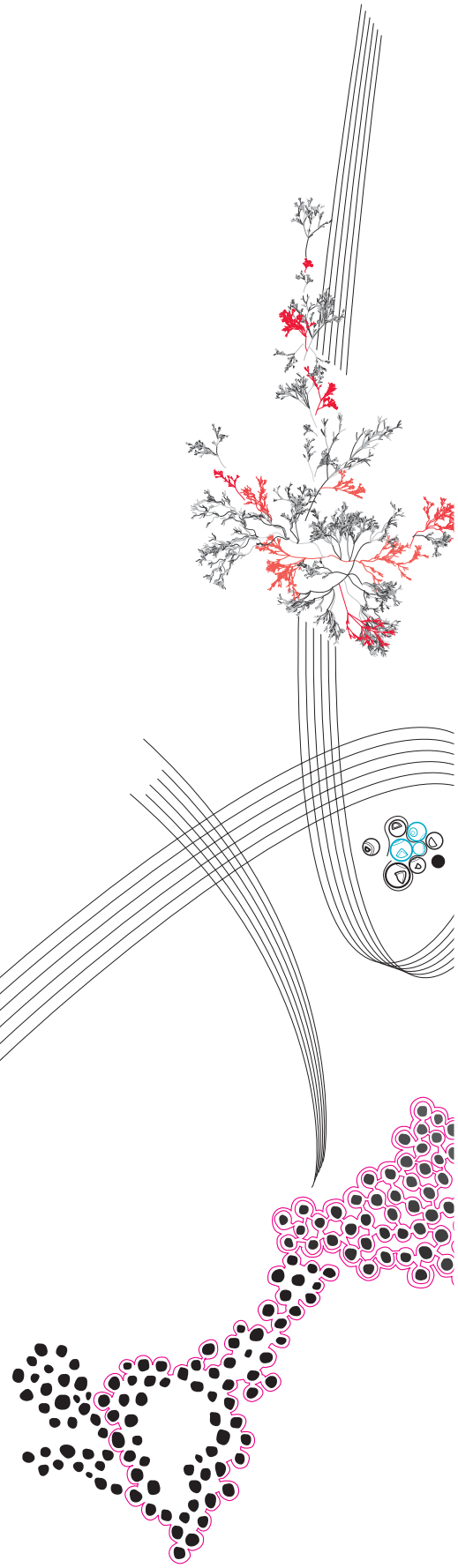


MSc Computer Science
Final Project

The effect of data imbalances on membership inference attacks in federated learning

Bram van Dartel

Supervisors:

Florian Hahn
Marc Damie
External: Ariton Debrliev

October, 2024

Services and Cyber Security
Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

Abstract

During previous years, Federated Learning has gained the interest of many parties who collaboratively want to train a Machine Learning model over privacy-sensitive data, as sharing the underlying training dataset is not needed. Whilst privacy is assumed, as no data is being shared, it has been shown that the setup is susceptible to Membership Inference Attacks, resulting in a leakage of information about the underlying dataset. Previous research has focused on understanding the impact of different setups and parameters of federated learning on attack accuracy, whilst data imbalances between clients have received less attention. Therefore, this research expands the state of the art by studying the impact of data imbalances in label space and feature space on a membership inference attack in federated learning. We define two metrics to measure this degree of non-IIDness in label space and feature space and use this on synthetic data to get an overview of the impact of this imbalance on the attack accuracy of a membership inference attack. Furthermore, we use several real-world datasets on which we use different data splits over the clients to show the impact between balanced and imbalanced data in a real-world setting on the attack accuracy. This research shows that the attack accuracy within federated learning is significantly higher for balanced datasets over unbalanced datasets, while most current research benchmarks their attack on balanced datasets.

Keywords: Membership Inference Attacks, Federated Learning, Non-IIDness, Data imbalances

Chapter 1

Introduction

Training a machine learning model is easier than ever, especially with deep learning, a machine learning method based on the concepts of the brain's inner workings. Deep Learning models, such as a neural network, can learn from data without explicit feature selection or data engineering [8]. This is all possible because it uses many connected neurons, each activated by an activation function. The activation function propagates earlier activations of neurons and recognises patterns in the input data. A lot of (qualitative) data is needed to train a deep learning model to recognise patterns. However, gathering enough data to train a neural network is not always possible. A solution to this can be to combine the data of different parties to train a neural network. Whilst this fits the solution for training a neural network, sharing datasets with another party might not be desirable. Think, for example, of personal data falling under the GDPR or company secrets, which we do not want to share.

