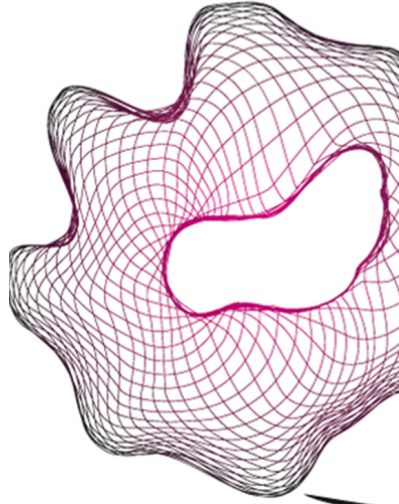


UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science



About the effect of white-box membership inference attacks on federated learning in large networks

L.W.L. Jansen
M.Sc. Thesis
October 20, 2022



Supervisors:

dr. ing. F.W. Hahn
dr. N. Strisciuglio
F. Mazzone, MSc.

External Committee Member:

M. Meijerink, MSc.

Services and CyberSecurity
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

About the effect of white-box membership inference attacks on federated learning in large networks

ABSTRACT

Deep learning algorithms have a wide variety of applications, such as simulating chess, image recognition, and assistance with medical diagnosis. These deep learning methods require a vast amount of data to perform well on their tasks. In applications that involve sensitive data, there are various security and privacy concerns for sharing them for deep learning. Federated learning is a novel decentralized deep learning approach used to protect better the confidentiality of individual training datasets from multiple data owners. In contrast to traditional learning methods that collect all data in a central place to do training, federated learning shares a model throughout a network for participants to use for training on their local data records. Once sufficiently trained, the model weights are sent back to the central entity that aggregates them into a new global model. Recent research shows various ways to extract information from these updates to infer properties about the training data in small networks. Therefore the effects of inference attacks in larger networks are currently unknown, while current applications of federated learning can have thousands of network participants. In this work, we expand the state of the art by studying the attack performance of membership inference attacks from Nasr et al. [16] in a federated network with an increased number of participants. This research shows that when increasing the number of participants to 25, the membership inference attack accuracy increases up to 84.85%.

CCS CONCEPTS

• Security and privacy → Privacy protections.

KEYWORDS

DEEP LEARNING, FEDERATED LEARNING, MEMBERSHIP INFERENCE ATTACKS

1 INTRODUCTION

Deep learning is a set of machine learning methods that use artificial neural networks to simulate a similar learning process, although simplified, to the behaviour of the human brain [8]. This allows these neural networks to interpret and analyze vast amounts of data records. To train deep learning models, there is a need for quality training data. Nowadays more and more data is recorded and hence available for such training tasks. Companies collect user data to provide them with online services they require, for example, Google [21], Facebook or Amazon [20]. The collection and use of personal information are protected by various privacy laws such as the General Data Protection Regulation (GDPR) [6]. Examples of personal data are identifiable data records such as addresses or pictures and medical records.

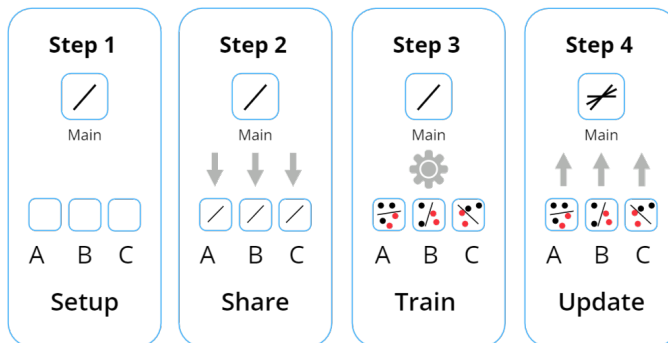


Figure 1: FL Training cycle

Visualization of a Federated Learning training cycle. The main entity shares a model inside a network of learning participants. Then each participant trains their copy of the model on their local dataset. After a predetermined training cycle, the updated model is sent back to the main entity which aggregates the updates into a new model. The process is repeated until either the model converges or a pre-defined training time has expired.

This raises various legal and ethical problems when training machine learning applications on these types of data. The companies are responsible for keeping these records secure and hence are reluctant to share the data at the risk of data breaches. A proposed solution is called Federated Learning (FL)[4]. In figure 1 an overview of the FL training cycle is provided. The cycle starts with a central entity that shares the models with the participants, who in turn start training their model copies on their local dataset. The updated models are then sent back to the central entity and aggregated into a new global model. This solution has been proposed as a privacy-preserving mechanism for settings with high privacy concerns such as medical research [3, 18]. The main benefit of FL is that the data from various sources does not need to be transferred to remote cloud networks to train a global model. Instead, the model weights are shared rather than the data. These weights can be seen as a mathematical aggregation of all the training data that it has been trained on. Recent research [2, 14–17, 19] shows that there is still a large amount of information stored inside the model updates. Hence an attacker that knows how to extract this information summary can infer properties about the training data. For example, Zhu et al. [29] devised a so-called Deep Leakage of Gradients (GLD) attack, where the attacker tries to minimize the loss between the received model updates of another network participants and its own randomly generated dummy gradients. Once this loss is minimized the attacker can reconstruct data that is similar to the original dataset with a high accuracy rate.

In this work, we focus on white-box membership inference attacks. White-box refers to the visibility of the attacker, where the entire model and the intermediate computations are visible to the attacker as opposed to black-box where only the input or output

is visible. With membership inference, the attacker wants to infer if a data sample x was used during training or not. This can lead to privacy leaks as the attacker can thus identify if personal data from a person was used inside a dataset. One of the most extensive works of the last years is the paper by Nasr et al. [16]. They performed both active and passive membership inference attacks in different threat scenarios. An overview of these scenarios and their definitions can be found in appendix A. The scenario we focus on is that of the local attacker, who has access to the entire model and dataset of *one* participant. In the passive setting, the local attacker has access but only observes the target model while in the active setting the local attacker tries to actively manipulate the target model to increase the attack accuracy. Under the assumption that the attacker has prior knowledge of a subset of the target dataset, their attacks show high attack accuracy across all scenarios. From their work, we identified new questions to investigate further to expand the state of the art in FL research.

We contribute by taking a look at the impact of passive and active attacks on federated learning in networks with an increasing number of participants. The current state of the art proved various attacks are possible and effective. However, they are limited in the number of participants used in the FL networks. We specifically answer the question: *What is the impact of different network sizes and data distribution on the accuracy of passive/active membership inference attack models?* As federated learning techniques become more popular [7, 22], it is crucial to identify the impact of privacy leakage attacks in large networks. To answer our main research question, we divided the problem into sub-questions. The first is how can the white-box membership inference attacks by Nasr et al. [16] be reproduced on a large dataset? Second, what are the effects of different data distributions and input model selections on the membership inference attack accuracy? Finally, what are the effects of membership inference attacks on federated learning in large networks?

