorized access to facial data captured by these cameras and the potential for misuse of this data poses serious threats to individuals' privacy. Current privacy preservation solutions often compromise data usability with noise application-based approaches and vulnerable centralized data handling settings. To address these privacy challenges, we propose a novel approach that combines Adversarial Machine Learning (AML) with Federated Learning (FL). Our approach involves the use of a noise generator that perturbs surveillance data right from the source before they leave the surveillance cameras. By exclusively training the Federated Learning model on these perturbed samples, we ensure that sensitive biometric features are not shared with centralized servers. Instead, such data remains on local devices (e.g., cameras), thereby ensuring that data privacy is maintained. We performed a thorough real-world evaluation of the proposed method and achieved an accuracy of around 99.95% in standard machine learning settings. In distributed settings, we achieved an accuracy of around 96.24% using federated learning, demonstrating the practicality and effectiveness of the proposed solution.[1]

1. Introduction

The widespread use of surveillance cameras, especially in smart cities, has raised serious privacy concerns [1,2]. Cameras with face recognition technology can collect sensitive information about people's daily activities and movements, posing a potential threat to privacy. Recent incidents, such as major Chinese surveillance providers exposing personal data, including photos, addresses, and ages, highlight these risks [3,4]. For example, Facebook received a hefty fine of $550 million due to collecting unauthorized facial data [5]. This highlights the urgent need to tackle privacy concerns in the age of increased surveillance. Smart cities leverage cutting-edge technologies and surveillance cameras to improve public safety, traffic management, and urban planning. However, the significant use of surveillance data raises significant concerns regarding privacy and security [2]. The use of cameras in urban management provides significant advantages, but it also poses a risk to individual privacy [2]. Achieving the right balance between utilizing surveillance data for the common good and upholding privacy is a crucial challenge of this era.

Handling vast and diverse data sources in modern industries can be a challenge for traditional machine learning approaches, which collect and process all data on a central server. Moreover, this centralized data collection approach raises serious privacy concerns [6]. To address these challenges, federated learning (FL) has emerged as a promising approach to machine learning. FL allows multiple clients to work on their data locally and train their own machine learning models, after which they share only the model parameters with a central server. The server aggregates this information and creates a global model without sharing raw data with other parties involved in the training process. However, it is important to note that even model-sharing can reveal sensitive information under certain conditions [7]. Therefore, when working with sensitive data like biometric images, health records, or financial information, FL can still pose privacy risks if strong privacy protection measures are not in place.

Various privacy techniques have been developed to address the privacy issues in Federated Learning (FL) [7,8]. These techniques include cryptographic methods and data perturbation (through noise

* Corresponding author.
E-mail addresses: farah.wahida@student.rmit.edu.au (F. Wahida), chamikara.mahawagaarachchige@data61.csiro.au (M.A.P. Chamikara), ibrahim.khalil@rmit.edu.au (I. Khalil), atiq@ou.edu (M. Atiquzzaman).

[1] The code is available at: https://github.com/farah-wahida/Privacy-Preserving-Face-Recognition-in-Distributed-Smart-City-Surveillance.

https://doi.org/10.1016/j.comnet.2024.110798
Received 20 October 2023; Received in revised form 19 July 2024; Accepted 9 September 2024
Available online 11 September 2024

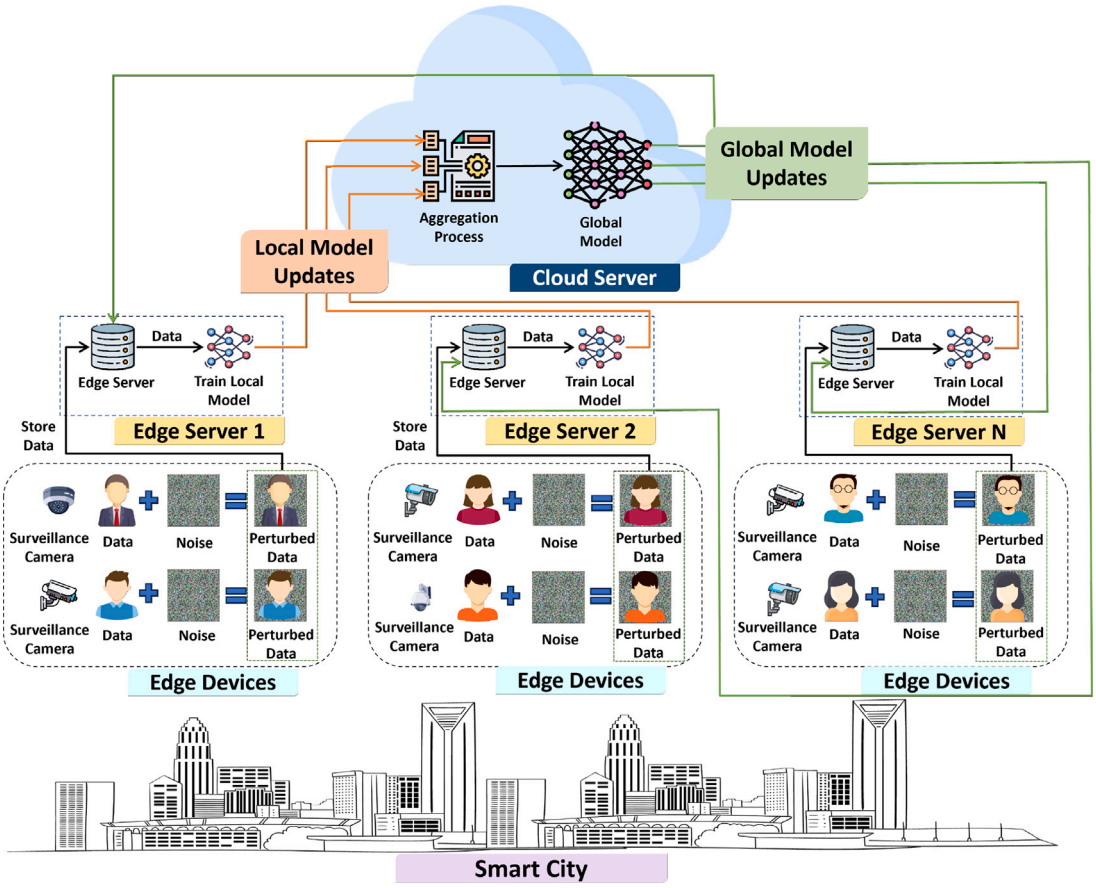


Fig. 1. Privacy-preserving federated learning framework in smart city surveillance: Surveillance cameras capture facial images and introduce noise. Local data from multiple cameras are collected on edge servers. These edge servers host client models within the FL framework, where they communicate with a central server during the FL model training process.

addition or randomization). Two commonly used cryptographic methods in FL are Secure Multi-Party Computation (SMC) and Homomorphic Encryption. However, they have high computational and communication requirements, which can slow down FL significantly [8,9]. These methods are designed to ensure honest computation while restricting the ability to learn more. Data perturbation methods, particularly differentially private approaches, are preferred due to their strong privacy guarantees and efficiency [7,10].

Differential privacy can be categorized into global differential privacy (GDP) and local differential privacy (LDP). In GDP, a trusted curator adds calibrated noise to raw data [11,12]. In LDP, data owners perturb their data before sharing them with a third party [11]. LDP offers higher privacy levels because it introduces more noise compared to global differential privacy [6]. While most existing FL methods are based on global differential privacy, their dependence on a trusted party makes them less practical. On the other hand, local differential privacy provides a more practical solution for handling distributed clients in FL. Some previous approaches apply local differential privacy by adding noise or randomizing the model parameters of local models [13]. However, existing local differential privacy methods have not undergone rigorous testing on more complex datasets [7,14].

Fig. 1 provides a generic view of the proposed privacy-preserving federated learning framework tailored for smart cities. This image shows surveillance cameras (edge devices) placed throughout a city to gather data. Each edge device adds noise locally to protect individual privacy. Edge devices are connected to an edge server, and the global model is securely hosted in a cloud server. These edge servers function as clients within the FL setting. Each server independently shares its local updates, derived from training local models on perturbed data. These local updates are then incorporated into the global model following the conventional FL paradigm. We carried out comprehensive experiments under both centralized and FL settings. In the centralized setting, we were able to achieve a testing accuracy of 99.95%. In the FL setting, we were able to achieve a testing accuracy of 96.24%.

