CS5331 Special Problems in Computer Science: Security:

Adversarial Machine Learning

Team Members

Charishma Rathan Bala – R11908154

Keerthiga Kalidas – R11903641

Neha Bollu – R11903528

Sruthi Mandalapu – R11906160

Detailed Documentation:

1) **File1-Attack Before Federated Learning**

Overview:

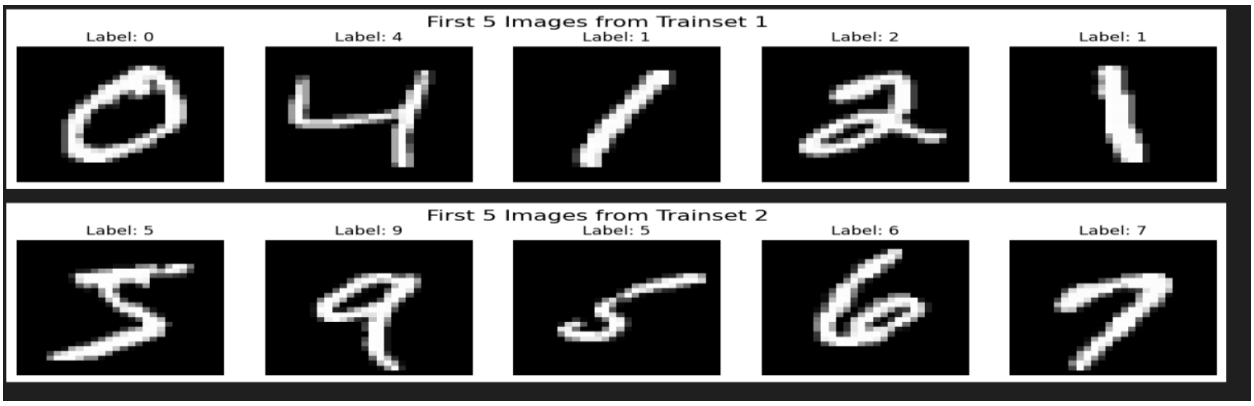
This notebook demonstrates an attack on individual datasets before setting up a federated learning framework. Below is a detailed explanation outlining its purpose, implementation, and observations.

Step 1: Imports for Dataset Loading, Declaring the Model, and Initializing MIA Attack

The code reads the MNIST dataset and then builds a CNN model using the layers available in Keras for classification. It imports the MIFace attack from the ART library for model inversion to evaluate the model. First, eager execution is disabled, second, warnings are turned off and finally the random number generator is set for the purpose of repetition.

Step 2:Data Preprocessing

The dataset used in this work is the MNIST dataset with 60k training images and 10k testing images. Data is split between two clients: Client1 receives 0-4 as digits and Client2 receives 5-9 as digits. The shapes of the datasets and the first 5 observations from each client's data are then displayed.