This document is structured in the following way, in section 2 we provide an overview of the current state of the art. Section 3 provides the necessary background knowledge for our topic. Section 4 states our contributions and methodology. In section 5, we provide an overview of our experiments and their results. In section 6, we show an extended discussion of our results and directions for future work. Section 7 gives a short conclusion of our work.

2 RELATED WORK

During the last five years, the topic of membership inference attacks has been widely researched. This section has an overview of what has been done so far. In 2017 Shokri et al.[19] first showed the vulnerability of deep learning models to membership inference attacks. This attack was in the black-box setting, which means that the attack only had query access to the target model. The attack was launched on the Machine-learning-as-a-service models from Google and Amazon. They created several shadow models of different data distributions to train their attack model. In addition, they concluded that the overfitting of a model increases its vulnerability to inference attacks. Later on, several other works looked at improving the black-box inference attacks from Shokri et al. [5, 9, 25, 26].

Another class of inference attacks called reconstruction attacks, or label inference, is where the attacker tries to reconstruct the original input data based on the gradient[13, 28, 29]. By setting dummy input values and using Stochastic Gradient Descent (SGD) to update dummy inputs towards the gradients of the target, the dummy inputs are reconstructed into the original data.

The attacks were first expanded into a white box setting by Nasr et al. [16]. They investigated white-box inference attacks against centralized and federated learning. Their research found that current black-box attacks were ineffective if applied in a white-box setting. Therefore, they developed new algorithms specifically for the white-box setting. This was a comprehensive study where they tested various attack scenarios. They looked at passive/active attacks and (un)supervised settings. Using publicly available pre-trained models, they show that their attacks show good performance against well-generalized models. Together with research by Pustozeroova and Melis et al., [12, 15, 17] these were the first works with a federated learning setting. They used networks of up to 5 participants to see the effects on the accuracy of their attacks.

Recently, Hu et al. [9] proposed a new version of the membership inference attack called Source Inference Attacks (SIA). They saw that most inference attacks were performed without knowing the source of a data sample while knowing this can cause further privacy issues. In their setting, an honest-but-curious server can infer the source of a data sample without interrupting the training process. The attack tries to exploit the property that the source model of a data sample has the lowest loss value on that sample.

3 BACKGROUND

This section provides the necessary background knowledge for understanding our research domain.

3.1 Machine learning definitions

Neural Networks. In deep learning, we use Neural Networks (NNs), which are layers of functions used to transform training data into a prediction value[1]. The weights of these functions can adjust or "learn" depending on the training data. A training cycle of a single data record is divided into two steps: forward and backward propagation. The network makes all the intermediate computations in the forward pass to arrive at the output prediction value. The input features are put through each layer in the network. Each layer consists of nodes with an input and an output. Each node applies its weight to the received input and puts it through an activation function.

An activation function shows how "active" the input of a node is. An example is the ReLU activation function, which transforms an input to a number between 0 and infinity. An example of a ReLU function is depicted in figure 3. The figure shows that the function outputs between 0 and infinity. An activation function clips the impact of each node on the training. A node that does not provide relevant information for computing the correct prediction value contains a negative value and is therefore clipped to zero.

The output of the final layer is the prediction value, which becomes the input for the error function together with the label. The error function outputs the difference or error between the prediction value and the ground truth. Then the gradient of the error

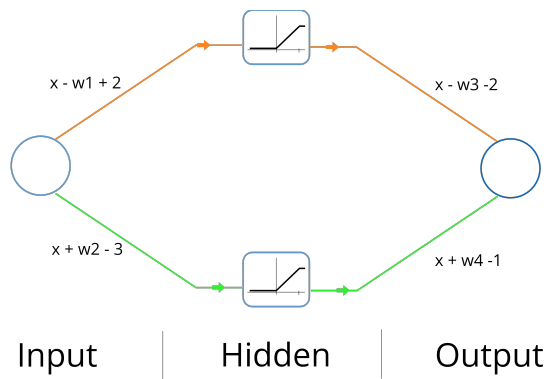


Figure 2: Neural Network Example

The first layer is the input layer with one node, this connects to two nodes in the second layer. In this 'hidden' layer, the inputs are adjusted according to the weights and put through a ReLU activation function. Then the two nodes pass their input to the output layer node, which outputs a prediction value

value for the parameters of each layer, $\frac{\partial E}{\partial W_i}$, is computed. Then, the weights in the network are updated to help reduce the error on this input. This process refers to the actual learning phase or backward pass. We use SGD to compute the contribution of each parameter to the error and adjust the parameters to reduce the error on that data record. Once this is done, we take a new data record and again perform a forward and backward pass, and this way, the network "learns" from the training data. A network only consisting of fully connected layers is called a Fully Connected Neural Network (FCN). Fully connected means that a node in a layer is connected to every node in the next layer.

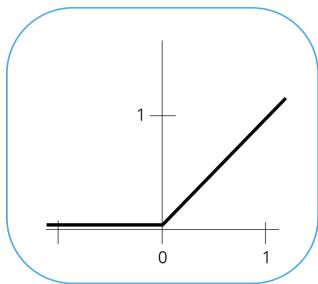


Figure 3: ReLU activation function

Image of the ReLU activation function. The image shows a graph, and the ReLU transforms numbers to fit into the interval of zero to infinity.

Convolutional Neural Network. This research uses another type of NN called a Convolutional Neural Network (CNN). A CNN consists of several convolutional, pooling and fully connected layers. In the convolutional layer, feature extraction is performed on the input, such as identifying edges and shapes. In the pooling layer, this feature map is reduced to improve performance. Finally, the features are put through the fully connected layer. An image is

represented as a series of values in a grid with pixel values indicating brightness or 'activation' levels. There is an extra dimension of the grid for the colour channels. Similarly to the human brain, each feature map in a CNN focuses on a specific image region. The first layers focus on identifying simple structures, such as lines or curves, and the later layers identify more complex patterns, such as faces or objects. [1]

Stochastic Gradient Descent. Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. In deep learning, this is used to optimize the parameters of the neural networks. By finding the gradient of the error function for each parameter, we know the impact of each parameter on the error value. By reducing this impact and thus the gradient, we minimize the error and improve the neural network's performance. Gradient descent does this by taking small steps to increase or decrease parameters to reduce the error. By finding the gradient of the error function, we know in which direction to reduce the parameter to decrease our gradient. [1]

At first, the ideal step size or learning rate is relatively large to reach a minimum fast. Then the step size is dynamically decreased to find the minimum. The ideal learning rate depends on many factors, including task and batch size. An example of these steps can be found in figure 4. In gradient descent, one would first run through all the data samples in the dataset before updating the weights. This may take a long time if the dataset is large. This is where Stochastic Gradient Descent (SGD) comes in. Instead of working through the entire dataset, SGD takes a random data sample and updates the model immediately. This way, the model converges much faster. Stochastic refers to a random value in a distribution, so during training, we pick a random data record from the training set for gradient descent of a certain number of times. SGD is used in deep learning to optimize the parameters of neural networks.