In 2016, Google coined the solution: Federated Learning [13]. Federated learning promises to make data sharing unnecessary, as instead of sharing the dataset, it opts to let all parties train the same neural network locally. The clients only need to share the current state of the neural network at most once every epoch. These states of all parties would then be collected by a central server, which aggregates the model using each weight's average and sends back the aggregated, collective model to each client to continue training on. Therefore, as no data has to be shared anymore, federated learning is considered a privacy-friendly way of training a machine learning model. This collaborative learning method has gained much attention since federated learning can achieve better model robustness than in the previous setting [22], called centralised learning.

The problem, however, is that when training a model, we let it remember patterns from the training data we fed it with [18], which it then applies to new given inputs. Even though federated learning shares no training data, it is still possible to gather information about it. Different attacks exist, each uncovering different information from the model [12]. In this research, however, we dig deeper into the Membership Inference Attack (MIA), as it is the most researched type of attack. Furthermore, there are multiple attacks mentioned by Lyu et al. [12] that reduce to an MIA [17], as the goal of an MIA is to determine whether a given datapoint was part of the training dataset or not. If an attacker can resolve this, it would mean the training dataset is not private anymore, meaning the privacy-preserving benefit of federated learning would be broken. Furthermore, Differential Privacy uses the MIA to provide an upper bound to privacy leakage that can occur, making the MIA one of the most interesting attacks [17, 6, 20].

Due to these threats of an MIA, much research has been done on improving the attack itself and finding measures to mitigate the attack's impact [23]. MIAs are not unique to federated learning, as they can be performed on any machine learning model. Therefore, the general setting has already seen a lot of research [6, 11, 15, 5]. On the other hand, as federated learning is emerging, there is a growing interest in this federated setup as well [23, 14, 19, 5] as an attacker has access to way more information than it has in a collective setup.

Even though there is an increase in research, current research has focused on different parameters of the model setup and federated learning setup whilst using the same few datasets to test upon. These few datasets have one thing in common: they are not datasets for federated learning. Most current research uses these datasets and uniformly splits the dataset over the clients, not considering what a real-world setup would look like. In these experiments, the distributions would be identical and independent, called Identically and Independently Distributed (IID) datasets; for real-world federated data, this is hardly ever the case [4]. Moreover, it has already been shown in a regular machine learning setup by Humphries et al. [6] that the degree of IIDness has an impact on the attack success rate of a MIA.

Even where current MIA literature mentions the existence of IID data and non-IID data, they mostly only suggest data be either IID or non-IID [7, 6, 24], whilst the training data can be distributed in different ways between the individual clients. Hence, there are different degrees of non-IIDness. Then, when just concluding a dataset is non-IID, it says little about the degree of non-IIDness then [4]. Furthermore, non-IID datasets have multiple relevant properties, such as non-IIDness in the label space and non-IIDness in the feature space. Current research looking into the IIDness of datasets mostly focuses on only one property: the label space. Again, this property is often only stated whether it has been chosen as IID or non-IID without any degree. Therefore, not all conclusions drawn on MIAs in federated learning in current research can be assumed. This is because there are different assumptions about non-IIDness for different real-world contexts. There is no research yet on the relation between the *degree* of non-IIDness and the attack, whilst there are signs from centralised machine learning setups that one might exist.

The research of Humphries et al. [6] comes closest to ours, but they used a regular machine learning setup. In their setup, they assumed an attacker with less useful information on the target model and dataset than we can accept for a federated learning setup. In the federated learning setup, the attacker has insight into all iterations of the target model. Moreover, as the attacker is a client in the protocol, it also has part of the dataset on which the target model is trained.
With our extra knowledge of the target model, we look for a relation between the non-IIDness of the partitioning of the dataset over the clients and the success of an MIA. We use two non-IID properties: non-IIDness in the label space and non-IIDness in the feature space. To report on a possible relation, we will first answer the question: *How can we measure the non-IIDness of a partitioned dataset?*, followed by *what is the impact of different non-IID properties on the MIA in federated learning?*. Last, we answer the question *to what extent are real-world federated learning datasets non-IID?*

The rest of the document will go more into depth on this research, starting with the

preliminaries needed in Section 2, followed by our approach of measuring the IIDness of a dataset in Section 3. Section 4 will then go into depth on our experiment setup, and then we look into the synthetic data generation method in Section 5. Afterwards, our results will be elaborated upon in Section 6, after which we relate our results with current research in Section 7. Then, in Section 8, we discuss our limitations and give direction to future works, closed by a conclusion in Section 9.