The rest of this article is organized as follows. The underlying concepts used in the proposed framework are presented in Section 2, followed by the proposed methodologies in Section 3. Section 4 presents the results and their discussions, while Section 5 explores related work. Finally, Section 6 presents a conclusion and highlights the broader implications of the work.

2. The background

This section provides brief descriptions of the fundamentals utilized in the work.

2.1. Federated learning

Federated Learning (FL) [15] is a collaborative approach that involves N distributed parties connected to a central server. All parties agree to train local deep neural network (DNN) models with the same setup. The FL process begins with the central server initializing the model parameters and sending them to the clients for their model setup. Clients then train their local models independently using their local data for several local epochs. Afterwards, they share the updated model parameters with the server. Using methods such as federated averaging, the server aggregates these model parameters from all clients to create the federated model.

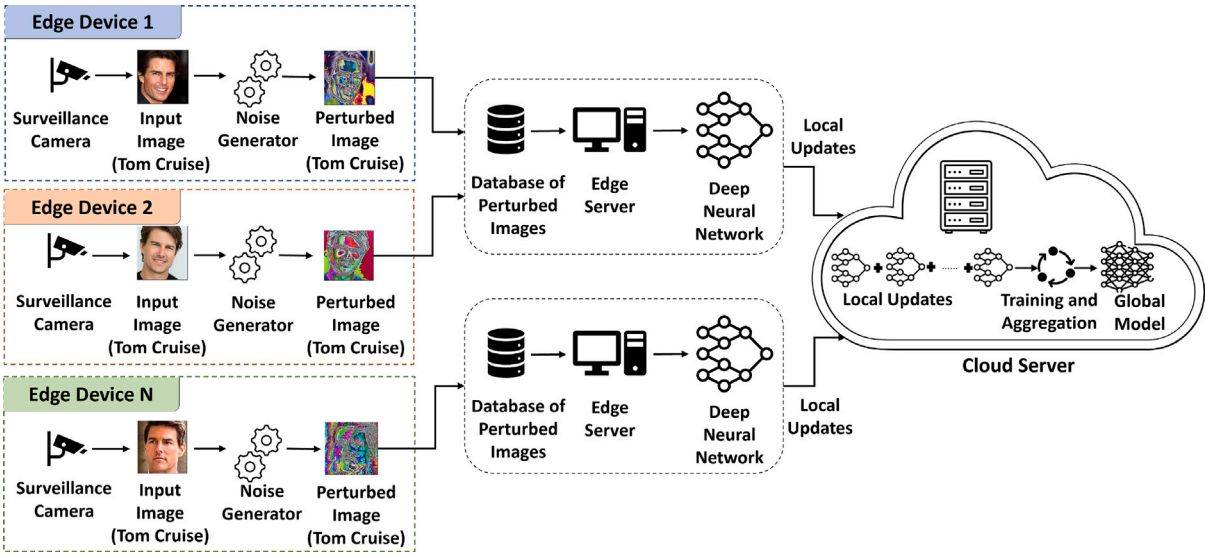


Fig. 2. Workflow in privacy-preserving federated learning for surveillance cameras: Surveillance cameras capture input images, which are then forwarded to the noise generator. This generator perturbs the images. Edge servers train local models within the federated learning setup.

Let us consider a set of clients, denoted as $C = \{C_1, C_2, \ldots, C_n\}$, where each C_i represents an individual edge device. The overarching objective is to train a global model Θ by leveraging the local datasets held by each client. Each client, C_i, maintains its local dataset $\mathcal{D}_i$, consisting of image-label pairs, $\mathcal{D}_i = \{(x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \ldots, (x_{i_m}, y_{i_m})\}$. The training process continues iteratively as follows:

1. **Local model updates**: At each iteration t, each client C_i computes a local update $\Delta\Theta_i^t$ on its local dataset $\mathcal{D}_i$ by minimizing a loss function $L(\Theta, \mathcal{D}_i)$, where Θ is the current global model. Mathematically, this is represented as:

$$\Delta\Theta_i^t = \arg min_{\Delta\Theta} L(\Theta + \Delta\Theta, \mathcal{D}_i)$$

2. **Transmit updates**: After computing their local updates, clients transmit only the model update $\Delta\Theta_i^t$ to a central aggregation node.
3. **Global model update**: The central server collects the model updates from all clients and aggregates them to update the global model Θ. This aggregation is typically performed using techniques such as federated averaging:

$$\Theta^{t+1} = \sum_{i=1}^{n} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \cdot (\Theta^t + \Delta\Theta_i^t)$$

4. **Iterate**: Steps 1 to 3 are repeated for a fixed number of iterations or until the convergence criteria are met.

2.2. Adversarial machine learning

Adversarial machine learning is a specialized area of Artificial Intelligence (AI) that focuses on making AI systems more resilient and secure, especially when faced with adversarial attacks [16]. These attacks involve manipulating AI systems by making subtle changes to their input data. DeepFool is a well-known adversarial attack that works by creating precise perturbations in input data that are specifically designed to deceive AI models [17]. These perturbations exploit weaknesses in the decision-making process of AI models, resulting in incorrect or unexpected outputs. DeepFool is a widely used technique in the field of adversarial machine learning and has become a fundamental method for generating adversarial examples.

The DeepFool adversarial attack can be described as follows.

$$X_{adversarial} = X_{original} + \delta$$

1. $X_{original}$ corresponds to the unaltered, genuine data.
2. $X_{adversarial}$ represents the data with these subtle yet calculated modifications.
3. δ denotes the small, computed adjustment applied to the original data.

The aim of DeepFool is to identify the smallest possible modification δ to the input data X_{original} that can cause misclassification by the AI model. Essentially, DeepFool tries to find out how to change the input data in the most undetectable way possible to deceive the AI model. If an AI system encounters such modified data, it can lead to errors, which can be especially problematic in applications such as self-driving cars, security systems, or medical diagnoses.

3. Proposed work

The proposed approach utilizes adversarial noise generation techniques to enable privacy-preserving collaborative training of a machine learning model protecting the confidentiality of sensitive IoT data obtained from smart city surveillance cameras. The methodology is primarily aimed at balancing between privacy and utility to ensure the privacy of data while also maximizing their utility.

Fig. 2 illustrates the workflow of the proposed solution. The proposed approach involves a network of edge devices (surveillance cameras), each connected to an edge server within the federated learning setting. These edge servers are linked to a central server and manage local model training and parameter updates. The Noise Generator creates privacy-preserving examples by perturbing human face images. Each edge server collects the perturbed images and maintains a local learning setup that hosts machine learning models tailored to the locally collected surveillance data specifications (e.g., resolution and the number of channels). Consequently, the performance of the FL setup relies entirely on the perturbed images.

The following sections provide a detailed description of the proposed model's overall process.

3.1. Image perturbation

The proposed approach integrates an Adversarial Noise Generator to perturb input with optimal noise for privacy and utility. Fig. 4 shows the main flow of the proposed work and Algorithm 1 provides the main steps of the proposed algorithm.