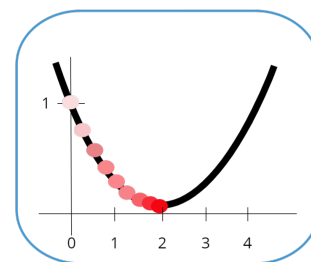


Figure 4: Gradient Descent steps

Example Image to visualize the workings of gradient descent. The line represents the gradient, and the dots represent the steps taken during training.

3.2 Federated Learning

Federated Learning[4, 27] or collaborative learning is a machine learning technique that allows multiple data owners to train a joint model over their private datasets without sharing the latter. This technique shares the model instead of the data, so the participants can train their model copy locally before returning it to the central server where the updates get aggregated. This proposed solution

differs from more traditional machine learning approaches that share both the model and data. In FL, there is a group of k participants with their own local dataset D_i for $1 \leq i \leq k$. A central server keeps track of the current version of the model parameters W . All participants decide upon a single machine-learning task and model. During each training phase, a participant downloads the current global model parameters; the parameters are updated using SGD on their local dataset D_i . Once the local models have been sufficiently trained, their parameters are sent to the central server. This transfer can be done simply by storing the model as a file and sending it to the central server. There, they are aggregated together using the FedAvg algorithm [4]. This algorithm takes the average of every weight of the models from the participants to create a new set of global model parameters. A new cycle starts with the central entity sending a copy of the new global model to all the participants. The training process is repeated until the model converges to an optimal set of parameters.

3.3 Defining the membership inference attack

For our work, we use the white-box membership inference attack developed by Nasr et al. [16]. They provide an extensive description of the attack in their work, but we shall reiterate this description for completeness.

Supervised training. To construct an inference attack model, the attacker needs to identify a meaningful mapping between the model’s behaviour on a given data record and whether the record is a member of the dataset or not. In order to learn this relation, the attacker must have access to some known members and non-members of a dataset. This access is one of our research assumptions, similar to Nasr et al. [16]. This work focuses on the local attacker, which means the attacker has white-box access to the target model f_i of a single participant i . In addition, the attacker knows the membership of some data records from the target dataset D_i and the aggregated model f . In other words, there is an overlap between D_{attack} and D_i , and the attacker knows which records are part of this overlap. Using this overlap, he can train the attack model in a supervised way and use the attack model to attack the rest of the dataset D_i . While training in an unsupervised setting is possible, it was out of scope for this research due to time constraints. Therefore, we do not describe the unsupervised training, but a description can be found in the original paper by Nasr et al. [16].

Attack model. In a forward pass, the attacker can input the target record x through model f and compute all the hidden layers $h_i(x)$, the output prediction $f(x)$ and the error function $L(f(x), y; W)$. These values can be found in the attack model overview in figure 5. The attacker then computes the gradient of the error value for the parameters of each layer $\frac{\partial E}{\partial W_i}$ in a backward pass. All these components, together with a one-hot encoding of true label y , are the input features for the attack model. As the label can also be categorical, it is transformed into numerical data using one-hot encoding. The attack model consists of convolutional neural networks (CNN) and fully connected networks (FCN) at the base. The reason for this is that these layers serve as feature extraction components. Here, feature extraction means that during training, these neural networks learn to focus on a subset of features or properties of the

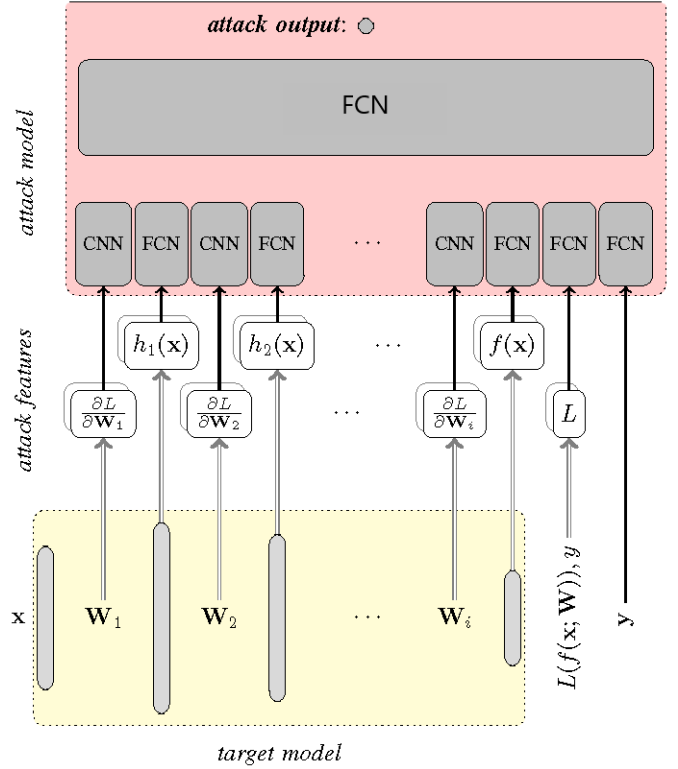


Figure 5: Attack model architecture. Source: Nasr[16]

This attack model architecture is similar to the attack model provided by Nasr et al.[16]. The only difference is that the unsupervised training component was removed in this version as it was not used in this research. It shows the input flow from the target model into the attack model towards the inference attack output.

input data that benefit the prediction value the most. As the attacker receives target model updates at various times during a training cycle, we have several distinct computations of the same hidden layer at different times. Once the attack is started, all computations from subsequent time frames are added to the CNN and FCN layers. This setup is meant to preserve the relation across time for the individual components of the target model. Then the outputs of every CNN and FCN layer get passed through a single FCN into a single score value. This value represents the probability that (x, y) is a member or not, or more formally $Pr\{(x, y) \in D_i\}$. A visual overview of this attack architecture can be found in figure 5.

Passive attack. The goal of the attacker is to correctly predict whether a given data record (x, y) is part of the target dataset D_i . The entire process of the attack goes as follows. After a single round or epoch of training of a federated learning network, the local attacker saves a snapshot of the target model f_i . For this explanation, we assume a single model snapshot, but the attacker can use multiple snapshots as input for the attack model g by simply appending them to the input vector. The attacker loads a target model snapshot to generate inputs for the attack model g . Each data record $(x, y) \in D_{attack}$ is passed through the target model and

the prediction value, the gradients of the final layer, the one-hot encoding of label y and the loss value are collected. These four elements are then appended into four separate arrays that serve as the input vector for the attack model g . The same process is repeated for additional target model snapshots that are used for the attack. Next, inputs from each snapshot are passed into the attack model g . figure 5 shows the flow of this for a single snapshot. In the case of multiple snapshots, the bottom process of the attack model is repeated for each snapshot. For each snapshot, all separate neural network components compute a forward pass. The outputs from every snapshot are concatenated into a single input array for the final FCN. This final component outputs the membership prediction values, which is the probability of membership: $g(x, y) = Pr((x, y) \in D_i; f_i)$. The probability is a value between 0 and 1, with 0 as a non-member and 1 as a member. The accuracy of the attack is then the amount of correctly classified data records, i.e. the membership prediction value is higher than 0.5. When training, the attack model also performs a backward pass to adjust the attack model weights to predict the data records' membership better.