Chapter 2

Preliminaries

This section elaborates on the preliminaries used to perform this research. In later sections, we build further on the preliminaries explained here.

2.1 Machine Learning

The required knowledge of machine learning mainly lies within deep learning. In our case, we will use a neural network as a basis, together with Stochastic Gradient Descent (SGD) to train the model.

2.1.1 Neural Network

The model we will use in this research is a neural network, part of the deep learning category within the machine learning categories. These deep learning models are designed to work like beliefs about the inner workings of the human brain: neurons. The neurons in a neural network can be activated with a linked activation function, activated by an input given by either a datapoint or a neuron earlier in the network.

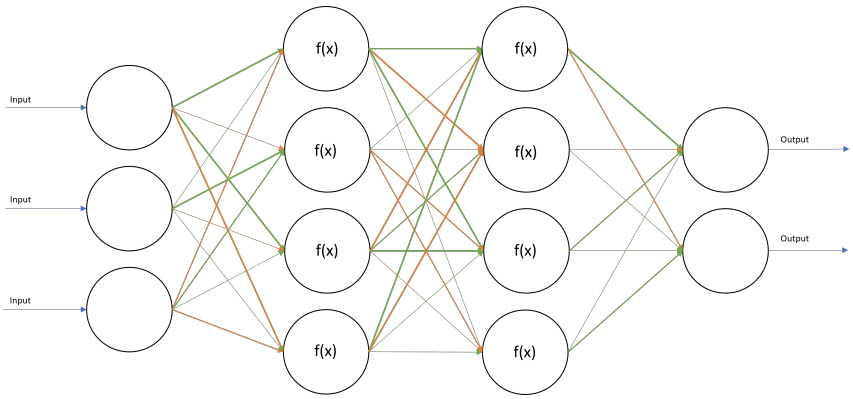


FIGURE 2.1: A schematic view of a neural network. On the left, three different inputs are given. This input will propagate to the first hidden layer of neurons with certain weights, denoted by the colour and thickness of the connecting lines.

The advantage of a deep learning model is that we do not need to select relevant features to train the model, as that is something the model will learn itself [8]. Furthermore, deep learning is a good solution for high-dimensional data, such as image or speech recognition. A neural network can have many inputs, which can be reduced in the different layers. In

the end, it is common to have the number of outputs equal to the number of classes: the probability of an input being in each class. The neurons in each neural network layer are connected, and each connection has a certain weight. This weight can be adapted during training to predict better the correct result based on the given input data. Next to the weight, a bias can be set to affect the degree of activation of a function, which can also be influenced during training. Figure 2.1 shows a neural network with three inputs, two hidden layers of four neurons and two outputs.

Training a neural network
Training a neural network is done by updating the weights of the different connections between the neurons to give the correct output for a specific input. When training the network, we must quantify the model error when providing a particular input, called the loss function, denoted L. This function describes the mismatch between prediction and ground truth [16].

We aim to minimise the model error by using the loss function's derivative, often called *gradient*, to find the lowest output value. As the loss function is the difference between the prediction and the ground truth, and the prediction is a formula in which all neurons are considered, finding the global minimum of this gradient can be infeasible. Therefore, training is often done using Stochastic Gradient Descent, which iteratively optimises the neural network and lets the model converge to a local minimum of the loss function.