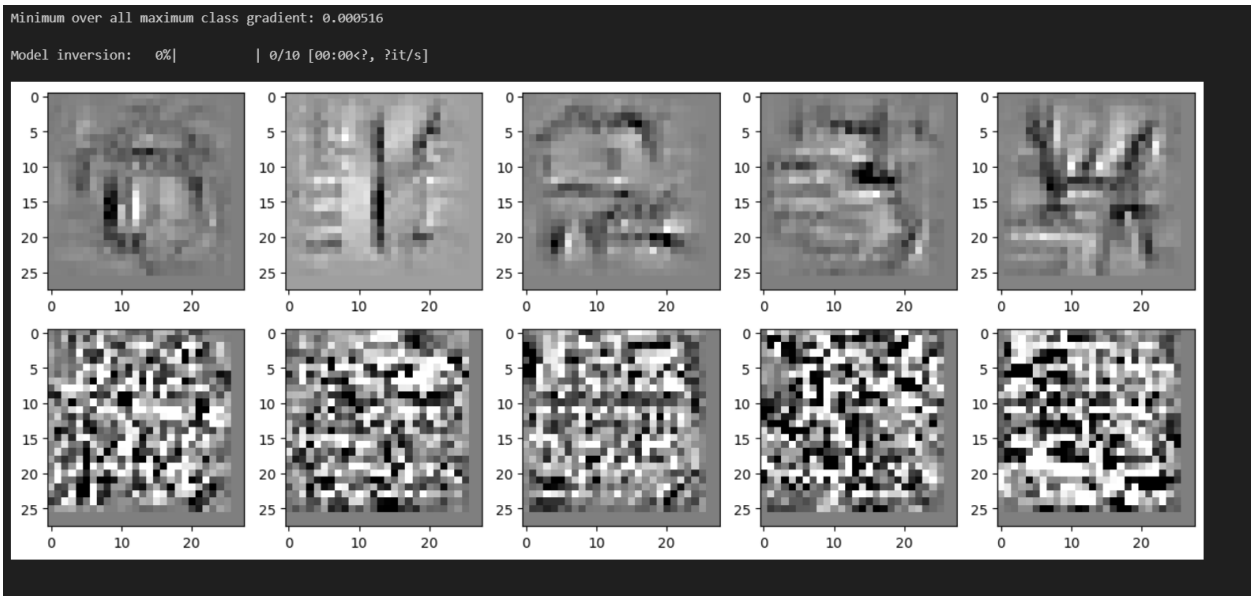


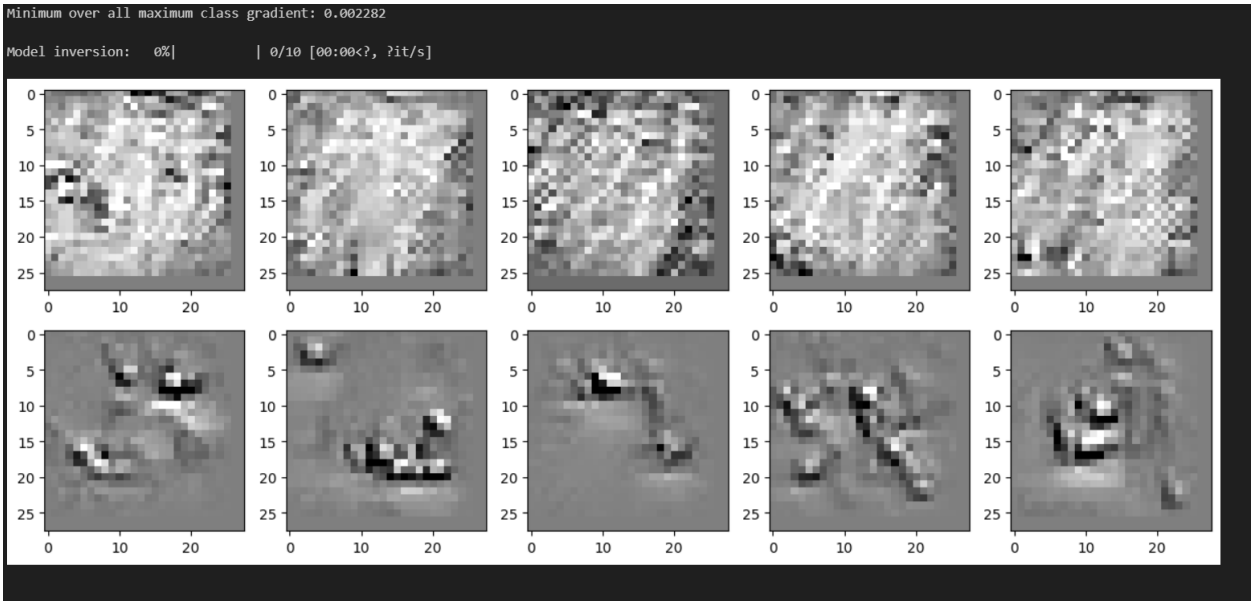
PART 1: Train the Model

A CNN model is defined for MNIST classification, with Client1 training on digits 0-4 and Client2 on digits 5-9. Both clients test their models on their own and the other's dataset. The results show that each client performs well on its own test set

PART 2: Perform Model Inversion Attack

Model Inversion Attack (MIA) is performed on both Client1 and Client2 to reconstruct input images. Client1, trained on digits 0-4, successfully reconstructs these digits, while Client2, trained on digits 5-9, successfully reconstructs its corresponding digits. Results show that each client's MIA can accurately reconstruct the images it was trained on.





2) **File2-Attack After Federated Learning**
   The notebook imitates federated learning with Client1 and Client2 that used different fragment of the MNIST dataset. All clients train a CNN model and Federated Averaging (FED-AVG) performs the model averaging based on the updates. MIA is used to examine image reconstruction; we find that each client can predict its own digits and, with MIA, reconstruct from the models.

Step 1: Imports for dataset loading, declaring the model and the MIA attack

The very first step is imports necessary libraries for the project. It loads the MNIST dataset using Keras, defines the CNN model architecture with Keras layers, and sets up the Model Inversion Attack (MIA) using adversarial-robustness-toolbox. It also configures TensorFlow by disabling eager execution and suppressing warnings, and sets a random seed for reproducibility.

Step 2:Data Preprocessing

The notebook preloads and partitions the MNIST dataset for federated learning purposes; Client 1 is learning digits from 0 to 4; and Client 2 from digits 5 to 9. Finally, the shapes of the dataset is shown and five samples from each client are plotted to ensure the data has been loaded and structured correctly.

PART 1: Federated Learning Setup Overview

In federated learning, clients train models on local data, and exchange incremental model updates or weights instead of data. The CNN model is employed with both client training and testing and exchanging weights via communication round. FED-AVG process initializes the global model and then in each step advances the accuracy of this model on the test sets of clients.

PART2: Perform MIA Attack

Below, the attack mechanics is explained in detail, especially focusing on the attack process and its results.

1. **Attack Setup**
   The attack begins by creating an instance of the MIFace attack class:

```
# Create the attack
attack = MIFace(converged_model, max_iter=20000, threshold=1.5)
```

The converged model is a target model of the inversion attack for instance, a classifier such as CNN trained using MNIST data set. The attack iterates through a maximum of 20,000 iterations, in a bid to achieve the best solution to the optimization for image inversion though this incurs huge computation. The scalar value of 1.5 controls the adaptation in the inversion process and when the attack is successful to get the generated image, which will have high probability to be classified correctly.

2. **Attack on Black Sample**
   The process encompasses creating a new black image with all pixel values starting from one is then created. This has the objective of matching every one of the ten MNIST classes through this optimisation with Model Inversion Attack (MIA). Each class accounts for gradients which means the image is gradually adjusted to the target class.

3. **Attack on White Sample**

   The input now is a black and white image; the pixel values in this image are set at 0. The objective is to reconstruct images correspond to 10 MNIST classes from scratch with this white picture. Finding the output gradient of the model, white image is updated further with an attempt to optimize it for the target class.
4. **Attack on Grey Sample**
   The input image starts in the grey form and pixel values are set to 0.5 which is in the middle of the grey tone scale. The aim is to train the learning model, in order to generate class-specific images from this grey intermediate image. The attack works by using the output gradient of the model to manipulate a grey image until it aligns with a target class.