While the original idea by Nasr et al. [16] was to use the weights and gradient from all layers of the target model as input, their work showed that using all hidden layers was equally effective, only the last layer. Therefore, we chose only to select the last layer¹.

Active attack. The active attack is similar to the passive attack only before the membership inference attack is performed. A gradient ascent attack is performed on the target model to extract more information about members of other participants. The active membership inference attack exploits the SGD algorithm's functionality. Nasr et al. [16] do this by reversing the algorithm to perform gradient *ascent* instead of descent. Using the attack dataset D_{attack} , the attacker trains the target model again using the gradient ascent version of the SGD algorithm. In other words, the loss on each data record is maximized instead of minimized for data records in D_{attack} . The weights from the now manipulated target model are sent to the central entity and aggregated into a new version of the global model. Then the next epoch starts, the new global model is shared, and the participants train their target model. Now the attacker can perform the membership inference attack as described in the previous paragraph. The difference to the passive setting is that the attacker manipulated the target model to maximize the loss on the data records from D_{attack} . Regular training continues in the next epoch, and the target model again reduces the loss on all data records from D_i . However, the loss of data records on non-members of D_i is not minimized. As a result, when the attack generates the inputs for the attack model. There will be a bigger difference between the gradients of members and non-members. So the active attack manipulates the model to increase the difference between the gradient of members and non-members samples.

3.4 Evaluation Metrics

Accuracy. The output of the attack model has two classes, "Member" and "Non-member". The accuracy of the attack is the number of correctly classified samples divided by the total number of classified samples. Accuracy is used best when the dataset is balanced,

¹Making this attack quite similar to a membership inference attack in the black-box setting.

meaning all classes have the same number of samples. This metric is a good fit for our experiments, as we have an equal or balanced number of member and non-member samples.

Precision & Recall. [23] In addition to the accuracy, the precision and Recall are also reported. The precision metric is calculated by dividing the number of correct classified members by the total number of samples classified as members. This metric shows how accurate the member classifications are. Recall shows how many member data samples are correctly classified as a member.

4 EXPERIMENT SETUP

This work aims to measure the effect of the white-box membership inference attack scales across different federated learning network sizes. At the start, we identified two key challenges we needed to solve to reach our goal. The first challenge was searching for a larger dataset similar to CIFAR100 for comparison. The second was implementing a simulation of an FL training network that was scalable according to the number of participants. The main reason was to test the effect of removing overlap between participant datasets and performing the attacks with large participant counts. The third is implementing the membership inference attacks from Nasr et al. [16]. On GitHub, we found a small snippet of the code used by Nasr et al. [16]. While this snippet contains the attack model and functions, the code was outdated. Therefore we updated all the code to Pytorch² version 1.12 and expanded it for simulating a federated learning network with a variable number of participants. The details of our implementation will be explained in the following paragraphs, and the code can be found on GitHub³ for reproducibility.

4.1 datasets

CIFAR100. First introduced by Alex Krizhevsky [10], the CIFAR100 dataset with 60.000 32x32 pixel images with 100 classes. Each class has 500 training and 100 testing images. The images are selected from the Imagenet dataset. Imagenet⁴ is a database of millions of images for free to researchers for non-commercial use.

Tiny ImageNet. For our experiments, we wanted to use a different dataset. There were two criteria for this dataset; it needed to contain more samples than the CIFAR100 dataset. Since our goal was to measure the effect of the white-box membership inference attack scales across different federated learning network sizes, we needed enough samples. The second criterion is that it needed to be similar in terms of characteristics to the CIFAR100 dataset to make the comparison between them possible. For example, it needs to contain images and samples with the exact dimensions and number of classes. We chose to use the Tiny ImageNet dataset from Kaggle⁵. The original Tiny Imagenet dataset has 200 classes, but we used a version that clustered the classes into 100 classes. The total dataset contains 100.000 training images, 10.000 validation, and 10.000 test images. Only the training and validation set had truth labels for each data sample. Since we focus on supervised training in this

²<https://pytorch.org/>

³https://github.com/known5/FederatedLearning_MIA

⁴<https://www.image-net.org/index.php>

⁵<https://www.kaggle.com/competitions/thu-deep-learning/data>

work, we use the validation set as test data. In order to match the exact dimensions of the CIFAR100 dataset, we down-sampled the images from 64x64 to 32x32 pixels. Tiny ImageNet has twice the number of data samples as CIFAR100 while having similar characteristics in terms of dimensions and amount of classes. We did not move to an even larger dataset because more data also means more computational power. Therefore, an increase of 100% in the number of samples was enough.

4.2 Implementation details

FL Implementation. Our first contribution is to reproduce the results by Nasr et al. [16] on a new dataset. We chose to use Pytorch⁶, a deep-learning framework for Python. All the code is open-source to make our results reproducible⁷. This framework is similar to the one used by Nasr et al.. To simulate an FL learning network, we implemented a server class that creates N clients class instances that all participate in training. We used the CIFAR100 dataset during our implementation phase for debugging and testing. The dataset contains a training and test set, and the test set is stored on the server side for testing the global model. How we distribute our data among the participants is explained in paragraph *Data Distribution*.

Target model training. For our target model, we used the same version for AlexNet as Nasr et al.. The model is a modified version of the original AlexNet[11] that contains less fully connected layers. Upon initialisation, each client gets a copy of the global model stored in the server class. We then start a training cycle where all clients perform training on their local training data. The server then aggregates the local model updates to create a new version of the global model, which is then shared with all the clients. After this, it is possible to evaluate the global model. Model versions can be stored every epoch to be used later for launching our attacks.

Training cycles. The attack was implemented to be run separately from the target training cycle. During training, the model version at each epoch is stored. This feature allows us to fine-tune the target training and run attacks with different parameters on the same target models without retraining them. The attacker loads a subset of these models to generate inputs for the attack model. For example, we select models at epochs 100, 150, 200, 250 and 300. From each model, we collect the last layer’s predictions, the loss value and the gradients. These are then stored in a list and passed to the attack model. The attack model then outputs a prediction value about the membership of the input data. The number of members and non-members in each batch is equally distributed to avoid bias towards one class.

Data distribution. The dataset is loaded at the start and then distributed across the participants. We can choose to work with and without overlap. Without overlap, we draw D_i random samples without replacement for participant i , such that $D_i \cap D_j = \emptyset$ ($for j \neq i$). Here the dataset size per participant decreases as the number of participants increases. If we allow overlap, We draw D_i random samples with replacement for participant i , such that $D_i \cap D_j \neq \emptyset$ ($for j \neq i$). We randomly draw samples without replacement from the entire training dataset for the member data for the attack data.

