# Anomaly Detection in Graphs

## Defense against Poison Attacks in Federated Learning Systems

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

# Background

1. What is Federated Learning?
2. Why Federated Learning?
3. Security Threat
4. Types of attacks
5. Defense Techniques

# Federated Learning

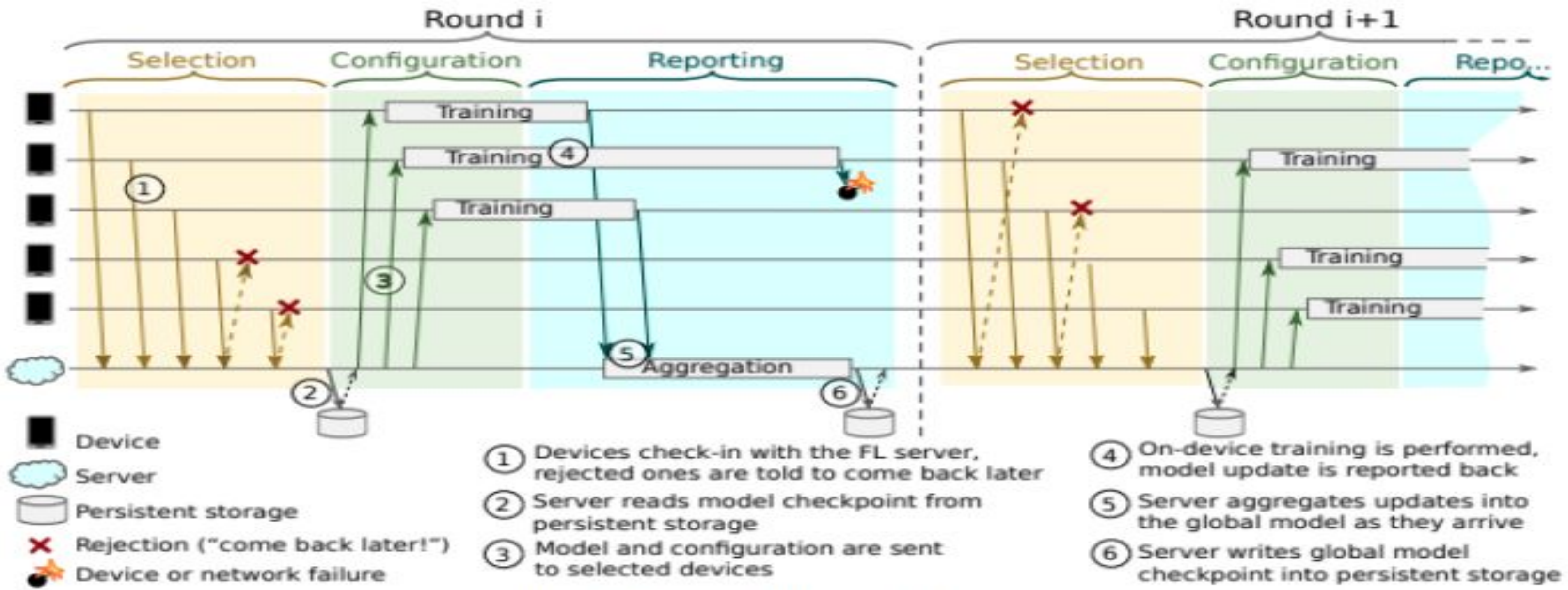



Figure 1: Federated Learning Protocol

# Adversarial Attacks

- Threat Model :
  - In Federated Learning scenario we assume that 80% of existing devices are not malicious.
  - Malicious devices are then injected into the system.
  - API is considered to be honest and cannot be compromised by adversaries.
- Goal :
  - Manipulate learning parameters in such a way that final global model has high errors for a specific class while being undetected from the server.
- Adversary Capability:
  - Adversaries cannot compromise the learning process or the API methods.
  - All they can effect is the training data.
  - Also they do not have any idea about the global models architecture.

