# DV2607 Säkerhet i AI-system, Rapport för inlämningsuppgift

1st Philip Wollsén Ervius
*Student i DVAMI21h*
*Blekinge Tekniska Högskola*
Karlskrona, Sweden
phao21@student.bth.se

2st Amin Afzali
*Student i DVAMI21h*
*Blekinge Tekniska Högskola*
Karlskrona, Sweden
moaf21@student.bth.se

## I. Centrala adversarial input attacker

### A. Beskrivning av adversarial input attacker

Adversarial input attacks exploit vulnerabilities in AI models to change its behavior. There are multiple approaches to these attacks, both in how they identify the behavior that they can exploit and how the attack is executed. It's vital to consider the amount of access the attacker may have, both to the model and the insights to how it operates. This can be done through analyzing the existing behavior (black box). Alternatively, the attacker may already have access to the internal parts of the model.

A main consideration in adversarial attacks in the context of computer vision, is for poisoned images to appear to be unchanged, i.e clean. Thus, sophisticated methods involving specialized noise or trigger patterns may be applies in order to fool the model. The goal may be to achieve a specific classification (targeted attack) or simply to avoid getting a correct classification (untargeted attack). One might generate such a noise by computing the error associated with the target image in regards to the desired classification, and using gradient descent to optimize the noise to achieve the desired classification. An absolutely vital attribute that these attacks rely upon, is for them to resemble clean data. Otherwise, the attack is likely to fail. In other words, adding too much noise is seldom a viable option.

Thankfully, there are many methods available for detecting and mitigating the impact of these attacks. By analyzing the distribution of the image, it's possible to detect it as poisoned (if it's an outlier or has values outside the possible range). Transforming input images by rotating, smoothing them out or applying a random noise of varying degrees may also prove efficient in stopping the attack. This is referred to as pre-processing defense. A third alternative is adverserial training, which aims to make the model robust by training it on poisoned images.

### B. Implementation av adversarial input attack

The chosen attack is a custom version of the Iterated Fast Gradient Sign attack (I-FGSM). The idea is to calculate the gradient in relation to the loss for an image to the desired classification, and apply it stepwise until the image is classified as the target label. It is similar to a gradient decent, except the image is being changed, not the model. By applying the gradient iteratively, the resulting classification gets closer and closer to the desired outcome.

Only one function is required for generating the poissoned data, namely "Create_attack_image" which accepts an image in the form of a tensor, the desired label for the poisoned image and a parameter delta which is the noise multiplier (similar to learning rate). The implementation also includes early stopping, returning the poisoned image as soon as it achieves the desired result.

The function starts by copying the input image. It then enters a while loop, where it calculates the loss for the image in respect to the desired label. Then it multiplies that gradient by delta and adds it to the image. Thereafter, it checks if the poisoned image achieves the desired result and returns if it does. The image is kept between the allowed pixel values 1 and 0.

### C. Defense measures

The defense consists out of two parts: (1) forcing the attacker to apply a very strong noise in order to achieve the desired label. This is done by training the model on poisoned images, making it more robust to the adversarial noise. Increasing the necessary intensity of the noise serves the purpose of increasing the cost of executing the attack, reducing the control over the attack (as having a much higher step size is necessary in order to combat the heightened threshold, which has a higher chance of introducing excessive noise), and making attacks easier to detect.

The second part is to analyze the distribution of images to detect outliers. A secondary model will determine if the input is poisoned. The goal will not be to detect all poisoned images, but only the ones where the noise is substantial.

### D. Implementation of defense measure

A set of poisoned images are generated and used to train the model. Images that contain too much noise are skipped to ensure that the image is not too distorted, since that'd associate the label with the noise instead of the actual picture. The standard deviations and means of RGB values of all poisoned images are calculated. These will be the features for

the random forest. For labels, a threshold is set by observing the degree to which noise distorts the image, after which any image with stronger noise than the threshold is labled 1 (for excessive noise) or 0 (for low amount of noise). The random forest uses this data to learn to identify images that have been poisoned to a high degree.