Stochastic Gradient Descent
To optimise the loss function, we need to find the gradient of the loss function L with regard to every weight $W_{i,j}$ in the neural network, where (i, j) are the row and column in the network. This is denoted as $\frac{\delta L}{\delta W_{i,j}}$. When having this gradient, using different input datapoints and a chosen (small) learning rate, we can tweak the weights such that the loss function minimises and our model performs better. In Stochastic Gradient Descent, this is done in iterations where a random datapoint is selected from the training dataset. By doing this, the number of calculations that have to be performed decreases with regard to Batch Gradient Descent, where the calculations are run over all datapoints. Because of these iterations, a small learning rate, α, is applied to the weight updates. This is done so as not to overshoot the local minimum. An update of weight $W_{i,j}$ is then done according to

$$W_{i,j}^{t+1} \leftarrow W_{i,j}^{t} - \alpha \frac{\delta L}{\delta W_{i,j}},$$

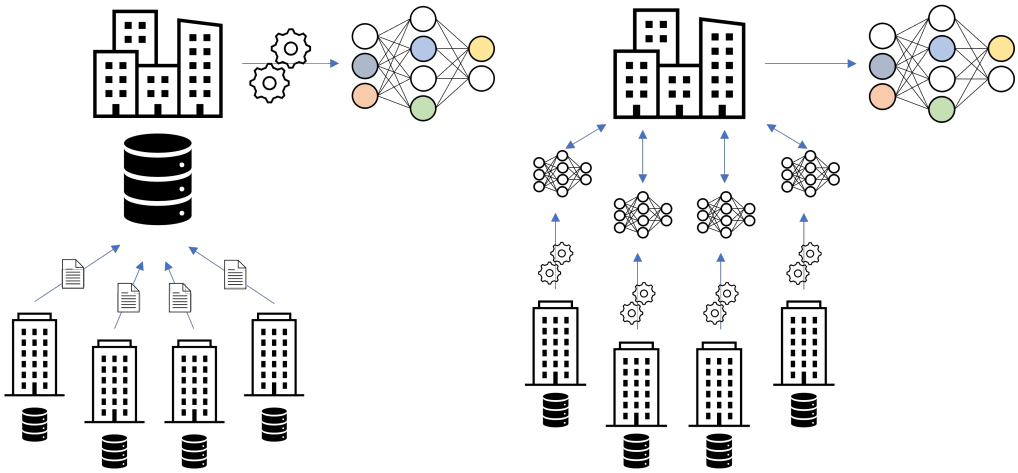
where t is the iteration and α the learning rate.

2.2 Federated Learning

As mentioned, the 'regular' setup for collaborative learning is Centralised Learning, where the data is gathered collectively, and the model is trained on this data by one central party. In federated learning, however, the model is trained on the local dataset per client, and then the model weights are aggregated by one central party. This is shown in Figure 2.2b.

2.2.1 Training a Federated Learning model

To train the collaborative model, the server starts by initialising a model with arbitrary weights and sending this model to all clients. Then, the federated learning protocol starts



(A) Example of a Centralised Learning setup. All clients have their datasets and share this with a central server. The central server performs the training on the entire dataset of the clients and outputs a model.

(B) Example of a Federated Learning setup. All clients perform training on their dataset and share their parameters with a server. The server performs the aggregation algorithm on the received parameters and outputs a collaborative model.

FIGURE 2.2: Two different setups to collaboratively train a model: On the left Centralised Learning and the right Federated Learning.

in iterative rounds, where the model is trained increasingly in each round. Two algorithms are mainly used, namely FedSGD and FedAvg [13]. As we will work with FedSGD, this will only be highlighted.

In the FedSGD method, every client trains the model with one round of Stochastic Gradient Descent as described before. After this training, the client sends back the gradients to the server, which aggregates them and applies them to the global model. After the global model is updated, it is sent to the clients again, repeating the process. This algorithm is inefficient communication-wise, as all weights are shared after each round of training; however, it guarantees finding a local minimum, meaning it will converge over time.

2.2.2 Dataset partitioning

As Federated Learning has multiple clients with a part, also called partition, of the global, overall dataset, each partition is different, with different model properties, benefits and challenges. We mainly distinguish partitioning into two categories: horizontal FL and vertical FL [21]. We will focus on the horizontal FL setup.

In horizontal FL, the key idea is that multiple datasets have overlapping features but almost no overlapping users. Therefore, horizontal FL increases the user sampling size, which overcomes the problem of datasets that are too small and do not generalise enough to perform well.