# Adversarial Attacks

- Label Flipping attack :
    - Flip the labels of the data points For a particular class on local device.
    - Poisoned data is then feeded to the Federated learning system.
    - Mnp = model without poisoned data, Alpha is the malicious participants availability.
- Attack timing :
    - Attacks done later tend to propagate more into the model parameters than before

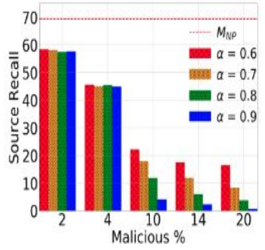



Figure 1: Evaluation of impact from malicious participants' availability $\alpha$ on source class recall on F-MNIST Data. Results are averaged from 3 runs for each setting.

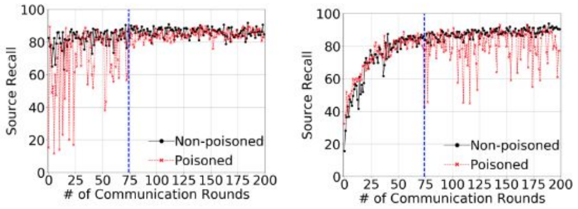



Figure 2: Comparison of recall on early attacks ( before round 75 ) and late attacks (after round 75) on the F-MNIST dataset.

# Defense Strategies

There are 2 prominent works against Poison attacks that we came across:

1. **Golden Test Set:** Systematic poisoning attacks on and defenses for machine learning in healthcare by Mozaffari-Kermani

2. **Neighboring Data:** Identifying and handling mislabelled instances by Muhlenbach

Both these methods can't be used in a Federated Learning architecture.

# Work Done

# Datasets

1. **MNIST**
    a. 60k training samples - 10k testing samples
    b. 28x28 image dimensions - monochrome
    c. Easy dataset, i.e., can achieve over 90% accuracy using simple models
2. **Fashion MNIST**
    a. 60k training samples - 10k testing samples
    b. Same dimensions as MNIST
    c. Tougher dataset, simple models touch about 80% accuracy

The reason for using FMNIST is to show that the model is consistent irrespective of the dataset, and more importantly to understand how to accommodate the challenges posed by tougher problems.

# Attack Strategy (Work done)

- Setup
  - All the attacks had 100 clients, with 20 of them adversaries.
  - There was a total of 60 rounds, and the attack started from the 30th round to give the model some time to learn the distribution of the dataset.
  - 15 clients were selected at random and were trained upon.
  - Attacks were made on 3 models  FFN, CNN and CNN2 (with extra convolution layers)

# Attack Strategy

**Algorithm 1:** Data_Gen Algorithm

**Input**: Training data $\mathcal{D}_{\text{train}}$; Global model $G_t$; Noise samples $\mathcal{Z}_{\text{noise}}$; Label space $Y$.
**Output**: Poison data $\mathcal{D}_{poison}$.
*Initialize generator* **G** *and discriminator* **D**
**for** $t \in (1, 2, \cdots, T)$ **do**
  Set **D** $\leftarrow G_t$
  **for** *local epoch* $e_k \in E$ **do**
    **for** *batch* $b \in \mathcal{Z}_{noisy}$ **do**
      Run **G** to generate sample $x_{\text{fake}}$
      Send generated sample $x_{\text{fake}}$ to **D**
      **if** $x_{\text{fake}}$ *belongs to targeted class* $y_m$ **then**
        Set $x_m \leftarrow x_{\text{fake}}$
        return $x_m$
        **else**
          Update **G** based on Eq. 1
        **end**
      **end**
      **if** $x_m^{label} = y_m \in Y$ **then**
        Assign attacker-chosen label $y_n$ to $x_m$
        Add $(x_m, y_n)$ to $\mathcal{D}_{poison}$
      **end**
    **end**
  **end**
  Return $\mathcal{D}_{poison}$
**end**

- Global model is used as the discriminator
- Xfake data is generated using Generator G.
- Aim of Generator is to generate such samples for which discriminator gives high probability.
- Generated data is flipped and then returned.