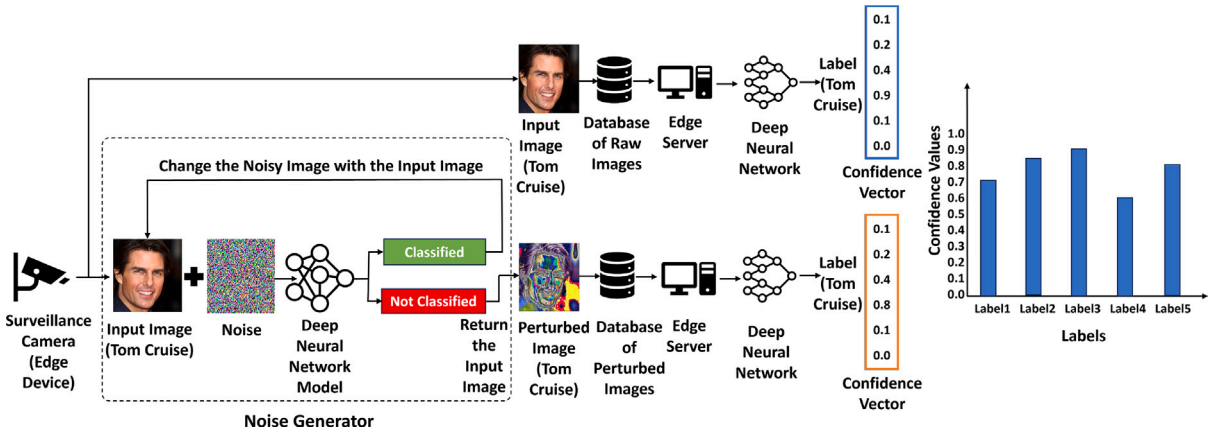


Fig. 3. Confidence assessment of raw image and perturbed image: In the conventional approach (first pathway), an input image from a surveillance camera is stored in the raw image dataset and sent to edge servers. Then, an embedded deep neural network predicts the image's label by generating a confidence vector. However, in the proposed work (second pathway), a noise generator is connected to each surveillance camera. Here, an input image undergoes iterative transformations, introducing noise until the deep neural network misclassifies it. The last iteration before the network misclassifies the image is the one where the network correctly classifies the image with the highest level of perturbation possible. This perturbed image is then sent to the edge server and processed by a trained deep neural network to predict its label. This contributes to producing an updated confidence vector.

3.1.1. Noise generator

The main objective of the noise generator (refer to Fig. 3) is to introduce maximum perceptible changes (as opposed to adversarial noise generation) to the original image while still ensuring correct classification. The perturbed image undergoes classification by the neural network model to determine if it is classified correctly. This process repeats until the model misclassifies the image, at which point the algorithm stops the perturbation process. This image, obtained just before the misclassification occurs, is saved as the output of the optimally perturbed image with maximum perturbation (δ) while preserving minimal features for correct classification.

The amount of noise introduced is random from image to image, even if they belong to the same class (refer to Fig. 7). This adaptability ensures that privacy is preserved without sacrificing utility, as the perturbed examples accurately represent the original data.

At each iteration, the perturbation is updated as follows:

$$\delta_{new} = \delta + \text{sign}(\nabla_x L(\mathcal{W}(x_i+\delta;\theta), y_i)) \times \frac{|(\nabla_x L(\mathcal{W}(x_i+\delta;\theta), y_i))|}{\|(\nabla_x L(\mathcal{W}(x_i+\delta;\theta), y_i))\|_2} \times |L(\mathcal{W}(x_i+\delta;\theta), y_i)| \quad (1)$$

where $\nabla_x L(.)$ denotes the gradient of the loss with respect to the input image.

Here,

- δ_{new}: This is the updated perturbation that is added to the original input to create an obfuscated image. It maximizes the loss while staying within bounds.
- δ: This is the original perturbation applied to the input image.
- $\text{sign}(\nabla_x L(\mathcal{W}(x_i+\delta;\theta), y_i))$: This term determines the direction in which to update the perturbation. It is the sign of the gradient of the loss function with respect to the perturbed input. This helps the algorithm to move the perturbation in a direction that increases the loss.
- $\frac{|(\nabla_x L(\mathcal{W}(x_i+\delta;\theta), y_i))|}{\|(\nabla_x L(\mathcal{W}(x_i+\delta;\theta), y_i))\|_2}$: This term normalizes the gradient to have a unit L_2 norm (Euclidean norm). It ensures that the perturbation's magnitude does not grow too large.
- $|L(\mathcal{W}(x_i+\delta;\theta), y_i)|$: This term represents the magnitude of the loss with respect to the perturbed input. It indicates how much the loss can be increased by updating the perturbation.

In essence, the algorithm calculates an updated perturbation that moves the input image in a direction that increases the loss, perturbs its features, and encourages privacy preservation while also ensuring that the perturbation's magnitude remains bounded. This process iteratively modifies the perturbation to generate a perturbed example. Fig. 6 provides a flowchart illustrating the key steps of the proposed method.

3.1.2. Convergence point

The Noise Generator achieves the convergence point (i.e., maximum adversarial perturbation while maintaining correct classification) for each image as follows.

Here,

- x_i: Original input image
- y_i: True label of the original image
- θ: Neural Network Parameters
- $L(\mathcal{W}(x_i+\delta;\theta), y_i)$: Loss function indicating the discrepancy between the predicted label and the true label

The goal of the Noise Generator is to find a perturbation δ such that:

$$\delta^* = argmax_\delta L(\mathcal{W}(x_i+\delta;\theta), y_i) \quad (2)$$

Subject to constraints:

- $\|\delta^*\|_p \leq \varepsilon$: Bounded perturbation magnitude
- $\mathcal{W}(x_i+\delta^*;\theta) \neq y_i$: Misclassification

The convergence point, in this case, refers to the perturbation δ^* that satisfies these conditions. The Noise Generator iteratively updates δ to get closer to this convergence point. The algorithm stops either when the misclassification condition is met, or a predefined iteration limit is reached.

3.2. Privacy assessment (γ-AdvNoise privacy)

Let D be the original dataset and D^p be the perturbed version generated through the proposed perturbation mechanism, M. The post-processing algorithm FL is applied to both datasets to yield the outputs (class labels) in $\{Y\}$. The perturbation Convergence Property states (refer to Section 3.1.1 and Algorithm 1) :

$$P(M(D) \in \{Y\}) \approx P(M(D^p) \in \{Y\}) \quad (3)$$

This property signifies the convergence of perturbation guarantees, ensuring that M's output probabilities are approximately equal for both the original and perturbed datasets, given that the maximum

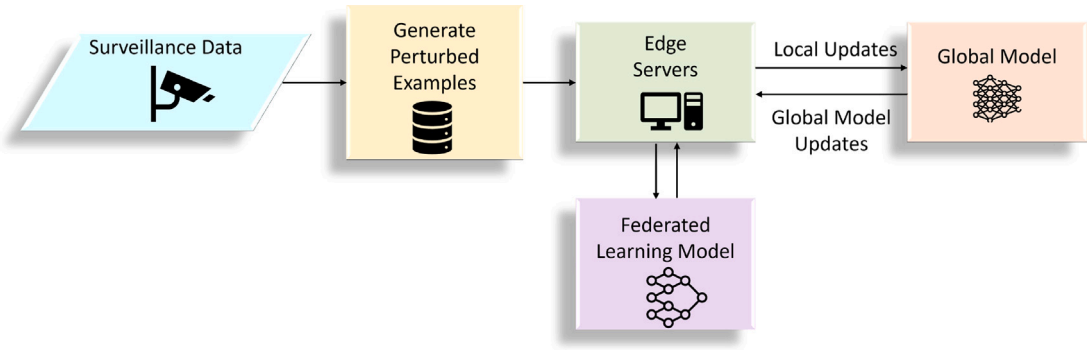


Fig. 4. Flowchart of the main flow of the proposed work.

Fig. 5. Robustness in image recognition: This figure displays four distinct images of Tom Cruise, showcasing variations in appearance. These include his depiction in the first and latest "Top Gun" movies, a photograph with sunglasses, and a side profile. The solution has the ability to consistently perform well despite these apparent differences. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

perturbation is at the point of correct classification of a particular input example (refer to Section 3.1.1) and it is upper bounded by ϵ (the bounded perturbation magnitude).

Theorem 1. *For the FL algorithm, let D^p be the perturbed image dataset obtained from D by applying the perturbation algorithm M on D. Then,*

$$|P(FL(D) \in \{Y\}) - P(FL(D^p) \in \{Y\})| \leq \gamma \tag{4}$$

where γ is bounded by $n.\epsilon.\rho$; here, n is the number of samples in D, ϵ is the bounded perturbation magnitude (refer to Section 3.1.2), and ρ is the reduction factor (refer to Algorithm 1).