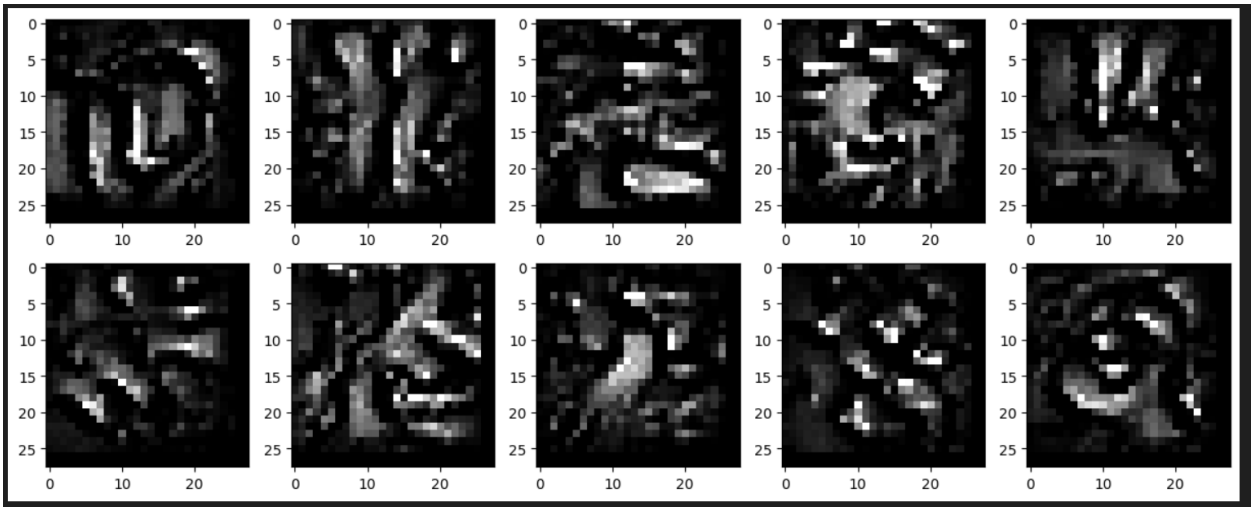
5. **Attack on Random Sample**

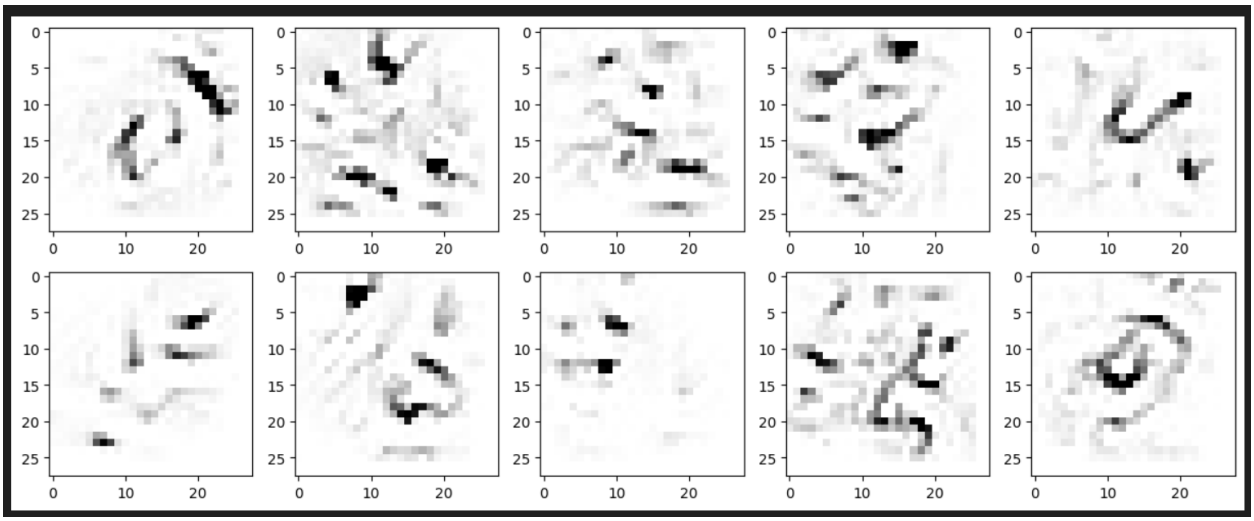
   The input image stated as the random image having pixel intensity in the range of 0 and 1. The purpose is to reconstruct the target class images using the inversion of the classifier. In the model, the gradient of the output is calculated and the random image is adjusted throughout a number of iterations towards the required class prediction.