# Attack Strategy

**Algorithm 2:** `PoisonGAN` Algorithm

**Input**: Global model $G_t$; Training data $\mathcal{D}_{\text{train}}$; Loss function $\mathcal{L}$; Local epock $E$; Batch size $b$; Learning rate $\eta$.
**Output**: Poisoned local updates $\Delta\hat{L}_t^p$.
*Initialize generator* **G** *and discriminator* **D**
**for** $t \in (1, 2, \cdots, T)$ **do**
  *// Server execution*
  Send $G_t$ to the participants
  Receive updates from participants: $\Delta L_{t+1}^i$
  Update the global model: $G_{t+1}$
  *// Participants execution*
  Replace the local model: $L_t^i \leftarrow G_t$
  **if** *the user type is* $\mathcal{A}$ **then**
    Initialize $D$ by the new local model $L_t^i$
    **for** *each epoch* $e \in (1, \cdots, E)$ **do**
      Generate poison data $\mathcal{D}_{poison}$
      // by using `Data_Gen` in algorithm 1
      **for** *each batch* $b_p \in \mathcal{D}_{poison}$ **do**
        $L_{t+1}^p = L_{t+1}^p - \eta_{adv}\nabla\mathcal{L}(L_t^p, b_p)$
      **end**
    **end**
    Calculate poisoned update: $\Delta L_{t+1}^p = L_{t+1}^p - L_t^p$
    Scale up the update: $\Delta\hat{L}_{t+1}^p = S\Delta L_{t+1}^p$;
  **end**
  **else**
    Update local parameters: $L_{t+1}^i$
    // by running local training algorithm
    Calculate benign update: $\Delta L_{t+1}^i = L_{t+1}^i - L_t^i$
  **end**
  Upload the local update $\Delta L_{t+1}^i$ (including $\Delta\hat{L}_{t+1}^p$) to the central server $\mathcal{S}$
**end**

- This algorithm simply explains the workflow of FL server.
- Each device has a pre decided type, which tells us whether it is an adversary or not.
- Updates are then sent to the server along with the non poisoned updates.

# Defense Strategy

We explore 2 components in the Defense Strategy:

1. Randomization

2. Credit Score

# Randomization

Family of models ensures that the erroneous predictions we make aren't consistent, that is different models make different mistakes violating the one to one assumption made by the GANs

1. Feed Forward Networks
2. Convolution Neural Networks
3. Autoencoders followed by FFNs

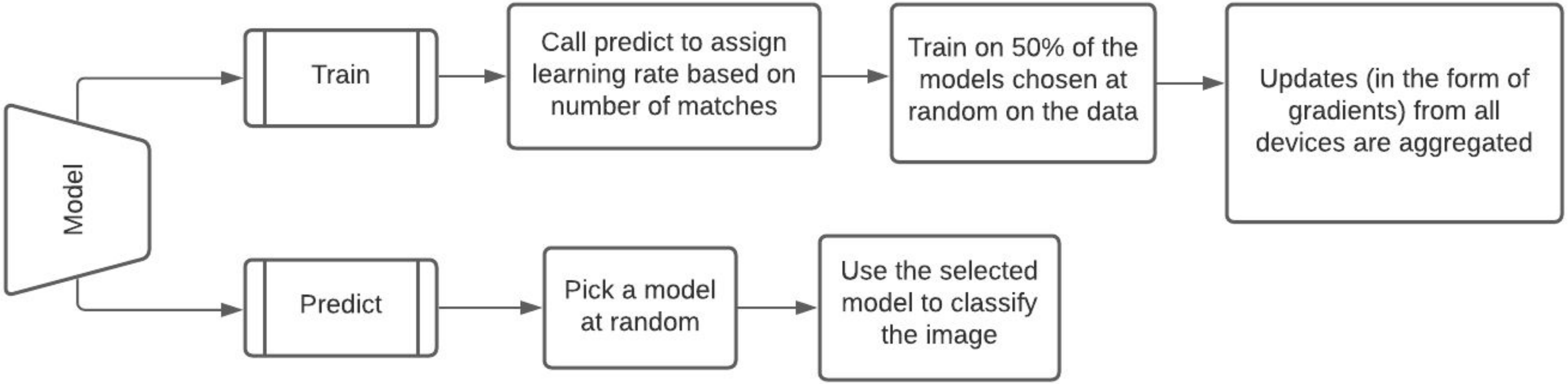
# Credit Score

We devise a system of credit score for each user with the following characteristics:

1. Higher score if the labels being sent are consistent with model's prediction
2. High score if the past updates were consistent, even if the current one isn't
3. Easy to decrease but tough to increase for a user
4. Thresholding to ensure it isn't too small or too large