The non-member samples are drawn from the test set as these are not used by the target models for training. For the attack model, we create a balanced training and testing dataset. For the last set of experiments in section 5.2, the distribution method for the no overlap setting was adjusted. Now each client gets an equal number of samples per class. The reason for this change was to provide the attacker with only samples from one client dataset. Therefore, measure the footprint of one participant in the global model.

Passive attack. The passive attack implementation works with the attack model we explained in section 3.3. At the start of the simulation, we load a subset of target models, for example, 5. Then, we load the models for epochs 100, 150, 200, 250, and 300 and initialise them. At each batch, all the data is sequentially passed through these models. The last layer output, gradients and the loss value are collected and passed on to the attack model. The attack model then passes each component through a separate neural network and outputs a prediction value. Depending on the batch size, this can be a different number of prediction values. The loss is then calculated using the Mean Squared Error Loss function of Pytorch. Finally, the loss gets back-propagated through the network, and the weights adjust accordingly.

Active attack. The active attack works the same as the passive attack, except that the target models are influenced during their training cycle. When training the target models, the attacker can pass the attack data through its local copy and use gradient ascent to increase the loss, as described in section 3. When training the target models, the gradient ascent attack can be run at every epoch or a specific interval. By tweaking the batch size and the learning rate, the impact of the gradient ascent can be adjusted. In the next section, we elaborate more on our findings for these parameters.

4.3 Resources

In order to run the experiments, we used the hardware cluster of various computers and GPUs provided by the University of Twente for its research groups. Using the SLURM scheduling protocol⁸, experiments were submitted as jobs to the cluster and used the resources available moment or otherwise specified. Therefore, the experiments ran on several different GPUs. As this work focuses on the attack accuracy and not run-time performance, this did not matter in achieving our goal. However, for completeness, we want to specify the minimal specs of the used GPU—a Dell T630 server with an E5-2683-v4 processor and an Nvidia 1080-Ti with 11Gb of RAM. In general, running our experiments on the GPU required around 10Gb and the system required around 10-14Gb of RAM. On average training, the target model took 2 seconds per batch and training the attack model took around 4 seconds per batch.

5 RESULTS

Benchmark. The first step in achieving the goal of this work was to reproduce the membership inference attack from Nasr et al. [16]. Before moving on to other experiments, this step was important to serve as a baseline. These initial runs will be compared to the results of Nasr et al. for the local attacker to see if they are similar. In addition to the two settings similar to Nasr et al., we performed

⁶<https://pytorch.org/>

⁷https://github.com/known5/FederatedLearning_MIA

⁸https://slurm.schedmd.com/sched_config.html

Target model - 4 clients			Inference attack model			
Type	Train	Test	Training member	Training non-member	Test members	Test non-members
Normal	30.000	10.000	5.000	5.000	5.000	5.000
Gradient Ascent	30.000	10.000	5.000	5.000	100	100
Gradient Ascent - large test set	30.000	10.000	5.000	5.000	5.000	5.000

Table 1: Data distribution for attack reproduction

Data distribution for reproducing the attack from Nasr et al.[16]. The large test set row was added later to compare the active and passive attacks.

the active attack with a test set with 10.000 samples for comparison. In table 1 we can see the data distribution we used for these experiments. A total of 4 target models were trained with an overlap in their training data. These target models were trained for 300 epochs with a learning rate of 0.01, 0.001 and 0.0001 starting from epochs 0, 50 and 100. Every epoch, the aggregated global model is evaluated and stored to be used later for the attacks. The attack model learning rate is 0.0001 and is trained for 100 epochs. The attack model is also stored at every epoch as a backlog. All experiments are run with five different seeds⁹ to counter randomness in our results. We took the highest training and testing accuracy from every seed and reported the average accuracy in all the tables. In some experiments, the standard deviation of the test accuracy is also shown, as the accuracy was not very consistent.

The target model training scores are shown in table 2. To compare, 99% for training and 44% for testing were the results on Alexnet with CIFAR100 by Nasr et al.. At the same time, we have 99% training and 28% testing. Both of the test accuracy are quite low, but it is most important that the training accuracy is 100%. The membership inference attack works best with overfitted models [19, 24]. We could not compare the target training scores for the active attack as these were not reported by Nasr et al..

Type	Train	Loss	Test	Loss
Normal	99.99	0.0205	27.99	3.0529
Gradient Ascent	99.99	0.0203	28.06	3.1013

Table 2: Target training accuracy and loss values

Train and test accuracy for the target models. The datasets of the participants overlap. During the active training, the attacker manipulated the target model on epochs 99, 149, 199, 249, and 299. The highest scores are reported, but not on the same epoch. The loss values are associated with the scores.

In figure 6, the effects of the gradient ascent attack on the train/test accuracy can be seen. The rapid increase around epoch 50 is due to decreasing learning rate. The graph shows that the highest test accuracy is achieved around epoch 25, while the highest training accuracy is achieved around epoch 300. The accuracy for the benchmark attacks based on Nasr et al. is shown in table 4. Here the highest train and test, along with the standard deviation of the test accuracy across five seeds, is given. In addition, we provide precision and recall scores. For Comparison, Nasr et al. [16] reported a passive attack test accuracy of 73.1% and 76.3% for the active attack. Our higher score could be because there is no overlap between participant datasets resulting in more unique data samples

⁹Seeds refers to the seed used in random generators. In our framework, we added the option of setting the seeds for deterministic results

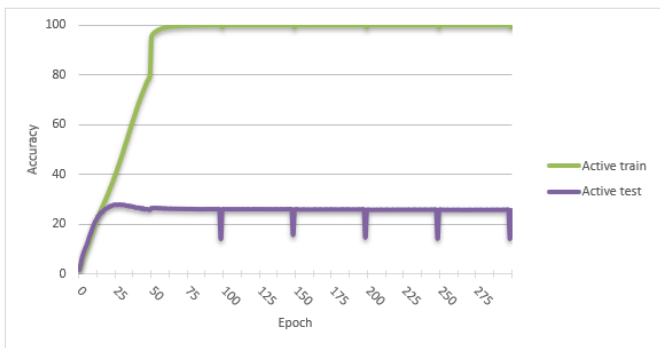


Figure 6: Target model accuracy for the active setting.

Train and test accuracy across 300 epochs is plotted, with the epochs on the x-axis and the accuracy in percentage on the y-axis. At epochs 99, 149, 199, 249, and 299, a gradient ascent attack was performed on the local model. Therefore the train and test accuracy is lower at epochs 100, 150, 200, 250, and 300

to distinguish members from non-members. There is no overlap since we could divide a larger dataset of 100,000 samples instead of 60,000 between the 4 participants.