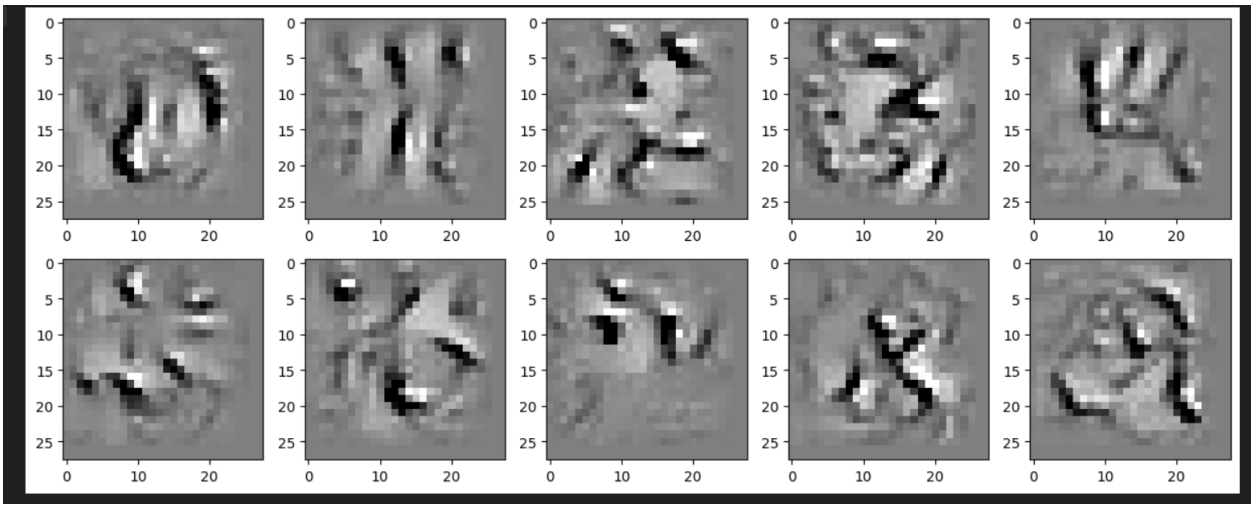
6. **Attack on Average Sample**

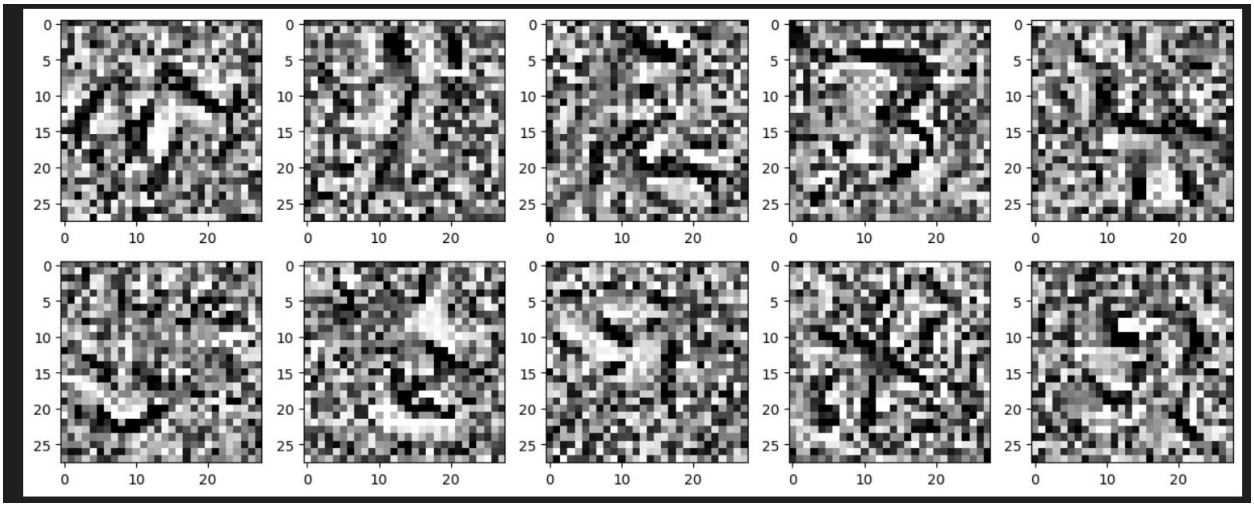
The input image is set as an average image; its pixel values are the mean of the test set images (x_test). Such, the task is to test whether the model can reconstruct class-specific images from average pixel values. The output of this model is then taken through differentiation with respect to the mean(s) and successive changes are made until the corresponding image for each of the predicted classes is obtained.

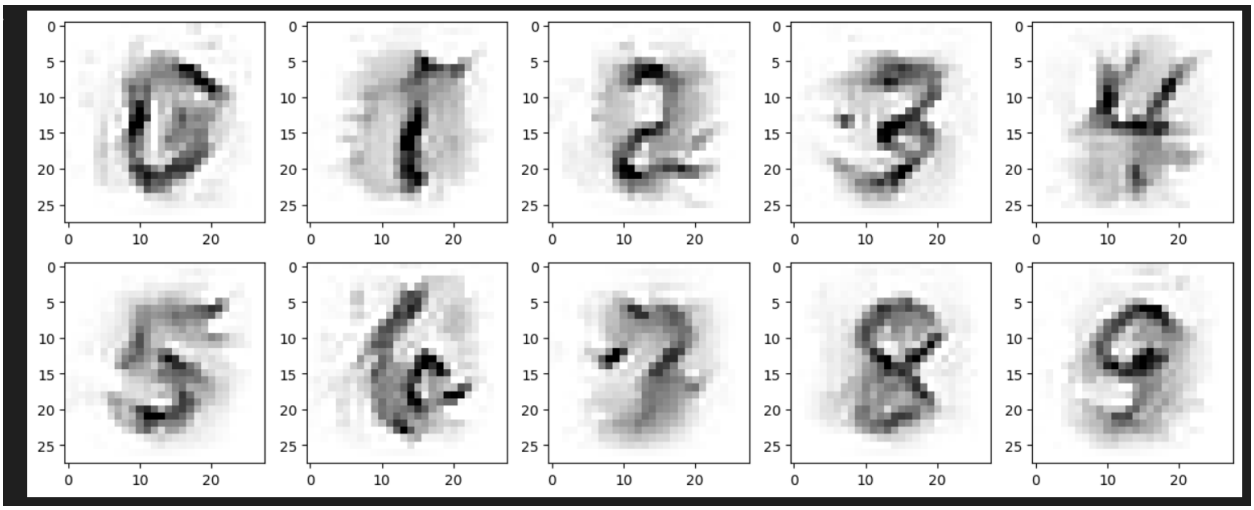
ATTACK OUTPUT











Part3: Perform MIA Attack - at different thresholds

Various threshold values ([0.1, 0.2, 0.5, 1, 1.5]) are utilized to modulate the presence of image inversion in the MIA attack. As the first input to launch the attack, the average input from the test dataset is taken. To compute the gradients of each class, in the attack, it is suggested as modifying the input image so as to break the class-specific features. Lower threshold values (i.e., 0.1) will increase the subtlety of their inference, while higher thresholds (i.e., 1.5) will lead to larger distortion. Evident is the inferred image at each threshold for observing the influence of a threshold on their quality and strength of the posed attacks.