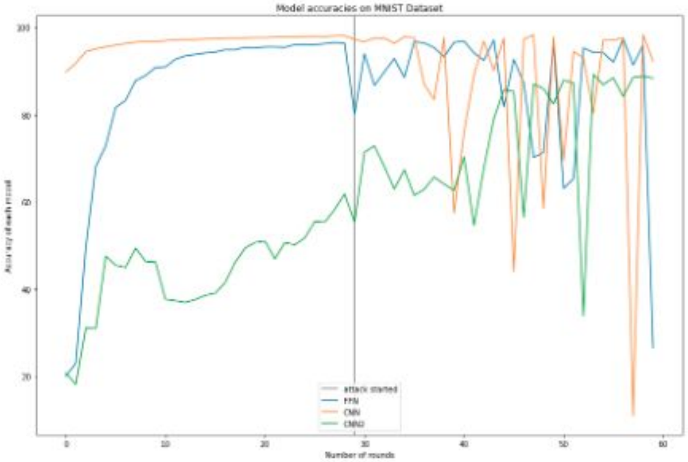
The credit score would be used to define the learning rate for the update being sent by the user, this provides a dynamic learning barrier allowing benign users to occasionally make mistakes/cover our model's inaccuracies.

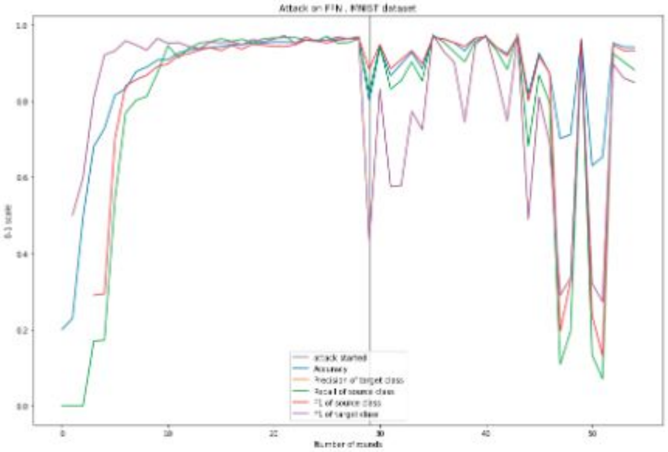
# Model Architecture



# Results and Analysis

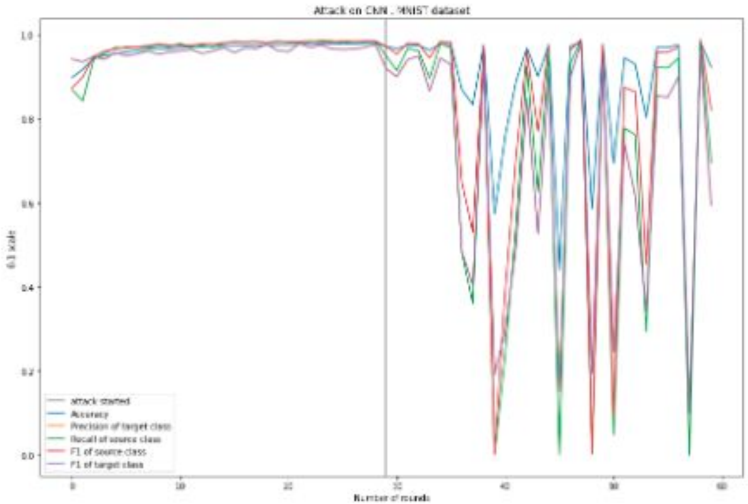
# Attack Results

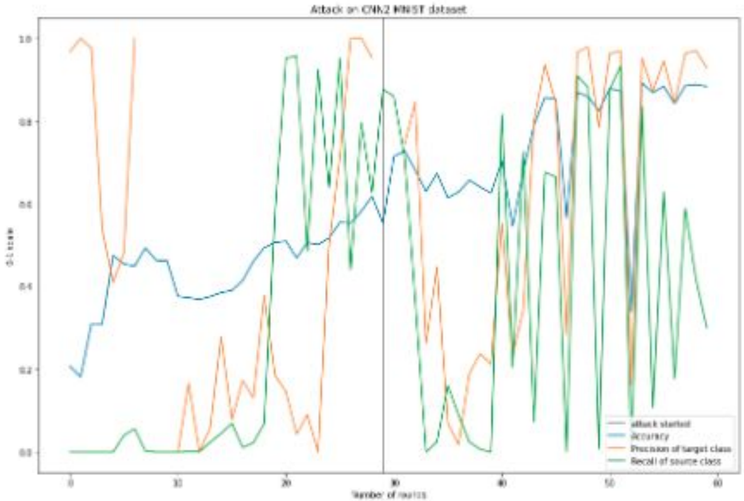



- Precision and recall takes the greatest fall with the attack
- However accuracy still remains somewhat uneffected