On the other hand, vertical FL combines different features of the same users to gain more information about users in the dataset. By combining this data, the aim is to enhance the model's ability. The advantage is increased feature dimension so that better predictions can be made.

2.3 Inference Attacks

In our research, we will be focusing on inference attacks on the Federated Learning framework. As different inference attacks, such as a property inference attack, can be reduced to a Membership Inference Attack (MIA), we will focus on this attack type. Because of this reduction, the MIA can be seen as a fundamental privacy attack [17].

A Membership Inference Attack (MIA) aims to determine whether a datapoint (x, y) is a part of the training dataset D of a machine learning model. This attack can reveal a lot about a datapoint depending on the dataset type. If this were a dataset full of people with a cardiac disease, and we could see if a datapoint x was a part of the training dataset, we could conclude that person x has a cardiac disease, therefore breaking their privacy. Intuitively, the attack works with the idea that the trained machine learning model, called the target or victim model, responds differently to an input datapoint it has already seen than to a new datapoint. Therefore, it is possible to train a new model, which we call the attack model, that can observe the different responses on datapoints, predicting whether a given datapoint was part of the training dataset. As we use the specific attack performed by Nasr et al. [14], we will go more in-depth into the inner workings of this particular attack.

2.3.1 Membership Inference Attack by Nasr et al.

Nasr et al. [14] have implemented several MIAs, amongst others, an attack for a passive attacker with white-box access to the target model in a supervised attack setup. This setup is relevant as we consider an attack where each participant adheres to the protocol and does not interfere actively by modifying responses that are not true; thus, it is a passive attack rather than an active attack.

In this FL setup, the goal is to determine whether a given target datapoint (x, y) was a member of the training dataset D of the target model f. This target model is the final model, which all clients have trained collaboratively. However, as the attacker is a client in the FL protocol, it can observe all intermediary rounds, namely T different iterations of the model. Using these T different iterations, the attacker has more information than an attacker of only the final model would have, which we call the white-box access the attacker has. Furthermore, the attacker also has its partition of the dataset D, e.g. $(x, y) \in D$, meaning it already has information about the data distribution of the training dataset. This, however, is only about the training dataset of the attacker itself, not the overall distribution amongst the clients. This is the same in the case of IID datasets, whilst the distribution per client might differ for non-IID datasets. Using their dataset, the attacker can train an attack model with data the target model f has already seen, also called a supervised setup.

With all these details about the target model f and the dataset D collected, the attacker can stack the attack features it has accumulated over the T iterations it has seen of the target model f. The attack features Nasr proposes are the derivatives of the weights, $\frac{\delta L}{\delta W_i}$, the loss function L and evaluations of neurons in all layers, $h_i(x)$. Nasr used an attack model, being a neural network, with Fully Connected Layers (FCN) for the features $h_i(x)$, L and target label y and components of a Convolutional Neural Network (CNN) for the gradient $\frac{\delta L}{\delta W_i}$. The correlation of the gradients in each activation function can be captured using the CNN. The final output of the CNN and FCN are reshaped into a flat vector and concatenated before being inserted into an FCN component with multiple layers, which

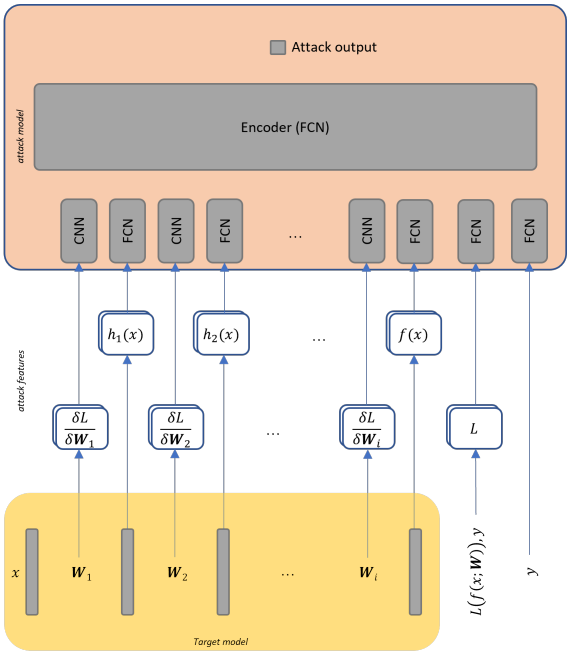


FIGURE 2.3: Schematic overview of the attack by Nasr et al. [14].

outputs a single score predicting the membership probability $Pr(x \in D)$, of the input data being in training dataset D. Figure 2.3 is a schematic overview of the attack by Nasr.