## II. Federated learning scenario

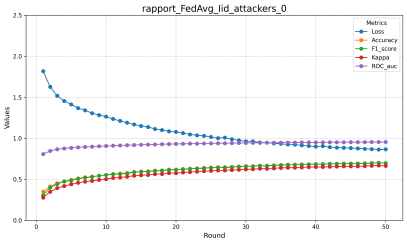


Fig. 1. FedAvg Iid Attackers 0

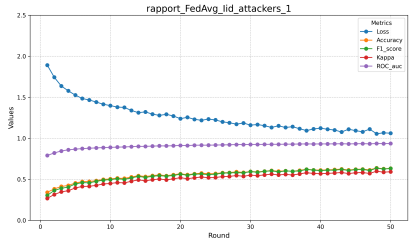


Fig. 2. FedAvg Iid Attackers 1

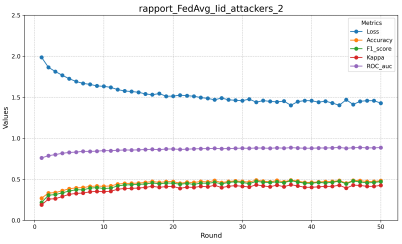


Fig. 3. FedAvg Iid Attackers 2

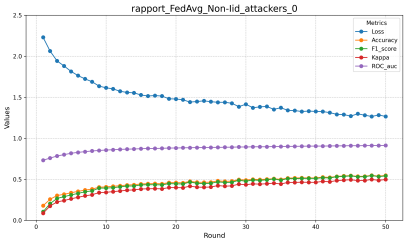


Fig. 4. FedAvg Non-Iid Attackers 0

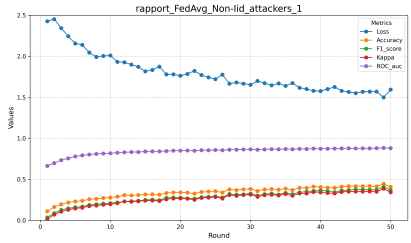


Fig. 5. FedAvg Non-Iid Attackers 1

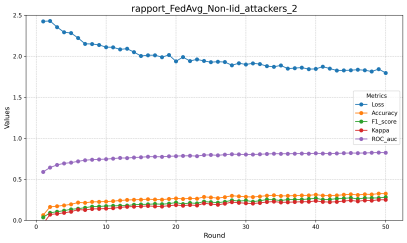


Fig. 6. FedAvg Non-Iid Attackers 2

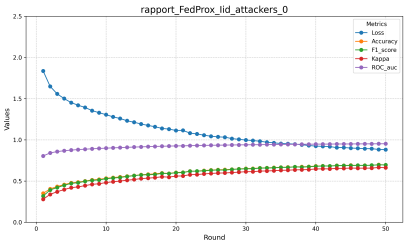


Fig. 7. FedProx Iid Attackers 0

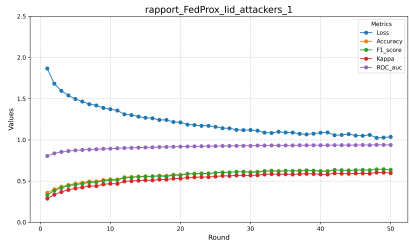


Fig. 8. FedProx Iid Attackers 1

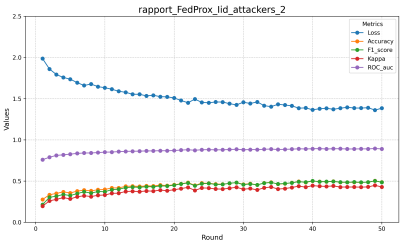


Fig. 9. FedProx Iid Attackers 2

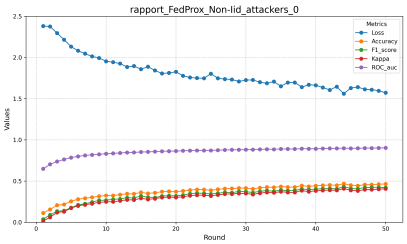


Fig. 10. FedProx Non-Iid Attackers 0

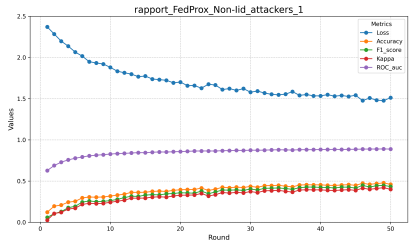


Fig. 11. FedProx Non-Iid Attackers 1

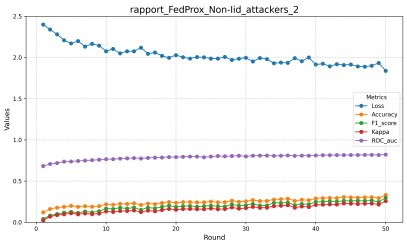


Fig. 12. FedProx Non-Iid Attackers 2

The plots display the performance metrics of the main model after each round. The blue line shows the loss value, purple represents ROC AUC, orange indicates accuracy, red corresponds to Kappa, and green reflects the F1 score.

In this case, the non-IID partitioning refers to Dirichlet partitioning. The model performed worse overall when non-IID data was used, with accuracy dropping by 10% to 25%. While Dirichlet partitioning is often considered more realistic than IID, it negatively impacts the model's performance. This is because patterns within the client models tend to clash more with each other, which reduces overall performance. This trend holds true even when an attacker is introduced.

The effect of changing the strategy from FedAvg to FedProx is negligible when the data partitioning is IID. However, in the case of non-IID data, FedProx performs slightly better than FedAvg. This is understandable because of FedPox being an extension of FedAvg to improve its performance for non-Iid partitioning. Under attack, the improvement from FedProx diminishes.

The attacking clients are performing an untargeted poisoning attack by altering the labels of the training data. This attack causes a performance decrease of approximately 7% when only one client is attacking. However, the performance drop increases to around 14% when there are two attacking clients. This result is quite significant, especially for an attack that has access to 20% to 40% of clients.

The ROC AUC is the only metric that remains unaffected by changes in the strategy, data partitioning, or the presence of attacking clients.