Type	Train	Test	Precision	Recall
Passive	97.90	80.57 ± (0.697)	0.809	0.800
Active	98.10	82.50 ± (3.482)	0.822	0.830
Active - large test set	97.69	80.41 ± (0.533)	0.808	0.798

Table 4: Reproduced attack scores on Tiny imageNet

Train and test accuracy for the passive and active membership inference attacks. In addition, Standard Deviation (SD), precision and recall are reported. The SD value can be found in brackets next to the test accuracy

Initially, the active attack test accuracy seemed slightly higher than the passive test score. However, there was an inconsistency when looking at the standard deviation of the test accuracy. This inconsistency might be because Nasr et al. [16] tested their active attack on a test set of only 100/100 members/non-members. Therefore we performed the same experiment with a test set similar to the passive attack. The result was a lower but more consistent test accuracy as the standard deviation had decreased.

5.1 Exploring the passive attack

No overlap. From the initial results, the question about the effects of overlap arose. In the paper by Nasr et al. [16], the participants' training data overlapped because the dataset was too small to give

Target model - 4 clients			Inference attack model			
Type	Train	Test	Training members	Training non-members	Test members	Test non-members
Normal	25,000	10,000	5,000	5,000	5,000	5,000
Gradient Ascent	25,000	10,000	5,000	5,000	5,000	5,000

Table 3: Data distribution for attack without overlap

Data distribution for testing without overlap between the client datasets. Since Tiny ImageNet contains 100,000 training samples, each client gets 25,000 samples.

each participant adequate data samples. With the larger Tiny ImageNet dataset, we tried to see the difference between having overlap and no overlap between participants. The data distribution for this setting can be found in table 3. We generated 4 participants, with each 25,000 data samples that did not overlap with each other. All parameters, such as the learning rate and the number of epochs, are similar to the previous experiment. The results for the target training can be found in table 5. The difference between these scores and the target model accuracy from table 2 is that the test accuracy is slightly higher. This difference further supports our previous theory that reducing the overlap between participants results in more unique data samples and, therefore, a higher attack test accuracy.

Type	Train	Loss	Test	Loss
Normal	99.89	0.054	31.21	3.387
Gradient Ascent	99.68	0.078	31.07	3.377

Table 5: Target training scores without overlap

Train and test accuracy and loss values for the target models for the no overlap setting.

The attack scores for the membership inference attack with no overlap between participant datasets can be found in table 6. The no overlap passive attack scores are significantly higher compared to the scores in table 4 with overlap. The active attack also performs better but is not outperforming the passive attack. When comparing the precision and recall scores between tables 4 and 6 the recall score is significantly higher for both the passive and the active setting. The attacks on the no-overlap target models can identify more members than the attacks from the previous experiment. However, it is unexpected that the passive attack scores higher than the active attack. A reason for this can be that the gradient ascent attack changes the model weight so that the model is less capable of distinguishing between members and non-members.

Type	Train	Test	Precision	Recall
Passive	98.13	89.47	0.832	0.989
Active	98.92	85.49	0.793	0.960

Table 6: Attack scores without overlap

Attack scores for the no overlap case. Next to the training and testing accuracy, the standard deviation of the test accuracy across the five seeds, the precision and recall are reported.

Data Distribution. In addition to the no overlap setting, different data distributions are assessed. It was clear from Nasr et al. that the more data the attacker has, the better the model will perform. However, would the train test split matter in this case? Therefore

we set up three cases of train test splits, as seen in table 7. For this experiment, we only performed the passive membership inference attack. These attacks were performed on the same target models used for the benchmark. Although there is only a slight increase in the attack’s test accuracy, it supports the theory that more training data results in higher test accuracy.

Type	Train set	Test set	Accuracy	Precision	Recall
case 1	2,500/2,500	2,500/2,500	80.26	0.800	0.812
case 2	5,000/5,000	2,500/2,500	80.51	0.806	0.804
case 3	7,500/7,500	2,500/2,500	80.96	0.813	0.803

Table 7: Data distribution comparison

Train test split score comparison results, there are three cases with a different train test split. The highest test accuracy, associated precision, and recall are reported for each case.

Model Inference Count. The initial benchmark attack takes five models from different epochs to generate input for the attack model. Models 100, 150, 200, 250 and 300 were selected in the benchmark. Multiple models were selected to preserve the changes across time between members and non-members in the models. However, would the number of models make a difference in attack accuracy? To investigate this, we lowered the number of models used by the attack, starting with the lowest number. Table 9 shows that one or five models do not make a significant difference. This lack of difference can be because the models have already reached a plateau as their training accuracy is around 99%, as seen in figure 6. Therefore, the number of selected models does not significantly influence the attack test accuracy.

Selected Models	Test Accuracy	Precision	Recall
100 - 150 - 200 - 250 - 300	80.57	0.809	0.800
150 - 200 - 250 - 300	80.45	0.808	0.799
200 - 250 - 300	80.54	0.808	0.802
250 - 300	80.37	0.808	0.796
300	80.46	0.807	0.801

Table 9: Input model count comparison

Comparison for using a different number of input models. The highest test accuracy, associated precision, and recall are reported for each case.

5.2 Large networks

This work aims to measure the effect of the white-box membership attack on federated learning in large networks. Therefore we assess the accuracy of the attack when the number of participants in the network is increased. Six sets of participants were tested, that is, 1, 5, 10, 15, 20 and 25 participants. Since having no overlap between

Target model			Inference attack model			
Type	Train	Test	Training members	Training non-members	Test members	Test non-members
Normal	4,000	10,000	2,000	2,000	2,000	2,000
Gradient Ascent	4,000	10,000	2,000	2,000	2,000	2,000

Table 8: Data distribution for membership inference in large networks

Data distribution for the experiments about attack accuracy performance in large networks. The participant dataset is kept constant at 4,000 samples. The attack dataset contains 2,000 members along with 2,000 non-members from the test set, both for training and testing the attack model.

participants performed the best, it is also used here. Therefore, the training dataset was scaled to the largest setting, as seen in table 8. The training dataset was kept constant to remove the adverse effects of decreasing the training dataset when the number of participants increased. All target models were trained for 300 epochs, after which the last epoch was selected to be used for inference. The epochs at which the learning rate is decreased change from 50 and 100 to 100 and 200. For participants sizes 1 and 5, the learning rate starts at 0.01, while the other cases start at 0.1. The learning rate needed to be changed to ensure the target model could overfit the training data. The final change is that the attack data is only sampled from one participant. Therefore we can measure to what extent that single dataset is represented in the global model. The results for training the target models can be found in table 10. As the amount of participants increases, the test accuracy increases as well. The reason for this can be that the global model has seen much more data samples when the number of participants is higher. Therefore it is capable of performing better on the test set. The changes in learning rate do not seem to affect the increase of test accuracy, as the test accuracy already increases by 10% when the number of participants increases from 1 to 5.

Participants	Train	Loss	Test	Loss
1	100.00	0.001	7.65	14.793
5	99.46	0.083	17.90	4.473
10	100.00	0.022	22.60	3.696
15	100.00	0.026	25.90	3.350
20	99.82	0.057	28.05	3.170
25	97.06	0.210	29.34	3.049

Table 10: Target training scores for large networks

Target model training scores for large networks. The table shows the highest train and test accuracy and associated loss values.