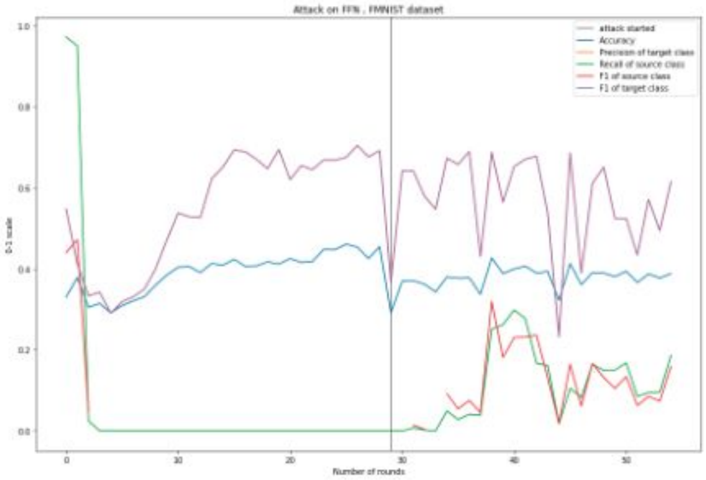
# Attack Results

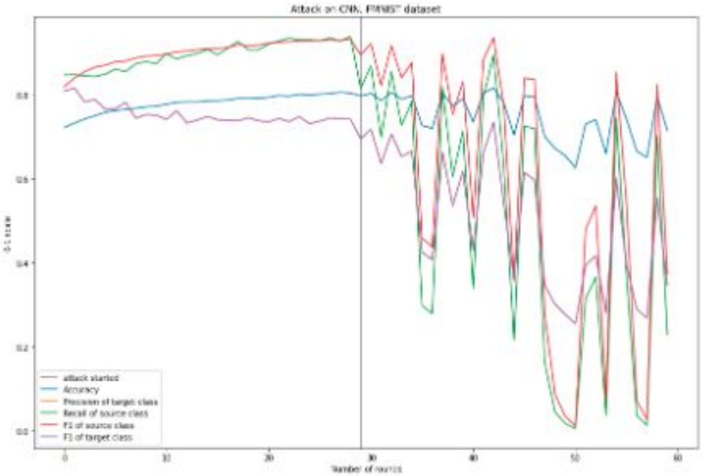



- Similar patterns in CNN
- CNN2 was not able to converge hence produced weird results but accuracy still managed to improve as model learned more.