Proof: Given that ϵ is the bounded perturbation magnitude and for all images in D is $\|\delta^*\|_p \leq \epsilon$, we can assert that,

$$P(FL(D^p) \in \{Y\}) \approx P(M(D^p) \in \{Y\}) \tag{5}$$

Consequently, we have,

$$|P(FL(D) \in \{Y\}) - P(FL(D^p) \in \{Y\})| \leq n.\epsilon.\rho \tag{6}$$

This inequality bounds the difference in the probability of the output class labels between D and D^p, ensuring that FL does not deviate significantly due to the perturbation introduced by M when ρ and ϵ are kept very low for large datasets. We call this guarantee as γ-AdvNoise privacy.

4. Results and discussion

In this section, we present a series of experiments that were carried out to assess the effectiveness of the proposed framework. We delve into the details of the experimental configuration, dataset selection, and the models used in Section 4.1 and Section 4.2, respectively. Section 4.3 and Section 4.4 analyzes the experimental outcomes and the complexity of the proposed algorithm.

4.1. Experimental setup

We utilized an AWS cloud computing service for the experiments. We used a Super Large SageMaker Notebook instance that comes with an NVIDIA T4 Tensor Core GPU. The instance has 64 vCPUs. It also has 256 GB of memory and a 2nd Generation Intel Xeon Scalable Processors (Cascade Lake P-8259L). Besides, it is equipped with a dedicated NVIDIA T4 Tensor Core GPU, which enhances its capabilities for machine learning and deep learning workloads. To strike a balance between computational power and cost-efficiency, we opted for AWS cloud super-computing environment's G4DN series. Specifically, we utilized the N11 GPU Notebook - XXLarge instance, designed for machine learning inference and well-suited for small-scale training tasks. We used Python and PyTorch [18] to build the federated learning environment. To ensure consistency, we maintain the following parameter values throughout all experiments: learning_rate = 0.004, epsilon = 0.03, reduction_factor = 0.95, maximum iteration = 50.

4.2. Datasets and models

We used the Pins dataset [19]. This dataset contains a wide variety of facial images. The images of the dataset have been collected from Pinterest and cropped. There are 105 celebrities and 17 534 faces in it. We chose this dataset because it encompasses complex real-life situations, featuring diverse age groups, various accessories, side profiles, and other factors, making it a highly representative reflection of real-world scenarios (refer to Fig. 5). In addition to the Pins dataset, we evaluated the performance of the proposed approach on the MNIST [20] and CIFAR-10 [21] datasets to further validate the generalizability of the proposed approach across different types of data. The MNIST dataset comprises 70,000 images of handwritten digits, each of size 28 × 28 pixels, spanning digits 0 through 9 (i.e., 10 classes). On the other hand, the CIFAR-10 dataset consists of 60,000 color images, each of size 32 × 32 pixels, categorized into 10 classes representing common objects such as airplanes, automobiles, birds, and cats.

We selected various deep-learning architectures. During the perturbation, we used the AlexNet model [22]. AlexNet is known for its robust feature extraction capabilities and consists of five convolutional layers and three fully connected layers. It employs ReLU activations, local response normalization, and dropout to improve generalization. For the traditional machine learning setup, we opted for the VGG16 model [23]. This architecture is known for its simplicity and effectiveness in image classification tasks. VGG16 comprises 16 layers, including 13 convolutional layers and 3 fully connected layers.

For the privacy evaluation in FL setup, we study the proposed work with ResNet18 [24], GoogLeNet [25], DenseNet [26], MobileNet [27], and ResNeXt [28].

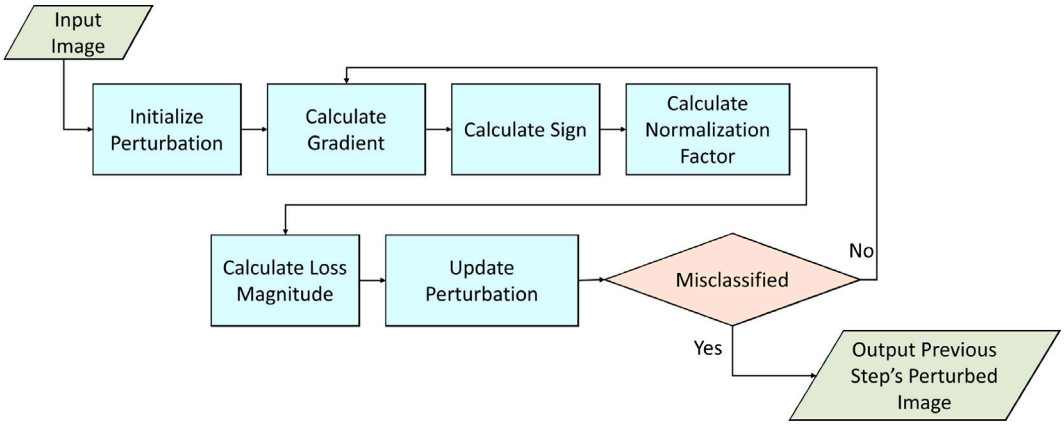


Fig. 6. Flowchart of the noise generation and perturbation process.

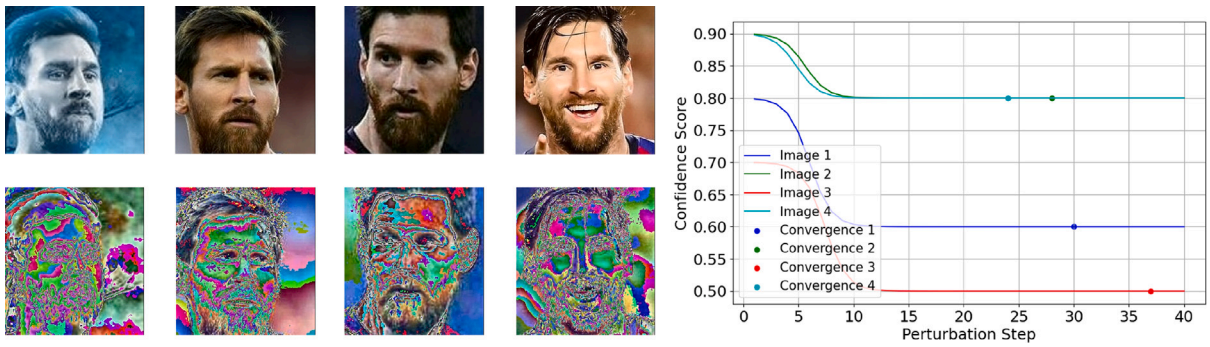


Fig. 7. Diverse convergence points in perturbed image generation: This figure shows four original and perturbed images of the renowned personality Lionel Messi from the pins dataset, each captured under distinct conditions. The accompanying graph illustrates the diverse convergence points for perturbed image generation, highlighting that although these images depict the same individual, their respective perturbations require different convergence points.

4.3. Analysis of results

Facial recognition systems face difficulties in recognizing people due to changes in their appearance over time. Factors like aging, hairstyles, and accessories such as sunglasses can affect the way a person looks. For example, we may have pictures of Tom Cruise from his younger years in the first Top Gun movie, which would look very different from his pictures in a sequel taken when he was older (refer to Fig. 5). Besides, Tom Cruise might be wearing black sunglasses in some pictures, or the picture could be taken from the side. These variations pose a challenge for traditional facial recognition systems. However, the proposed framework uses a dataset that already accounts for these situations. The dataset includes a wide range of facial variations like age differences, masked faces, side views, faces with glasses, and more. This diversity in the dataset reflects real-world scenarios and makes it difficult for traditional face recognition systems to function effectively. The model can handle these complex situations, demonstrating its practicality and real-world effectiveness in protecting privacy.

As shown in Fig. 8, the modified images demonstrate the effects of perturbation. This process effectively ensures the new dataset meets the requirements of federated learning for high utility while preserving privacy.

As depicted in Fig. 7, multiple images of an individual yield distinct perturbed instances, each converging to a different noise pattern. These differences enhance the dataset, thus significantly contributing to the robustness of the machine-learning models. When there are numerous examples, the models become better at comprehending different scenarios, rather than just memorizing content. This, in turn, improves the overall system's ability to protect privacy and perform with high robustness.