Chapter 3

Measuring non-IIDness

In our research, we rely on measuring the non-IIDness of data partitions between different clients. In our measurement definitions, we focus on a setup with only two clients in the FL protocol.

Variable	Definition
X, Y	Distributions of datapoints
x_i, x_j	Datapoints from partition i and j
y_i, y_j	Class of datapoints from partition i and j
C	Class a datapoint belongs to
P	A partition of the dataset. All partitions together form dataset D
D	Dataset containing partitions belonging to clients
Pr(x)	Probability of x

TABLE 3.1: Definitions of variables

3.1 Data dependencies and distribution

Data dependencies and distributions play an important role. In this FL context, all data dependencies and distributions are related to the partitions of the different clients, as a known global dataset does not exist in practice.

3.1.1 Defining IIDness

For a dataset to be considered Independently and Identically Distributed (IID), the following two definitions need to hold [25]:

- $\forall i \neq j \ Pr(x^{(i)}, x^{(j)}) = Pr(x^{(i)})Pr(x^{(j)})$ (Independently distributed),
  This means that for every i^{th} value and j^{th} value in the dataset, the probability of both x^i and x^j together is equal to the probability of x^i and the probability of x^j separate, meaning that the data points are independent.
- $\forall i \ x^{(i)} \sim D$ (Identically distributed),
  This means every datapoint in the dataset is sampled using the same not-fluctuating probability distribution.

Even though these properties might hold for a synthetic dataset, a real-world dataset is unlikely to qualify as an IID dataset immediately. Li et al. [9] have mentioned five ways in which a federated dataset can be considered non-IID:

1. Label distribution skewness over partitions, e.g. $Pr(y_i) \neq Pr(y_j)$ for partitions i, j. ,
2. Feature distribution skewness over partitions, e.g. $Pr(x_i) \neq Pr(x_j)$ for partitions i, j.
3. Feature difference within label, e.g. $Pr(y_i|x_i) \neq Pr(y_j|x_j)$ for partitions i, j.
4. Label difference within feature set, e.g. $Pr(x_i|y_i) \neq Pr(x_j|y_j)$ for partitions i, j.
5. Quantity skewness over partitions, e.g. $|x_i| \neq |x_j|$ for partitions i, j.

3.2 Selecting non-IID properties

In Section 3.1.1, we already defined five ways a federated dataset can be considered non-IID. In our research, we focus on only two properties, namely the label distribution skewness over partitions, from now on defined as non-IIDness in the label space, and the feature distribution skewness over partitions, from now on defined as non-IIDness in the feature space.

We focus on these two properties because, first of all, we consider a horizontal FL setup. Because of this setup, we assume that all partitions have the same features, as is a property of horizontal FL: each partition has new samples rather than new information about already existing samples. This means that $Pr(y_i|x_i) = Pr(y_j|x_j)$, so we can disregard the feature difference within a label. Furthermore, we assume the clients have a common ground truth, $Pr(x_i|y_i) = Pr(x_j|y_j)$, as we assume all clients are honest protocol participants. Therefore, we can also disregard the label difference within a feature set.

Having made these two assumptions, we are left to mention that we ignore the quantity skewness, $|p_1| \neq |p_2|$ for partitions p_1, p_2. We chose this because we believe this skewness is not a property of an IID dataset, as it does not influence the underlying distribution within a dataset. Furthermore, in FL, a quantity skewness is often resolved using weighted federated averaging.

3.3 Measurement definitions

To measure the non-IIDness between two data partitions, we first define the measurements for non-IIDness in the label space, followed by the non-IIDness in the feature space. These measurements are the basis for our research and are needed to generate synthetic data. The definition of the different variables used can be found in Table 3.1.

3.3.1 Measuring non-IIDness

To understand the setup of our metric, statistical distance is necessary.