Participants	Train	Test	Precision	Recall
1	98.14	51.39 ± (0.646)	0.512	0.580
5	96.38	55.93 ± (4.548)	0.563	0.527
10	93.44	66.60 ± (0.696)	0.738	0.515
15	92.66	74.03 ± (1.440)	0.775	0.678
20	92.51	80.76 ± (0.697)	0.796	0.827
25	90.72	84.85 ± (1.852)	0.792	0.944

Table 11: Passive attack scores for large networks

Inference attack scores for the passive local attacker. Next to the highest train and test accuracy, the standard deviation precision and recall associated with the test accuracy are reported.

In table 11, the scores for the passive membership inference attack on large networks can be found. The number of required training epochs was lowered from 100 to 25 for the attack model, as the highest test accuracy was already achieved in the first five epochs. As a result, the model overfits on the attack data, and the test accuracy decreases. The reason for lowering the number of epochs was to lower the run times of the experiments. The results show that the attack test accuracy increases when more participants are added. This behaviour can be explained by the fact that the global model has seen more data samples during training, as it is the average of multiple models that have seen separate parts of the primary dataset. Therefore, the attack model can make a much clearer distinction between members and non-members. This explanation follows the theory that more data means higher membership prediction accuracy.

Participants	Train	Loss	Test	Loss
1	100.00	0.004	7.64	14.668
5	99.42	0.086	17.35	3.801
10	100.00	0.023	22.93	3.754
15	99.99	0.026	26.06	3.313
20	99.82	0.056	28.02	3.185
25	96.94	0.214	29.32	3.040

Table 12: Target training scores, with gradient ascent attack, for large networks

Target model training scores, with gradient ascent attack, for large networks. The table shows the highest train and test accuracy and associated loss values.

Gradient Ascent attack in large networks. For the active set, the target models were trained using the same parameters as the passive setting. Only the learning rate of the gradient ascent attack was multiplied by 10 for the experiments with 10, 15, 20 and 25 participants, similar to the target model learning rate. The target training scores for the active attack can be found in table 12. These results are similar to the passive target training results in table 10. A reason for this could be that the effect of the active attack is decreased due to the local model updates being aggregated with a higher number of other model updates.

In table 13 the attack scores for the active attack in large networks are reported. While higher in some cases, the active attack scores are not significantly different compared to the passive attack scores. This lack of difference can be to the minimal nature of the active attack, as we implemented it only to occur one epoch before we used the target models and with a small learning rate. Another possible explanation is our theory for the passive attack. With a higher number of participants, the influence of the active attack

Participants	Train	Test	Precision	Recall
1	97.72	51.19 ± (0.612)	0.514	0.451
5	96.78	57.86 ± (0.996)	0.626	0.552
10	96.01	65.53 ± (1.672)	0.711	0.524
15	95.10	74.57 ± (0.992)	0.795	0.662
20	94.23	81.69 ± (0.326)	0.812	0.824
25	94.34	85.16 ± (1.190)	0.796	0.946

Table 13: Active attack scores for large networks

Inference attack scores for the passive local attacker. Next to the highest train and test accuracy, the standard deviation precision and recall associated with the test accuracy are reported.

gets negated as the footprint of a single participants model updates into the global model is smaller.

6 DISCUSSION

This work gives new insights into the workings of white box membership inference attacks on federated learning in large networks of multiple participants. The main conclusion is that the white-box membership inference attack accuracy increases as the number of participants in the network increases. This conclusion contradicts our hypothesis that the accuracy decreases by increasing the number of participants due to the FedAvg algorithm. While this is a similar theory to Nasr et al. [16], one of the key differences here is that we have the additional assumption that the attacker only has access to data from one participant. Our experiment shows that the capability of the attack model to distinguish members from non-members increases as the number of participants increases.

While the attack scores are significant, the setting for the experiments holds the following assumptions: The attacker has access to at least one target model and can use this to compute the forward and backward pass of the model. Furthermore, the attack dataset contains enough members, and the attack has knowledge of which samples are members. The experiment results indicate that removing overlap between target datasets improves the attack model accuracy by around 9% and, therefore, should be investigated as to why that is. An example setup could be to take two participants and gradually increase the intersection of their data samples. Then assess the attack accuracy for each interval. Since the setting with non-overlapping datasets performed best in our findings, we expect the attack accuracy to decrease as the number of overlapping data records increases.

After the last run of experiments, another question arose. Is the attack model predicting that a data record is a specific member of a particular dataset, or has it the general characteristics of a member? The reason for the question is that with a low number of participants, the footprint of the model updates of a single participant is more significant compared to a higher number of participants. Furthermore, one would expect more information in the target model for a high membership inference score compared to a higher number of participants. An experiment setup to verify this would be to train five target models with federated learning and one target model on its own. There is no overlap between the target datasets. The attack dataset only contains member samples from the first five target models. Then the attack is performed on both the global

model of the 5 participants and the stand-alone model. If they report the same accuracy, the model is simply predicting that a sample has the characteristics of a member and not that it is a member of the actual dataset.

In addition, while the active membership inference attack is featured in the findings, it was applied to a limited extent. Finding the ideal gradient ascent learning rate and epochs to launch the attack was out of the scope of this research. Instead, we decided to take a minimal version of the active attack and focus on exploring the passive attack. Therefore it would be interesting to assess further when to perform the gradient ascent attack and how to tune the attack parameters accordingly to maximize the attack accuracy.

7 CONCLUSION

In this work, a framework for simulating attacks in Pytorch was constructed. With this framework, the attacks from Nasr et al.[16] were reproduced and compared. Then the passive attack was further explored by accessing the effect of no overlap between target datasets, exploring different data distributions for the train and test sets and changing the number of input models. Finally, we performed membership inference attacks in networks with increasing participants. The active and passive settings increased to 85% on average with 25 participants by keeping the target datasets constant.