Users can adjust parameters, such as the epsilon value in the noise generator, to tailor the privacy-utility tradeoff according to specific application requirements. Smaller epsilon values result in lower perturbations, which provides reasonable privacy preservation with higher utility. On the other hand, larger epsilon values lead to more substantial perturbations, which enhance privacy but come at the expense of utility.

To assess the effectiveness of the proposed data perturbation technique, we conducted a runtime analysis on the datasets used in the experiments. The time required to perturb each image varies depending on factors such as the image resolution and the number of color channels. The detailed runtime analysis is presented in Table 2.

During the performance evaluation, we used VGG16 to represent the baseline. This analysis helps us understand how well machine learning models (under centralized settings) perform in comparison to different configurations of federated learning. This is a pre-trained model which we fine-tuned on the perturbed dataset. During the training and testing phases, we paid close attention to the hyperparameters and the model's performance.

Fig. 9(a) shows the VGG16 model convergence. The model demonstrates its ability to learn distinct features from the dataset, as we observe a significant reduction in both training and testing losses during the initial epochs.

As shown in Fig. 9(b), the VGG16 model rapidly reaches high levels of accuracy, highlighting its ability to classify surveillance images correctly. The accuracy of the testing set consistently increases, confirming the model's generalizability.

The Table 1 displays key performance metrics. The table shows the model's classification performance in the centralized machine learning setting.

Algorithm 1: The main steps of the proposed algorithm

Input: Surveillance (face) image: x
Neural network: Θ
Learning rate: η
Epsilon (ε) for perturbation
Reduction factor: ρ
Maximum iterations: max_iter
Output: Enhanced global model: Θ'

Step 1: Data Setup
Normalize the image x
Initialize neural network Θ

Step 2: Data Perturbation
Initialize perturbed image $x' \leftarrow x$
Initialize perturbation $\delta \leftarrow 0$
iterations $\leftarrow 0$
while *True* **do**
  Compute logits $f(x') = \Theta(x')$
  Predict label $k \leftarrow \arg\max f(x')$
  if *misclassification not achieved and iterations* $<$ max_iter **then**
    Compute gradient $g \leftarrow \nabla_x L(f(x'), y)$
    Update adversarial perturbation $\delta \leftarrow \delta + \text{sign}(g) \times \frac{|L(f(x'),y)|}{\|g\|_2^2}$
    Ensure perturbation control $\delta_i \leftarrow \max(-\varepsilon, \min(\varepsilon, \delta_i))$ for each element δ_i in δ
    Update perturbed image $x' \leftarrow x + \delta$
    Clip x' to valid pixel range
    iterations $\leftarrow$ iterations $+ 1$
  else
    Break
Apply reduction factor $x' \leftarrow x + \rho(x' - x)$;
Transmit x' to the edge server

Step 3: Collect Perturbed Images
Perturbed dataset: D^p = Gather all perturbed images x' generated from Step 2

Step 4: Federated Learning Using Perturbed Data
Initialize global model parameters W_G
for *each local client C* **do**
  Share perturbed data: $D_C = \{(x'_i, y_i)\}$
  Update client model W'_C using η
  Transmit W'_C to the global server
Aggregate client models: $W_G = \frac{1}{|C|} \sum W'_C$

Table 1
Performance metrics in centralized machine learning setting.

Train loss	Test loss	Train accuracy	Test accuracy
0.0422	0.0023	98.73%	99.95%

We implement early stopping after 35 epochs to bypass overfitting. The slight increase in test loss and reduction in training accuracy indicate this regularization technique helps prevent overfitting noise and improves model generalization.

The VGG16 model's 99.95% test accuracy highlights its robustness in classifying surveillance images with adversarial perturbations. This is due to the model's ability to capture intrinsic patterns in the perturbed dataset, contributing to the overall performance of the framework.

ResNet18 showed an impressive training performance, with a consistent decrease in training loss over 245 epochs, converging at 0.5128. The model's ability to capture complex patterns within the training data is evident. ResNet18 showed excellent generalization in testing with a stable test loss of 0.8155.

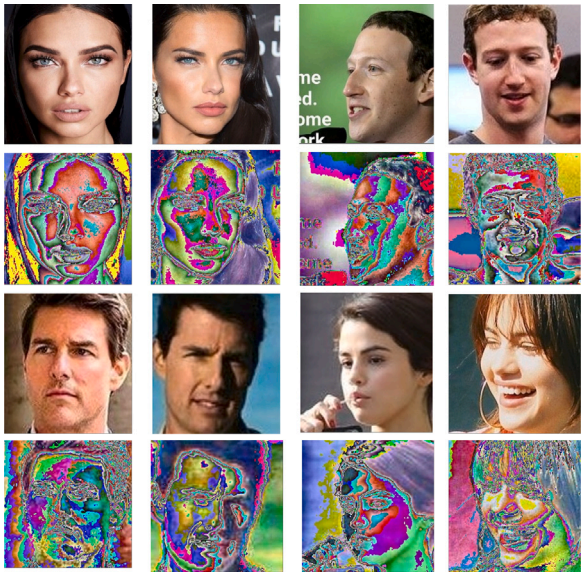
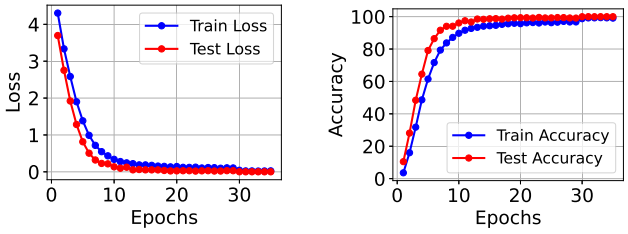
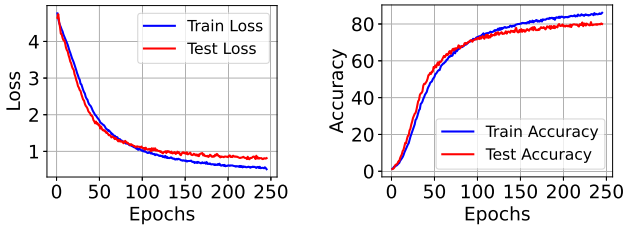


Fig. 8. Original and perturbed image examples: This shows a set of eight images featuring celebrities from the Pins dataset, with each of the four individuals represented by two distinct images captured from different angles. The remaining eight images correspond to perturbed versions generated by the noise generator for the same initial eight images.



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 9. Performance plot in centralized machine learning setting: Graph (a) displays training and testing loss, while graph (b) shows training and testing accuracy.



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

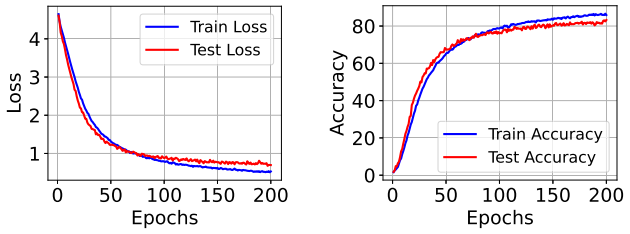
Fig. 10. Performance plot for model ResNet18 in federated learning environment: Graph (a) shows training and testing loss, while graph (b) illustrates training and testing accuracy.

ResNet18's robustness was demonstrated by the 80.01% test accuracy. The training and testing loss curves consistently decreased, signifying optimal convergence, as shown in Fig. 10(a). The corresponding training and testing accuracy curves in Fig. 10(b) indicate a continuous rise in accuracy during training.

After 200 epochs, GoogleNet's training loss stabilized at around 0.5281 with a training accuracy of approximately 85.91%. Although

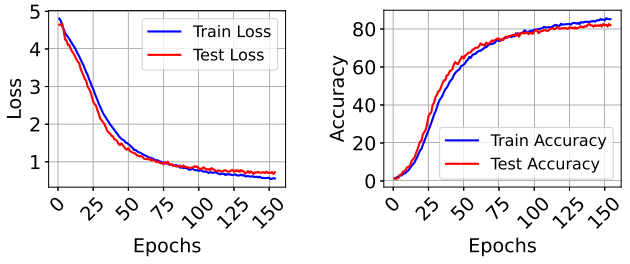
Table 2
Runtime analysis for data perturbation.