Statistical distance. To compare two partitions, we need statistical distance. Statistical distance quantifies the distance between two different probability distributions. As we see our datapoints over the various clients as probability distributions, we can quantify the divergence between the target and the attacker distributions. We use this quantification as a metric to show how similar the two distributions of the attacker and target are. Calculating a statistical distance requires our knowledge of the two distributions of the

data samples of the clients, but for real-world datasets, we do not know the underlying distribution. However, we can work around this by *estimating* the two distributions based on the datapoints we have of the two clients. These estimated distributions are what we call proxy distributions. For our synthetic generated datasets, we do not need a proxy distribution. As we are able to pick our distributions, we know the exact distribution of the datapoints. As we estimate real-world distributions into a Gaussian distribution, we will also use Gaussian distributions for the synthetic datasets. In our research, we use two different types of statistical distance: the Hellinger Distance and the 2-Wasserstein distance.

Hellinger Distance. The Hellinger distance (HD) is used as a metric for the non-IIDness in the label space, just as used by Gutierrez [3]. It is used as a metric for measuring the separation between two probability distributions and can be calculated for *discrete* probability distributions P and Q as

$$HD(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{n}(\sqrt{p_i} - \sqrt{q_i})^2}$$

If the Hellinger Distance equals 0, the distributions P and Q are similar, whilst a value of 1 indicates completely different distributions. As it is symmetric and satisfies the triangular inequality, we can denote the Hellinger Distance as a proper metric for distance.

2-Wasserstein distance. To quantify the statistical distance in feature space, we will use the 2-Wasserstein distance. The 2-Wasserstein distance uses an optimal transportation of mass based on the idea of optimally transporting piles of earth to pits to fill them. Here, the piles of earth are seen as one distribution, and the pits to fill are seen as the distribution from which we want to calculate the distance. The weighted distance travelled to fit the source distribution into the target, or the movement of dirt in our example, is then defined as the 2-Wasserstein distance. For two Gaussian distributions, the 2-Wasserstein distance is defined as:

$$W_2(\mu_1,\mu_2)^2 = ||m_1 - m_2||_2^2 + \text{trace}(C_1 + C_2 - 2(C_2^{\frac{1}{2}}C_1C_2^{\frac{1}{2}})^{\frac{1}{2}}),$$

where $\mu_1 = \mathcal{N}(m_1, C_1)$, $\mu_2 = \mathcal{N}(m_2, C_2)$ (the normal distribution with mean m_x and covariance matrix C_x) and $C^{\frac{1}{2}}$ denotes the principal square root of covariance matrix C. The 2-Wasserstein distance also satisfies the triangular inequality and thus can also be used as a proper metric.

We will use the 2-Wasserstein distance for our measurements in the feature space, elaborated in Section 3.3.3.

3.3.2 Measurement in label space

To measure the non-IIDness in the label space, we are interested in calculating the distances between the label distributions over the partitions. The goal is to have a non-IIDness in the label space of 0 when the distribution of the labels amongst the partitions is equal. On the other hand, we want this metric to be 1 when the distributions are furthest from each other, e.g. partition 1 contains all datapoints of dogs and partition 2 contains all datapoints of cats. An example of the impact of the label distribution on the non-IIDness in label space can be seen in Figure 3.1.

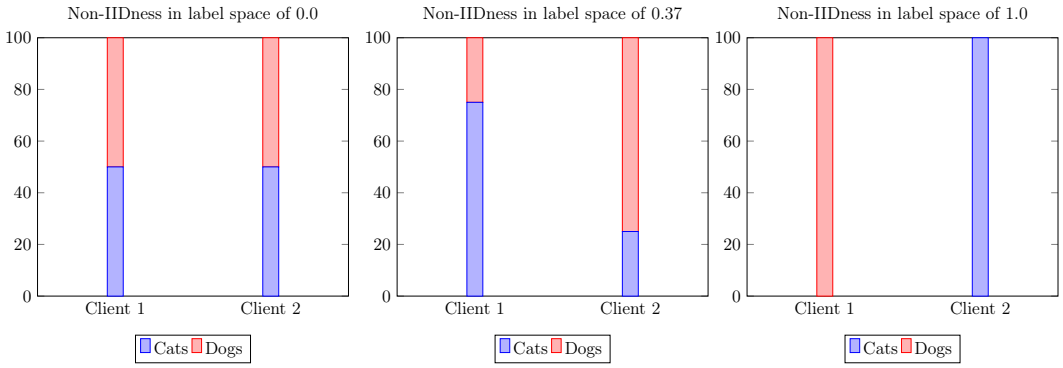


FIGURE 3.1: Impact of different distributions of the labels each client holds on the non-IIDness in label space metric.