Part4: Perform MIA Attack - at different iterations

The objective of the MIA is to conduct an experiment varying the number of iterations from 0 to 20,000 to see how an increased number of iterations enhances the effectiveness of the attack. The attack manipulates the average image of the test set by calculating the class gradients, indicating the changes needed to backtrack the image toward each class. As iterations increase, the attack will yield, for instance, an image reconstruction starting with lower iterates (e.g. 0 or 10) that are less accurate or clear visualizations. As the number of iterations increases from 1000 to 20,000, better fidelity to the class-feature model is achieved. For each number of iterations, there is visualization of the inferred images for all classes, which shows how the quality of the reconstructed images improves with increasing iterations, establishing the importance of iterative refinement in improving attack performance.

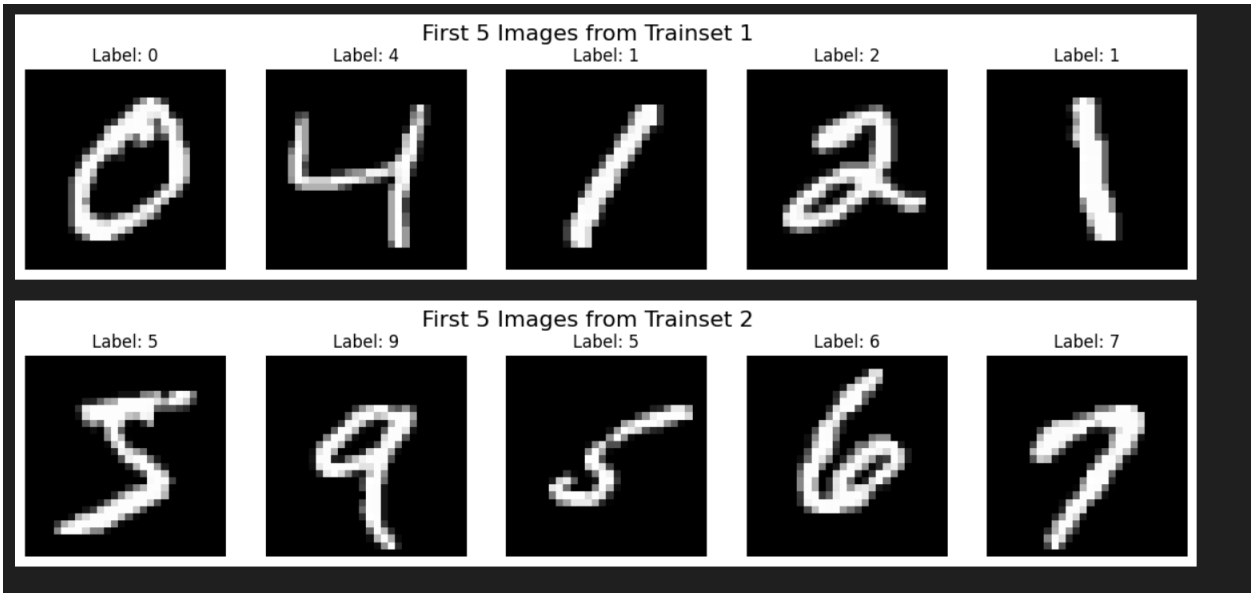
3) **File3-Defense**

Step1:Imports for dataset loading, declaring the model, MIA attack and for defense

The code first loads and preprocesses so-called MNIST for the purpose of recognizing handwritten digits and uses one-hot encoding for classification. In Keras with Sequential, for the neural network, some layers like Conv2D, MaxPooling2D, Dropout and Dense are used. For the purpose of testing adversarial scenarios, the MIFace model inversion attack from the ART library is used. Furthermore, the code also uses the TenSEAL library to study defense methods that employ homomorphic encryption while testing model inference to protect the confidentiality and integrity of data .

Step 2:Data Preprocessing

For the MNIST dataset, it is normalized on the pixel, the shape of inputs is changed to adjust channels, and labels are then one-hot encoded. It is split into two subsets for federated learning: For labels 0–4, Client 1 is in charge, while for labels 5–9, Client 2 takes over. They are checked by shape and for the first five samples the appearance of the splits confirms the results.



PART1: Federated Learning Setup and apply Homomorphic Encryption

Federated Learning (FL) facilitates model training across multiple clients, even if the client datasets are never transmitted to anyone, including the server. The Federated Averaging (FedAvg) algorithm works by receiving encrypted model weights from clients in multiple communication rounds and averaging them. For the purpose of improving the security of data, a resource called Homomorphic Encryption (HE) is integrated with the help of the TenSEAL library that enables Floating-point Computing for the HE's CKKS scheme. Promising steps include key generation, model weighting encryption, encrypted aggregation, model decryption, and updated models across clients. This arrangement guarantees safe and fast weight update and privacy in the FL process.