Dataset	Resolution	Color channel	Time consumed to perturb each image in the dataset (s)
Pins	Maximum resolution: 564 × 802 Minimum resolution: 102 × 82 Average resolution: 211.46 × 225.72	Both grayscale and RGB	0.0079
MNIST	28 × 28	Grayscale	0.0023
CIFAR-10	32 × 32	RGB	0.0046



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 11. The performance plot for GoogleNet in federated learning environment: Graph (a) presents the training and testing loss, and (b) shows the training and testing accuracy.



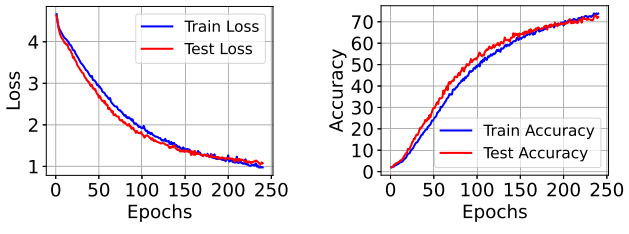
(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 12. The performance plot for model DenseNet in federated learning environment: Graph (a) presents the training and testing loss, and graph (b) shows the training and testing accuracy.

the model demonstrated consistent training dynamics, it showed slightly slower convergence. During testing, GoogleNet exhibited a test loss of about 0.6957 and a test accuracy of 83.23%, which underlines its effectiveness. As shown in Fig. 11(a), the training loss consistently decreased, while the testing loss reached a plateau after around 110 epochs. Besides, Fig. 11(b) illustrates the corresponding training and testing accuracy curves, showing a continuous accuracy rise during training.

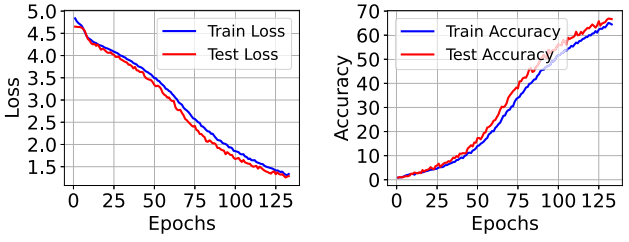
DenseNet demonstrated robust training dynamics, with the training loss converging to approximately 0.554 after 154 epochs. The training accuracy reached around 85.16%, highlighting its ability to model complex data patterns. During the testing phase, DenseNet maintained a stable test loss of about 0.7245, accompanied by a test accuracy of 82.01%, emphasizing its strong generalization capabilities (refer to Fig. 12(a)). The training loss consistently decreased while the testing loss stabilized after approximately 100 epochs. Fig. 12(b) shows the corresponding training and testing accuracy curves, demonstrating a continuous increase in accuracy during training.

After 240 epochs, MobileNet showed efficient training, with a training loss converging to around 0.9758. The model's ability to learn complex patterns was demonstrated by a training accuracy of 73.81%. Throughout the testing phase, the model retained a stable loss score of approximately 1.0742, achieving an accuracy of 72.33%. These results highlight the model's reliability and effectiveness in real-world



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 13. The performance plot for MobileNet in federated learning environment: Graph (a) presents the training and testing loss, and graph (b) shows the training and testing accuracy.



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 14. The performance plot for ResNeXt in federated learning environment: Graph (a) presents the training and testing loss, and graph (b) shows the training and testing accuracy.

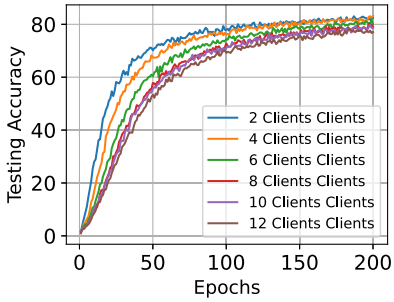


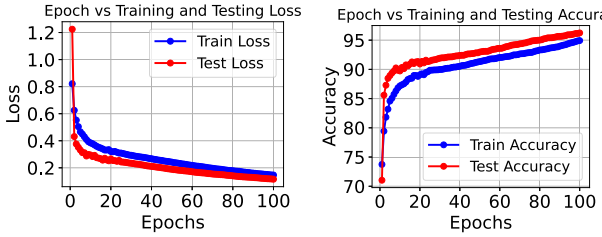
Fig. 15. Testing accuracy in the FL environment for the perturbed dataset with varying client configurations.

scenarios (refer to Fig. 13(a)). The graph shows that the training loss steadily decreased over time, while the testing loss remained relatively constant after around 170 epochs. Besides, Fig. 13(b) displays the corresponding accuracy curves for both training and testing, which demonstrate a consistent improvement in accuracy during the training phase.

After 133 epochs, ResNeXt's training loss converged at approximately 1.3365 and generated a training accuracy of around 64.47%. Although it took slightly longer for the model to converge, it was able to capture data patterns effectively. The model's test loss stabilized at

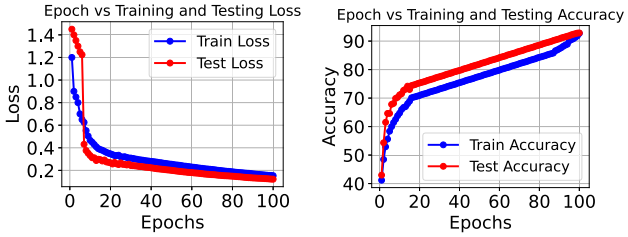
Table 3
Model performance comparison: The comparison is based on key metrics in a federated learning setting for both the raw and the perturbed datasets. Notably, ResNet18, GoogLeNet, and DenseNet exhibit better performance, showcasing higher accuracy. In contrast, MobileNet and ResNeXt demonstrate comparatively lower accuracy.

Model	Number of epochs	Raw dataset				Perturbed dataset			
		Train loss	Test loss	Train accuracy	Test accuracy	Train loss	Test loss	Train accuracy	Test accuracy
ResNet18	245	0.0376	0.4965	99.03%	92.67%	0.5128	0.8155	86.10%	80.01%
GoogleNet	200	0.4367	0.4453	88.64%	88.82%	0.5281	0.6957	85.91%	83.23%
DenseNet	154	0.3999	0.4877	89.45%	87.57%	0.5540	0.7245	85.16%	82.01%
MobileNet	240	0.6081	0.6311	83.50%	83.58%	0.9758	1.0742	73.81%	72.23%
ResNeXt	133	0.7757	0.7613	78.78%	80.33%	1.3365	1.2884	64.47%	66.67%



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 16. The performance plot for MNIST dataset using GoogleNet model in federated learning environment: Graph (a) presents the training and testing loss, while graph (b) illustrates the training and testing accuracy.



(a) Training and Testing Loss **(b)** Training and Testing Accuracy

Fig. 17. The performance plot for CIFAR-10 dataset using GoogleNet model in federated learning environment: Graph (a) presents the training and testing loss, while graph (b) illustrates the training and testing accuracy.

Table 4
Comparison of training and testing results across different datasets.

Dataset	Model	Number of epochs	Train accuracy	Test accuracy
Pins	GoogLeNet	200	85.91%	83.23%
MNIST		100	94.93%	96.24%
CIFAR-10		100	92.7%	92.87%

Table 5
Time complexity analysis for the proposed algorithm.

Steps	Time complexity
Iterative perturbation process	$O(max\_iter \times (2F + d))$
Reduction factor application	$O(d)$
The overall time complexity	$O(max\_iter \times (2F + d))$

about 1.2884 during the testing phase, with a test accuracy of 66.67%, demonstrating its reliable performance in decentralized settings. In Fig. 14(a), the training and testing loss curves for ResNeXt are depicted. The training loss consistently decreased, while the testing loss reached a plateau after approximately 125 epochs. Fig. 14(b) shows the corresponding training and testing accuracy curves, which indicates a continuous improvement in accuracy during training.