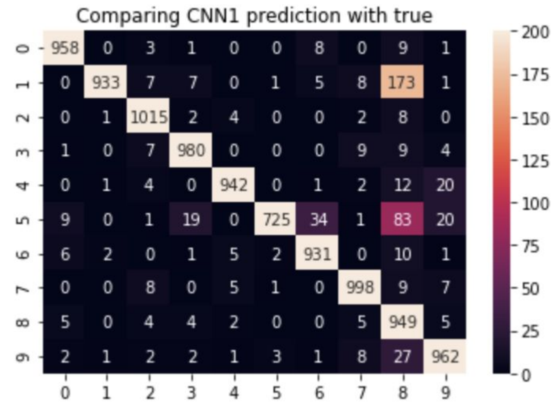
# Attack Results

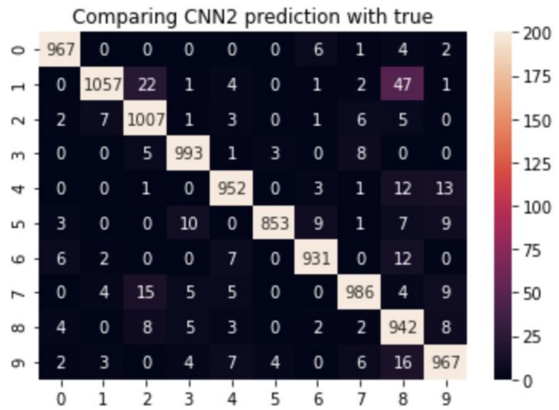


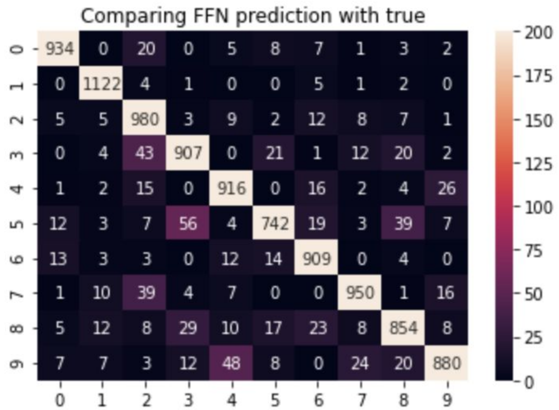



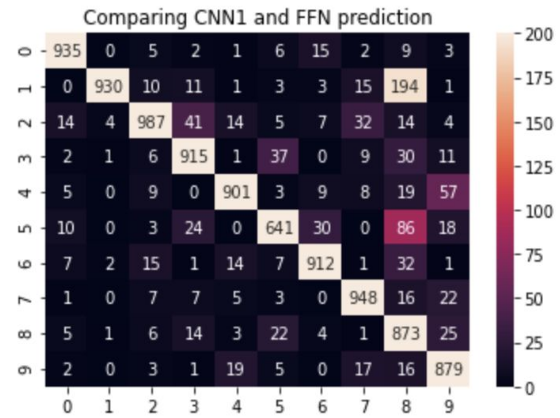

- We tried the same on different dataset (FMNIST) similar patterns are seen later on.
- Because FFN was not able to converge, random results are seen.