REFERENCES

- [1] Charu C. Aggarwal. 2018. *Neural Networks and Deep Learning*. Springer, Cham, 497 pages. <https://doi.org/10.1007/978-3-319-94463-0>
- [2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2018. How To Backdoor Federated Learning. 108 (2018). arXiv:1807.00459 <http://arxiv.org/abs/1807.00459>
- [3] Anna Bogdanova, Nii Attoh-Okine, and Tetsuya Sakurai. 2020. Risk and Advantages of Federated Learning for Health Care Data Collaboration. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering* 6 (9 2020), 04020031. Issue 3. <https://doi.org/10.1061/ajrua6.0001078>
- [4] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017* 54 (2017). arXiv:1602.05629
- [5] Christopher A Choquette-Choo, Florian Tramèr, Nicholas Carlini, and Nicolas Papernot. 2021. Label-Only Membership Inference Attacks. (2021). <https://github.com/cchoquette/>
- [6] European Commission. [n. d.]. *2018 reform of EU data protection rules*. <https://gdpr-info.eu/>
- [7] Hayden Field. 2020. Federated Learning Could Be the Next Big Thing for Data Privacy. <https://www.emergingtechbrew.com/stories/2020/10/09/federated-learning-next-big-thing-data-privacy> Accessed on: 16-10-2022.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [9] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, and Xuyun Zhang. 2021. Source Inference Attacks in Federated Learning. *ML* (2021). arXiv:2109.05659v1
- [10] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).
- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60 (6 2017), 84–90. Issue 6. <https://doi.org/10.1145/3065386>
- [12] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2020. Feature Inference Attack on Model Predictions in Vertical Federated Learning. (10 2020). <https://doi.org/10.1109/ICDE51399.2021.00023>
- [13] Lingjuan Lyu and Chen Chen. 2021. A Novel Attribute Reconstruction Attack in Federated Learning. (8 2021). <http://arxiv.org/abs/2108.06910>
- [14] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to Federated Learning: A Survey. (3 2020). <http://arxiv.org/abs/2003.02133>
- [15] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *Proceedings - IEEE Symposium on Security and Privacy*, Vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., 691–706. <https://doi.org/10.1109/SP.2019.00029> arXiv:1805.04049

- [16] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *Proceedings - IEEE Symposium on Security and Privacy* 2019-May (2019), 739–753. <https://doi.org/10.1109/SP.2019.00065> arXiv:1812.00910
- [17] Anastasiya Pustozero and Rudolf Mayer. 2021. Information Leaks in Federated Learning. Internet Society. <https://doi.org/10.14722/diss.2020.23004>
- [18] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M. Jorge Cardoso. 2020. The future of digital health with federated learning. *npj Digital Medicine* 3, 1 (2020), 1–7. <https://doi.org/10.1038/s41746-020-00323-1> arXiv:2003.08119
- [19] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. *Proceedings - IEEE Symposium on Security and Privacy* (2017), 3–18. <https://doi.org/10.1109/SP.2017.41> arXiv:1610.05820
- [20] Amazon Staff. 2022. How Amazon works to make machine-learning tools fairer for everyone. <https://www.aboutamazon.com/news/devices/how-amazon-works-to-make-machine-learning-tools-fairer-for-everyone> Accessed on: 26-09-2022.
- [21] Google Staff. 2022. Advancing the state of the art. <https://ai.google/research/> Accessed on: 26-09-2022.
- [22] Google Staff. 2022. Trend data on Federated learning as a search term Worldwide. <https://trends.google.nl/trends/explore?date=today%205-y&q=Federated%20learning> Accessed on: 16-10-2022.
- [23] Kai Ming Ting. 2010. *Precision and Recall*. Springer US, Boston, MA, 781–781. https://doi.org/10.1007/978-0-387-30164-8_652
- [24] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2017. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. <https://doi.org/10.48550/ARXIV.1709.01604>
- [25] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. (9 2018). <http://arxiv.org/abs/1709.01604>
- [26] Oualid Zari, Chuan Xu, and Giovanni Neglia. 2021. Efficient passive membership inference attack in federated learning. (10 2021). <http://arxiv.org/abs/2111.00430>
- [27] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. *Knowledge-Based Systems* 216 (2021), 106775. <https://doi.org/10.1016/j.knsys.2021.106775>
- [28] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. iDLG: Improved Deep Leakage from Gradients. (2020), 1–5. arXiv:2001.02610 <http://arxiv.org/abs/2001.02610>
- [29] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. *Advances in Neural Information Processing Systems* 32 (2019).

A THREAT SCENARIOS

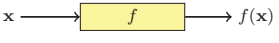
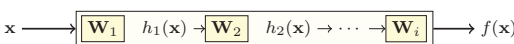
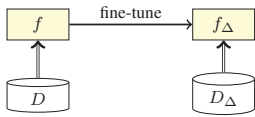
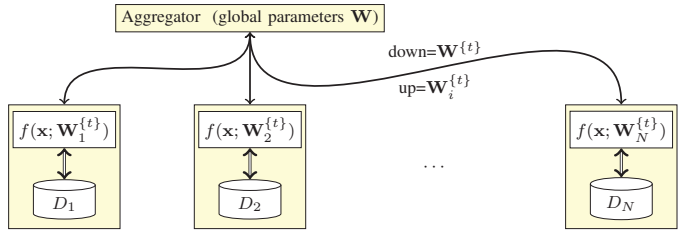
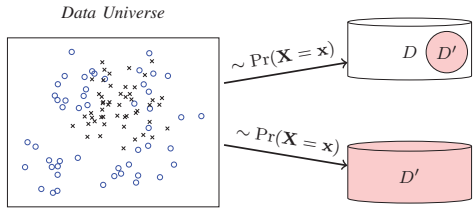
Criteria	Attacks	Description
Observation	Black-box	<p>The attacker can obtain the prediction vector $f(\mathbf{x})$ on arbitrary input \mathbf{x}, but cannot access the model parameters, nor the intermediate computations of $f(\mathbf{x})$.</p> 
	White-box	<p>The attacker has access to the full model $f(\mathbf{x}; \mathbf{W})$, notably its architecture and parameters \mathbf{W}, and any hyper-parameter that is needed to use the model for predictions. Thus, he can also observe the intermediate computations at hidden layers $h_i(\mathbf{x})$.</p> 
Target	Stand-alone	<p>The attacker observes the final target model f, after the training is done (e.g., in a centralized manner) using dataset D. He might also observe the updated model f_Δ after it has been updated (fine-tuned) using a new dataset D_Δ.</p> 
	Federated	<p>The attacker could be the central aggregator, who observes individual updates over time and can control the view of the participants on the global parameters. He could also be any of the participants who can observe the global parameter updates, and can control his parameter uploads.</p> 
Mode	Passive	The attacker can only observe the genuine computations by the training algorithm and the model.
	Active	The attacker could be one of the participants in the federated learning, who adversarially modifies his parameter uploads $\mathbf{W}_i^{(t)}$, or could be the central aggregator who adversarially modifies the aggregate parameters $\mathbf{W}^{(t)}$ which he sends to the target participant(s).
Knowledge	Supervised	<p>The attacker has a data set D', which contains a subset of the target set D, as well as some data points from the same underlying distribution as D that are not in D. The attacker trains an inference model h in a supervised manner, by minimizing the empirical loss function $\sum_{d \in D'} (1 - \mathbb{1}_{d \in D})h(d) + \mathbb{1}_{d \in D}(1 - h(d))$, where the inference model h computes the membership probability of any data point d in the training set of a given target model f, i.e., $h(d) = \Pr(d \in D; f)$.</p> 
	Unsupervised	The attacker has data points that are sampled from the same underlying distribution as D . However, he does not have information about whether a data sample has been in the target set D .

Table 14: Overview of inference attacks categories. Based on observation, target, mode of operation and prior knowledge. Source: Nasr[16]