To provide a concise comparative overview of the five models, we summarized the key results in Table 3.

As shown in Fig. 15, GoogleNet's performance is tested under varying client scenarios, specifically 2, 4, 6, 8, 10, and 12 clients. The figure illustrates the changes in training accuracy during federated learning, showcasing the robustness of the proposed solution in different multi-client settings.

For the MNIST dataset, after 100 rounds, the training accuracy was approximately 94.93%, and the testing accuracy was approximately 96.24%. In contrast, for the CIFAR-10 dataset, the training accuracy was 92.7%, and the testing accuracy was 92.87% approximately after 100 rounds. Fig. 16(a) shows the training and testing loss curves for the MNIST dataset, while Fig. 16(b) illustrates the corresponding training and testing accuracy curves. Fig. 17(a) depicts the training and testing loss curves for the CIFAR-10 dataset, and Fig. 17(b) presents the training and testing accuracy curves. Table 4 summarizes the detailed outcomes for the three different datasets using the same deep learning model in the federated learning environment.

4.4. Complexity analysis of the proposed algorithm

In this section, we analyze the time complexity of the proposed data perturbation algorithm. The core of the perturbation process involves adding noise to the input image until it is misclassified or the maximum number of iterations is reached. In each iteration, computing logits and gradient calculations have a complexity of $O(F)$, where F represents the complexity of a forward or backward pass through the network. If the number of pixels in the image is d, the operation of clipping the image values to the valid pixel range has a time complexity of $O(d)$. For a maximum iteration max_iter, the total time complexity of the loop will be $O(max\_iter \times (2F + d))$. The complexity of applying the reduction factor to the perturbed image is denoted as $O(d)$. Combining all the steps, the overall time complexity for data perturbation is given in Table 5. The perturbation process is linearly dependent on both the number of iterations and the number of pixels in the image.

5. Related work

In recent years, there has been significant research interest in integrating privacy preservation into federated learning in smart cities. The methods for preserving privacy in Federated Learning (FL) can be broadly divided into two categories: encryption-based techniques [9], often referred to as cryptographic methods, and data perturbation (modification)-based methods [7].

5.1. Privacy-preserving federated learning

Federated learning allows training models on decentralized devices while keeping user data local [15]. Bonawitz et al. proposed Federated Averaging as one of the protocols for training models on user devices while ensuring privacy during global model updates in FL [39]. Their work prioritizes communication efficiency and Byzantine fault tolerance, ensuring scalability despite non-i.i.d. data distribution. However, it still faces challenges regarding fairness and mitigating the impact of adversaries. Recent research by Yang et al. [40] integrates homomorphic encryption to enhance data privacy in federated learning aggregation. This approach allows model aggregation without

Table 6
Summary of related work.

Group	Method	Advantages	Limitations
Encryption-based privacy-preserving federated learning	Secure Multi-Party Computation (MPC)	➢ Secure aggregation of parameters [29], ➢ Ensures data privacy [29], ➢ Fault tolerance [29], ➢ Scalability [33]	➢ Requires trusted parties [30,31], ➢ Vulnerable to attacks [9] ➢ High communication costs [9,32]
	Homomorphic encryption	➢ Algebraic operations on encrypted data [34], ➢ Enhanced data privacy [34] ➢ Secure parameter aggregation [33]	➢ Scalability issues in a distributed setting [35,36] ➢ Low efficiency [35,36]
Perturbation-based privacy-preserving federated learning	Global differential privacy-based	➢ Private learning of algorithms (e.g., SGD) [10,37], ➢ Enhanced privacy guarantees [10,37], ➢ Fair trade-off between privacy and utility [10,37]	➢ Need for a trusted aggregator [10,37], ➢ Focus on privacy leaks among FL clients [10,37]
	Local differential privacy-based	➢ Strong privacy setting [14,38], ➢ Data randomization for security [14,38] ➢ Improved privacy budget management [14,38]	➢ Substantial consumption of privacy budgets (ϵ) [14,38], ➢ Lower accuracy compared to global differential privacy-based approaches [14,38]

direct access to individual contributions, providing enhanced privacy guarantees at the cost of potential computational overhead.

Table 6 present a summary that categorizes privacy-preserving methods in federated Learning into two groups: Encryption-Based Privacy-Preserving Federated Learning and Perturbation-Based Privacy-Preserving Federated Learning. The table outlines the key methods and their advantages and limitations.

5.2. Encryption-based privacy-preserving federated learning

Cryptographic methods in Federated Learning primarily focus on secure parameter aggregation at the FL server. The most widely adopted cryptographic approach for ensuring secure aggregation is known as secure multi-party computation (MPC) [29]. Multi-party computation allows for the secure evaluation of a function on private data, referred to as secret shares, which is distributed among multiple parties who may not trust each other [9]. However, many existing multi-party computation approaches for FL face several challenges. Some require a trusted party (e.g., VerifyNet [30] and VeriFL [31]), while others involve a notably high number of communications (e.g., approaches by Bonawitz et al. [9] and Bell et al. [32]). Furthermore, many of these multi-party computation approaches are susceptible to advanced adversarial attacks, such as backdoor attacks [41]. Nevertheless, the secure multi-party computation-based method encounters challenges in improving computational efficiency due to the significant computational resources required for each training round [42].

Another frequently employed cryptographic method is Homomorphic Encryption (HE), which enables algebraic operations on encrypted data to generate a ciphertext. This ciphertext can be decrypted to reveal the algebraic result on the original plaintext while ensuring security and privacy [34]. Nevertheless, scalability has posed a significant challenge for homomorphic encryption. Recent approaches like BatchCrypt aim to introduce less complex homomorphic encryption-based solutions for secure parameter aggregation in FL [33]. However, it is important to note that homomorphic encryption in a distributed setting can be inefficient for large-scale scenarios due to its low efficiency [35,36].

5.3. Perturbation-based privacy-preserving federated learning

The main perturbation-based techniques in Privacy-Preserving Federated Learning (PPFL) fall into two categories: global differential privacy-based and local differential privacy-based methods. Both global differential privacy-based (e.g., [10,37]) and local differential privacy-based (e.g., [38,43], and [14]) approaches have been introduced to Federated Learning [7].

Global differential privacy-based methods focus on ensuring the private learning of the algorithm (e.g., Stochastic Gradient Descent or SGD) [10,44]. In contrast, local differential privacy-based methods concentrate on randomizing the input data. This can involve the direct randomization of user inputs or randomization of the model parameters before they are sent to the aggregator for learning on randomized data. Examples of global differential privacy-based approaches include the methods by Robin et al. [10], and Asoodeh et al. [37]. Local differential privacy-based approaches include LDPFL [43], LDP-Fed [14] and the approach by Seif et al. [38]. One notable challenge with most global differential privacy-based approaches is the need for a trusted curator. These approaches primarily address privacy concerns among the Federated Learning clients [10,37]. In contrast, local differential privacy-based approaches offer a stricter privacy setting [14,38]. However, existing local differential privacy-based methods often consume substantial privacy budgets (ϵ) to achieve high accuracy, operate on generalized local differential privacy-based guarantees (e.g., $\alpha - CLDP$), or do not achieve the same level of accuracy as global differential privacy-based approaches. Therefore, there exists a significant trade-off between the privacy and utility of local differential privacy-based methods.

6. Conclusion

Our work presents a novel framework that addresses the challenge of balancing privacy and effectiveness of federated learning in smart city surveillance camera networks. To achieve this, we developed a method that utilizes noise generation and application to convert raw surveillance data into privacy-preserving data. The proposed approach ensures that individual privacy concerns are addressed while retaining the effectiveness of federated learning settings. Evaluations demonstrate the effectiveness of the proposed solution, with the highest testing accuracy of 99.95% in centralized machine learning settings and 96.24% in federated learning settings.

As part of future work, we plan to explore various directions, including managing non-IID data and optimizing communication.