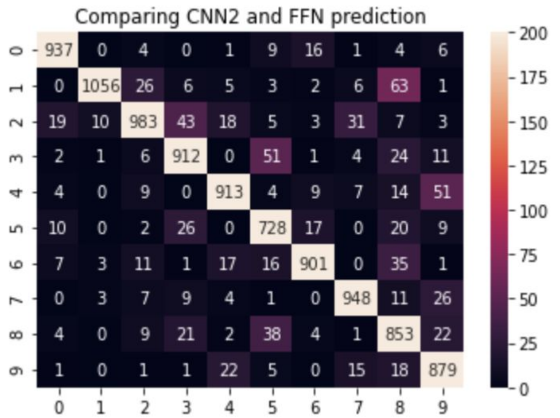
# Defense Results - MNIST

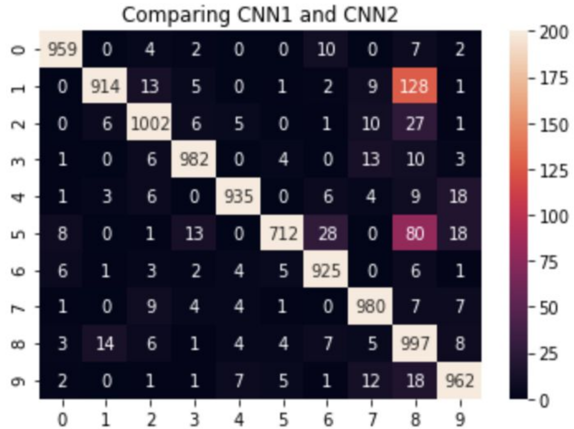


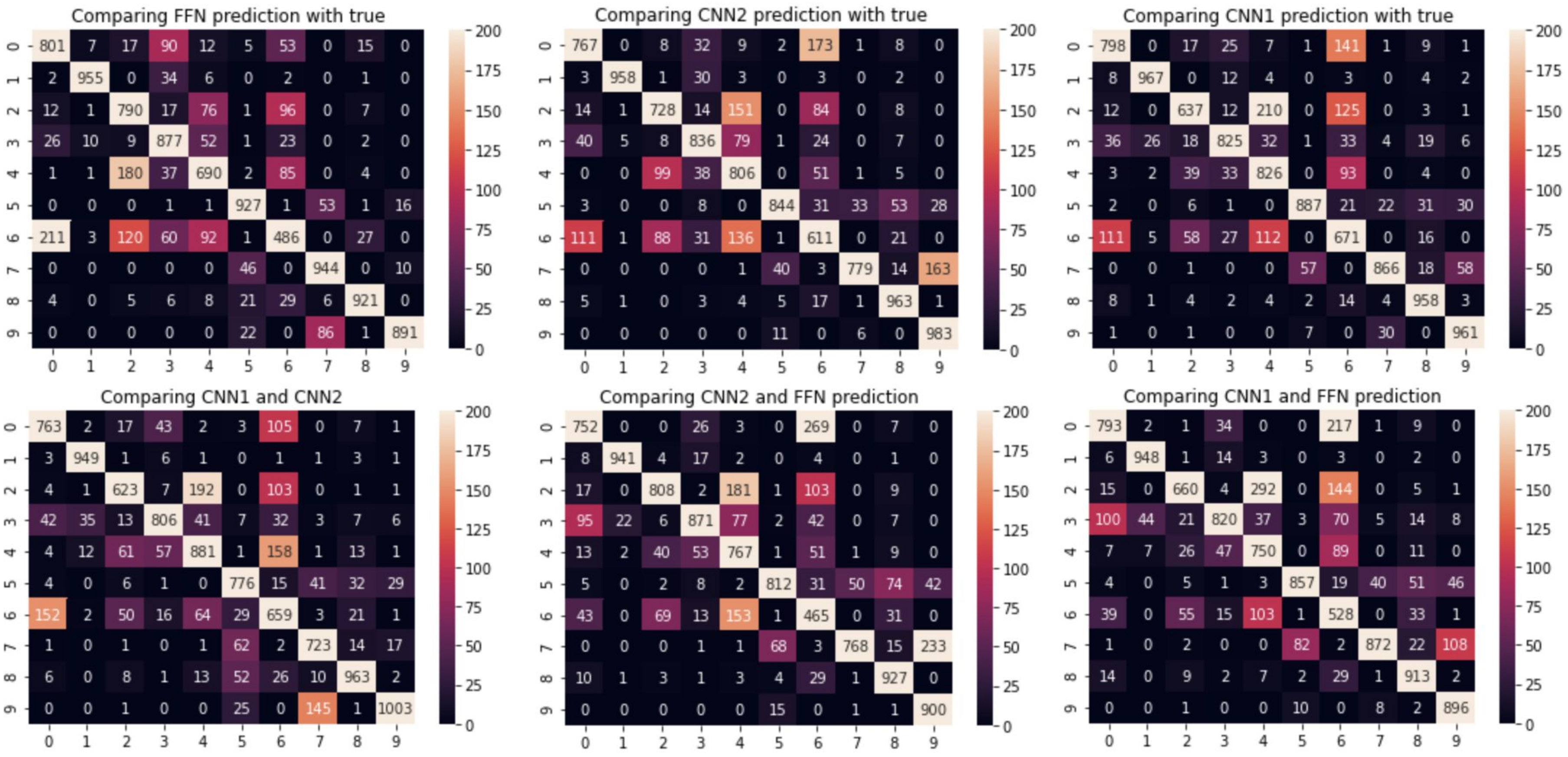



# Defense Results - FMNIST



# Future Work

The following components still need to be looked into:

1. **Credibility Score:** While we have narrowed down the list of requirements, which function to use and relevant weights of components needs to be fine-tuned.
2. **Expanding the randomization family:** We hypothesize that as the number of models increases, the system would become more robust. We are aiming for approximately 5-7 models.
3. **Scalability:** Currently the family of models is being sent to each device which is unnecessary, only the set of models being used for prediction and training should be shared.

# Thank you

Any questions?

# Federated Learning

- Federated Learning(FL) is a type of distributed Machine learning which enables training on edge devices data without invasion of privacy.
- Code comes to your device instead of data to server.
- Updates are untraceable because of in-explainability of deep learning models hidden layers.
- Whole process comprises of three phases:
    - Selection : The server selects a subset of connected devices (typically 100−200 each round) based on internal goals and their availability/networking costs.
    - Configuration: FL plan and the global checkpoint is then passed onto each device with the global model
    - Reporting : Each device trains the global model on their local device. Updates are sent back to the server and merged with the global model using a preselected aggregation technique.