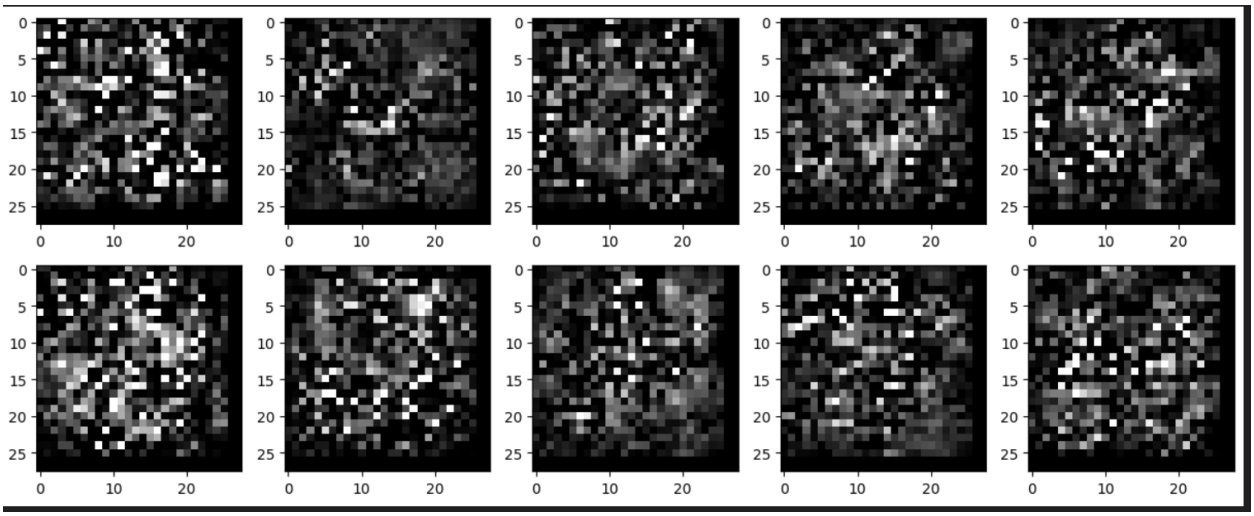
```
Communication Round 1
Test Set 1 - Accuracy: 0.9969
Test Set 2 - Accuracy: 0.9932
Client1 Weights
[[ 0.17064331  0.00465998  0.27132416  0.16361156  0.11087171 -0.09916199
   0.0395917   0.0094496   0.11911853  0.15050758 -0.05023036  0.00869079
   0.08361848 -0.0865319  -0.05055594  0.16742417 -0.04147151  0.05053979
   0.05741062 -0.20215505 -0.0223883  -0.2135417   0.0563137   0.07698046
   0.00223782  0.16372499  0.05948886 -0.19232199  0.1583745  -0.08396917
   0.1384893   0.14879633]]
Client2 Weights
[[ 0.00934195  0.01208514 -0.06094045 -0.1718946  -0.16290691  0.02000597
  -0.037304    0.06125158 -0.0629814  -0.09444547  0.07002956 -0.08388591
   0.03864778 -0.02284793 -0.05968269 -0.16680014 -0.20555423  0.18387994
  -0.01400697  0.18535955  0.13546924  0.15378612 -0.2881564   0.1558589
   0.09369459 -0.18871833  0.10460564  0.12997515  0.05897065 -0.22372495
   0.11763521  0.02273992]]
WARNING: The input does not fit in a single ciphertext, and some operations will be disabled.
The following operations are disabled in this setup: matmul, matmul_plain, enc_matmul_plain, conv2d_im2col.
If you need to use those operations, try increasing the poly_modulus parameter, to fit your input.
WARNING: The input does not fit in a single ciphertext, and some operations will be disabled.
The following operations are disabled in this setup: matmul, matmul_plain, enc_matmul_plain, conv2d_im2col.
If you need to use those operations, try increasing the poly_modulus parameter, to fit your input.
WARNING: The input does not fit in a single ciphertext, and some operations will be disabled.
The following operations are disabled in this setup: matmul, matmul_plain, enc_matmul_plain, conv2d_im2col.
...
   0.05776982 -0.07605778  0.10739009 -0.16499514  0.14838816 -0.31364896
   0.16549132  0.08032079]]
Converged model on client1 test set: 0.9932
Converged model on client2 test set: 0.9346
```

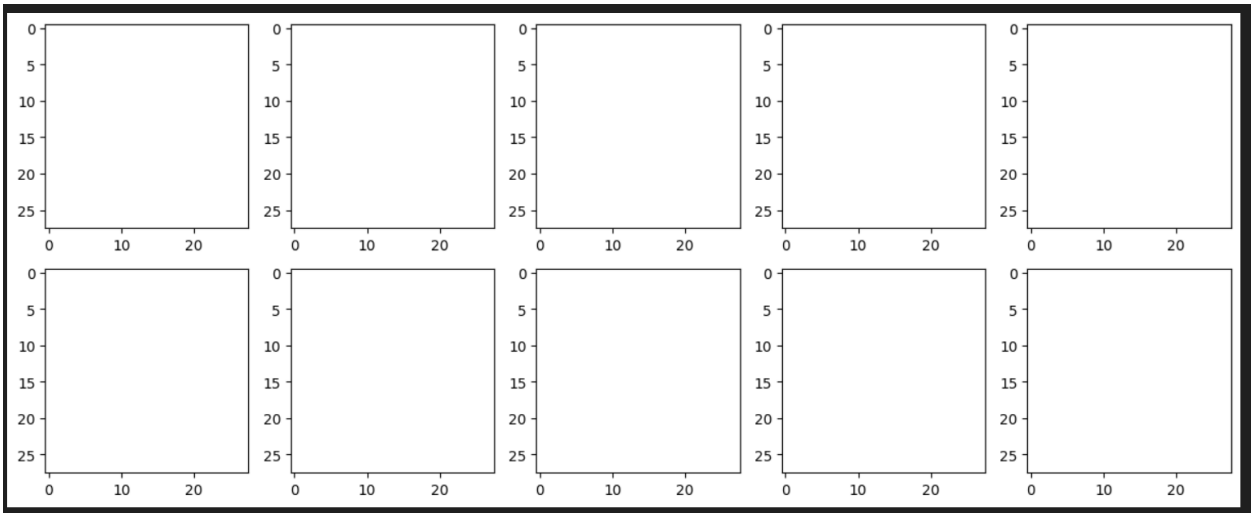
```
These are the decrypted weights in communication round1
0.08999262999999999
0.00837256
0.10519185499999999
-0.00414152000000001
-0.026017599999999995
-0.039578010000000004
0.0011438500000000018
0.03535059
0.028068564999999997
0.028031055
0.009899600000000001
-0.037597559999999995
0.061133129999999994
-0.054689915
-0.055119315
0.0003120149999999988
-0.12351287
0.117209865
0.021701825
-0.008397749999999995
0.056540469999999995
-0.02987779
-0.11592134999999999
0.11641968
...
0.108672575
-0.15384706
0.128062255
```