CRediT authorship contribution statement

Farah Wahida: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing. **M.A.P. Chamikara:** Conceptualization, Investigation, Project administration, Resources, Supervision, Validation, Writing – review & editing. **Ibrahim Khalil:** Funding acquisition, Project administration, Resources, Supervision. **Mohammed Atiquzzaman:** Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] E.M. Newton, L. Sweeney, B. Malin, Preserving privacy by deidentifying face images, IEEE Trans. Knowl. Data Eng. 17 (2) (2005) 232–243.

[2] K.W. Bowyer, Face recognition technology: Security versus privacy, IEEE Technol. Soc. Mag. 23 (1) (2004) 9–19.

[3] Chinese facial recognition company exposes data of millions of people, 2022, URL https://edgy.app/chinese-facial-recognition-company-data. (Accessed 10 April 2022).

[4] What is, 'Skynet', China's massive video surveillance network, 2022, URL https://www.abacusnews.com/whowhat/skynet-chinas-massive-video-surveillance-network/article/2166938. (Accessed 10 April 2022).

[5] Facebook agrees to pay 550 Million to settle privacy class action, 2022, URL https://www.nytimes.com/2020/01/29/technology/facebook-privacy-lawsuit-earnings.html. (Accessed 10 April 2022).

[6] P.C.M. Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, M. Atiquzzaman, Local differential privacy for deep learning, IEEE Internet Things J. 7 (7) (2019) 5827–5842.

[7] K. Wei, J. Li, M. Ding, C. Ma, H.H. Yang, F. Farokhi, S. Jin, T.Q. Quek, H.V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469.

[8] Y. Zhang, G. Bai, X. Li, C. Curtis, C. Chen, R.K. Ko, Privcoll: Practical privacy-preserving collaborative machine learning, in: European Symposium on Research in Computer Security, Springer, 2020, pp. 399–418.

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

[10] R.C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective, 2017, arXiv preprint arXiv:1712.07557.

[11] P. Kairouz, S. Oh, P. Viswanath, Extremal mechanisms for local differential privacy, Adv. Neural Inf. Process. Syst. 27 (2014).

[12] X. Xiao, Y. Tao, Output perturbation with query relaxation, Proc. VLDB Endow. 1 (1) (2008) 857–869.

[13] L. Sun, J. Qian, X. Chen, LDP-FL: Practical private aggregation in federated learning with local differential privacy, 2020, arXiv preprint arXiv:2007.15789.

[14] S. Truex, L. Liu, K.-H. Chow, M.E. Gursoy, W. Wei, LDP-Fed: Federated learning with local differential privacy, in: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, 2020, pp. 61–66.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[16] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8026–8037.

[19] https://www.kaggle.com/datasets/hereisburak/pins-face-recognition.

[20] https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

[21] https://www.kaggle.com/datasets/oxcdcd/cifar10.

[22] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.

[23] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[24] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2016.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, ..., A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2015.

[26] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017.

[27] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, ..., H. Adam, MobileNets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017.

[29] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T.D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, et al., SAFELearn: Secure aggregation for private federated learning, in: 2021 IEEE Security and Privacy Workshops, SPW, IEEE, 2021, pp. 56–62.

[30] G. Xu, H. Li, S. Liu, K. Yang, X. Lin, Verifynet: Secure and verifiable federated learning, IEEE Trans. Inf. Forensics Secur. 15 (2019) 911–926.

[31] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, T. Baker, V eri fl: Communication-efficient and fast verifiable aggregation for federated learning, IEEE Trans. Inf. Forensics Secur. 16 (2020) 1736–1751.

[32] J.H. Bell, K.A. Bonawitz, A. Gascón, T. Lepoint, M. Raykova, Secure single-server aggregation with (poly) logarithmic overhead, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 1253–1269.

[33] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, Y. Liu, {$BatchCrypt$}: Efficient homomorphic encryption for {$Cross-Silo$} federated learning, in: 2020 USENIX Annual Technical Conference (USENIX ATC 20), 2020, pp. 493–506.

[34] C. Gentry, A Fully Homomorphic Encryption Scheme, Stanford university, 2009.

[35] J. So, B. Güler, A.S. Avestimehr, Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning, IEEE J. Sel. Areas Inf. Theory 2 (1) (2021) 479–489.

[36] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.

[37] S. Asoodeh, W.-N. Chen, F.P. Calmon, A. Özgür, Differentially private federated learning: An information-theoretic perspective, in: 2021 IEEE International Symposium on Information Theory, ISIT, IEEE, 2021, pp. 344–349.

[38] M. Seif, R. Tandon, M. Li, Wireless federated learning with local differential privacy, in: 2020 IEEE International Symposium on Information Theory, ISIT, IEEE, 2020, pp. 2604–2609.

[39] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, et al., Towards federated learning at scale: System design, Proc. Mach. Learn. Syst. 1 (2019) 374–388.

[40] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, B. He, A survey on federated learning systems: Vision, hype and reality for data privacy and protection, IEEE Trans. Knowl. Data Eng. (2021).

[41] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2938–2948.

[42] M.S. Riazi, K. Laine, B. Pelton, W. Dai, HEAX: An architecture for computing on encrypted data, in: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, 2020, pp. 1295–1309.

[43] P.C. Mahawaga Arachchige, D. Liu, S. Camtepe, S. Nepal, M. Grobler, P. Bertok, I. Khalil, Local differential privacy for federated learning, in: European Symposium on Research in Computer Security, Springer, 2022, pp. 195–216.

[44] H.B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private recurrent language models, 2017, arXiv preprint arXiv:1710.06963.

Farah Wahida is currently pursuing a Ph.D. degree in Computer Science in School of Computing Technologies, RMIT University, Melbourne, Australia. Her research interests include Machine Learning, Privacy-Preserving techniques, Cybersecurity, Edge Computing, and Distributed System.

M.A.P. Chamikara is a research scientist at CSIRO's Data61. He received his Ph.D. from RMIT University, Australia. Before joining CSIRO's Data61, Chamikara was a lecturer at the University of Peradeniya in Sri Lanka. He earned both his M.Phil. and B.Sc. Honors in Computer Science from the same university. His research has been accepted and published in highly reputable venues such as AAAI, WWW, EORICS, AsiaCCS, CIKM, IEEE IoT, IEEE Industrial Informatics, and Information Sciences. His research interests include data privacy, machine learning security, distributed privacy-preserving machine Learning, explainable AI, and human-centric cyber security.

Ibrahim Khalil is a Professor at the School of Computing Technologies at RMIT University, Melbourne, Australia. He received his Ph.D. in Computer Science from the University of Bern, Switzerland, in 2003, marking the beginning of a career that would span continents and sectors. Before his tenure at RMIT University, Khalil amassed significant experience in the tech hubs of Silicon Valley, where he worked as a software engineer focused on secure network protocols and smart network provisioning. His academic journey also includes valuable stints at EPFL and the University of Bern in Switzerland, and Osaka University in Japan. A prolific contributor to the fields of Blockchain and Privacy, Khalil has led several high-profile ARC discovery and linkage grants in Australia between 2017 and 2022, alongside securing international European grants, industry grants, one Department of Defence Strategic Policy Grant, and a QNRF grant from Qatar. His research portfolio is diverse, encompassing Privacy, Blockchain, secure AI data analytics, distributed systems, e-health, wireless and body sensor networks, biomedical signal processing, and network security, reflecting a broad and impactful engagement with the frontiers of computing and technology.

Mohammad Atiquzzaman (Senior Member) received his B.Sc. from Bangladesh University of Engineering and Technology, Bangladesh, in 1982. He received his M.S. and Ph.D. degrees in electrical engineering and electronics from the University of Manchester, Manchester, U.K., in 1984 and 1987, respectively. He currently holds the Edith Kinney Gaylord Presidential professorship and the Hitachi Chair Professor in the School of Computer Science at the University of Oklahoma. He is the Editor-in-Chief of the Journal of Networks and Computer Applications, founding Editor-in-Chief of Vehicular Communications, and has served/is serving on the Editorial Boards of various IEEE journals including IEEE Journal on Selected Areas in Communications. He co-chaired numerous IEEE international conferences, including IEEE GLOBECOM/ICC. His research interests include communications networks, Internet protocols, wireless and mobile networks, satellite networks, and optical communications.