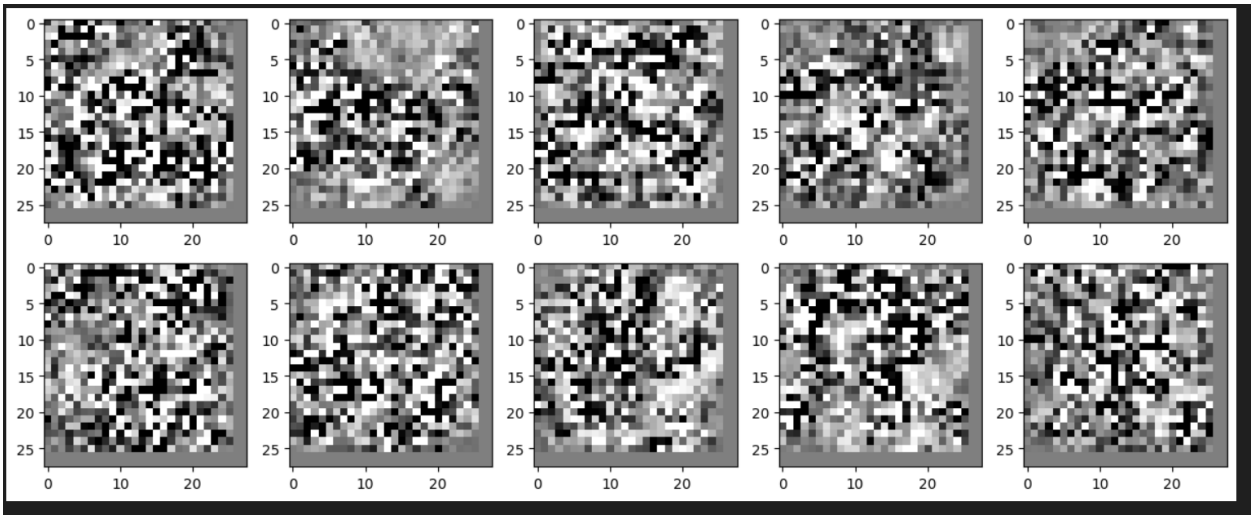
Part2: Check for Defense

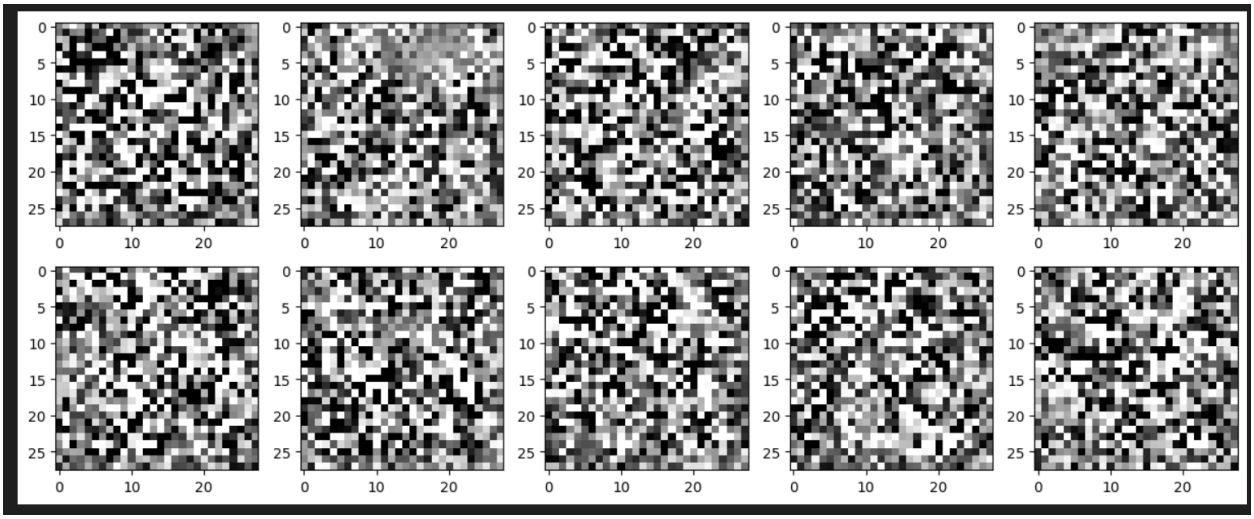
The code realizes a Model Inversion Attack (MIA) on the client model where the attacker does not have access to the encrypted weights of the model. The attack is done over five initializations; black, white, grey, random, and average samples to enable reconstruct the input data from the returned model outputs. For each case, the class gradients are computed, and for the computed gradients, MIA is performed to generate the probable images. The gradients are reshaped and analyzed; the reconstructed images are displayed. This is done for various different initializations, in order to test the model, and to see its behavior over such conditions. The

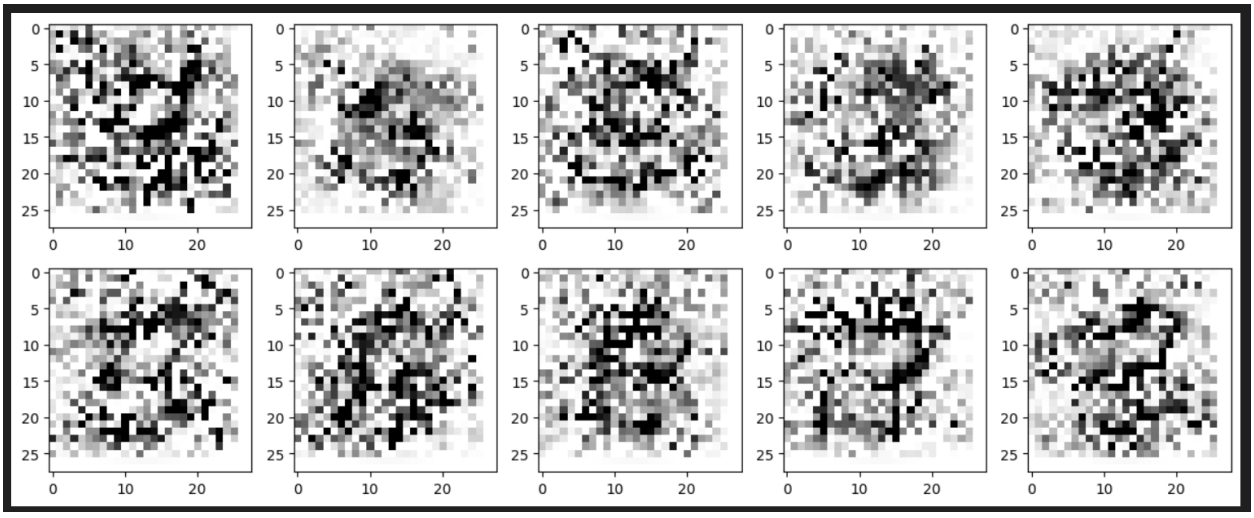
results, shown by a sequence of plots, allow evaluating the performance of the under-consideration model in case where the attacker has no control over the model parameters.







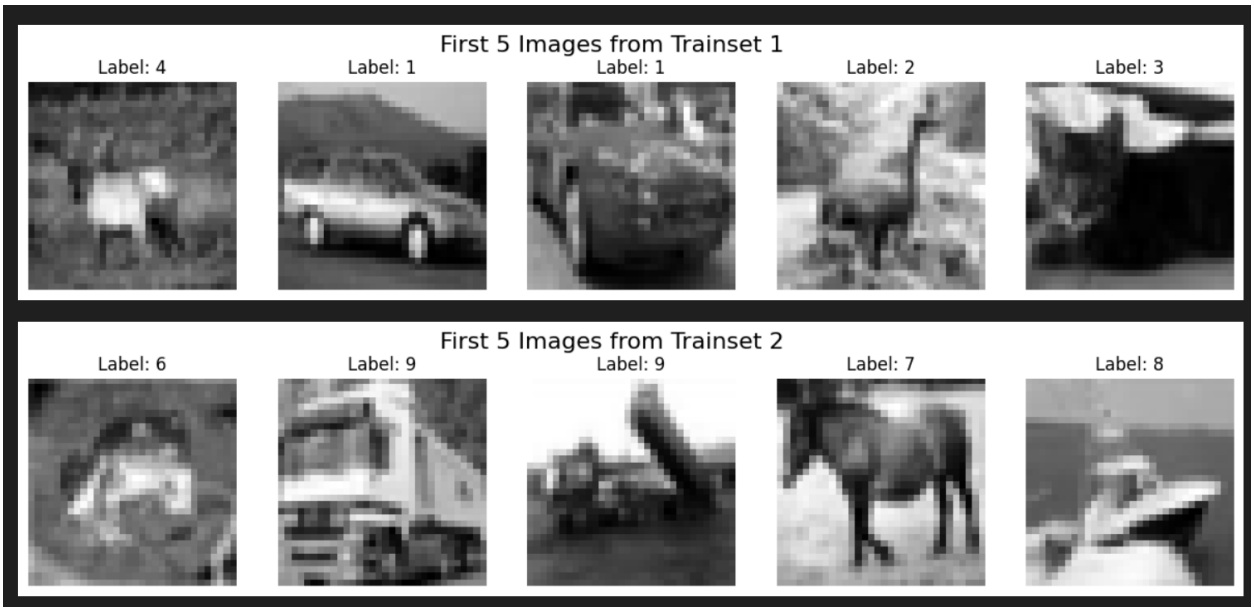




4) File4-Attack On Federated Learning using CIFAR 10 dataset

Step 1- Data Preprocessing

The code starts with data augmentation of the CIFAR-10 where the images are scaled and the color channels reduced from 3 to 1., then the datasets are split into two sets according to the class labels. It first converts the RGB images to grayscale by first taking a weighted sum of the RGB channels and reshaping them by adding an extra channel dimension. The labels are also changed from one hot encoding into the class indices. The dataset is then split into two sets: of which one is a set of images extracted randomly from classes 0 to 4, and the other from classes 5 to 9. Last of all, a function is proposed that would allow one to visualize the initial five pictures from each of the training subsets and which guarantees correct data processing and splitting.



Step 2: Before Federated Learning

The code transfers two CNN classifiers on different portions of CIFAR-10 set, attaining 0.7092 and 0.8184 accuracies of the Test Set 1 and Test Set 2 respectively. For each classifier the model inversion attack of the MIFace method is conducted to infer the information about the training samples. The attack is performed with two initializations: It means a uniform gray image and the mean image of the test set are used as inputs for this RecognizeSoft implementation. The images generated with the corresponding 10 CIFAR-10 classes are given, with the first classifier yields the first five classes(Aeroplane, Automobile, Bird, Cat, Deer) and the second yield the second five classes(Dog, Frog, Horse, Ship, Truck).

Step 3: **Apply Federated Learning**

The proved code performs a simple Federated Learning (FL) with two client models; Both models train themselves with different datasets at their locations and after each round the weights are updated using a Federated Averaging or FedAvg algorithm. They both undergo 10 communication rounds and each client trains locally for 5 epochs then the weights are merged to update a global model. The effectiveness of the model is tested on twofold test messages set for the purpose of underlining better results in succeeding rounds. Finally, the Model produced after a Federated Learning process is under- attacked by Model Inversion Attack (MIA) by applying different types of initialization methods such as Black-box, White –box, Gray box, random, and average sample. For each kind of initialization, the rewarding mechanism is used to obtain the adversarial images for each class among the ten chosen ones; then the attack synthesizes their reconstruction based on the initial model to display the extent to which the attack can

approximate the original data. For each plot, the inferred images of the model corresponding to the given initialization method are represented which shows the weaknesses of the model towards such attacks.