To create this metric, we use the statistical distance explained in Section 3.3.1 already. Specifically, we create a counter for all the datapoints' labels for each partition. From these counts, we normalise to a percentage-wise distribution, which we call our label distribution of a partition. We will use the Hellinger Distance (HD) to measure the distance between two distributions, as it provides us with a metric to calculate distances of datapoints for categorical values. Furthermore, this metric has been discussed already by Gutierrez et al.[3] for the use of non-IIDness in the label space[1].

Then, with the mentioned distributions, the non-IIDness in the label space can be calculated as:

$$non - IIDness_{Label}(p_1, p_2) = HD(p_1, p_2)$$

To make this more concrete, we use the following, simple, example. Client 1 has a dataset containing 218 dogs and 654 cats, so a total of 872 datapoints. Now, Client 2 has a dataset containing 654 dogs and 218 cats, also making a total of 872 datapoints. When normalising this to percentages, Client 1 has 25% dogs and 75% cats, whilst Client 2 has 75% dogs and 25% cats. The distribution p_1 will then be [0.25, 0.75] and the distribution p_2 becomes [0.75, 0.25]. Giving this as input into the calculation for the Hellinger Distance results into approximately 0.37, which is our demonstration in Figure 3.1. This example can be extended to multiple labels easily as well, but cannot be extended to multiple clients: only two distributions can be given as input.

3.3.3 Measurement in feature space

To measure the non-IIDness in the feature space, we are interested in the distances between the feature distributions of the partitions. The non-IIDness is bound between 0 and 1 in the label space, but this is not the case for the feature space. This is due to our usage of the 2-Wasserstein distance. Unlike the Hellinger Distance, the 2-Wasserstein Distance is derived from the optimal transport theory, meaning it accounts for both the shape of the distributions and the relative location of their mass.

In Figure 3.2, a simple example of non-IIDness in feature space is given. In this figure, based on only two features, two classes are shown. One class is seen in the left bottom corner and one class is found in the right top corner. Two clients, red and orange, have

[1] Originally, we have used a different, self-made, metric for non-IIDness in the label space. As the aforementioned paper has been released during our research, we have opted to convert from our metric to the in this section described metric.

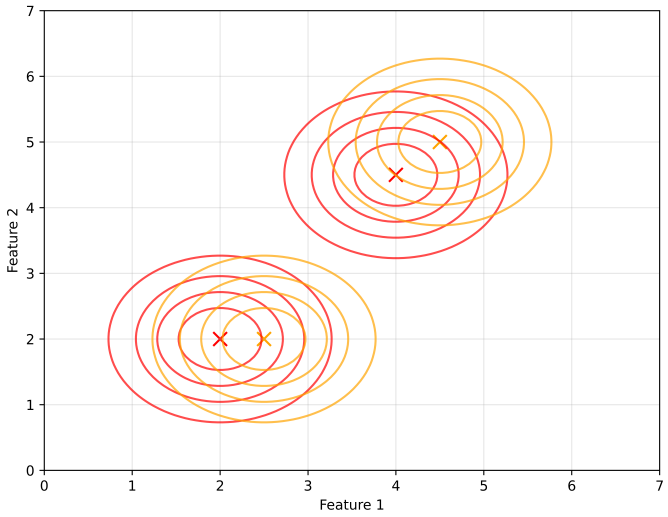


FIGURE 3.2: Example of non-IIDness in feature space

a part of these datapoints. It can be seen, however, that both clients have their center of the data distribution at a different location. The difference between the two distributions of the red and orange client for a class, is what we then call the non-IIDness in the feature space.

To measure then the non-IIDness using the 2-Wasserstein Distance, as explained in Section 3, we have to overcome one problem: the Wasserstein distance uses optimal transport. The complexity of calculating this optimal transport increases as the dimensions increase. Furthermore, we do not necessarily know the exact distribution of a feature within a partition, as we only see the datapoints drawn from this distribution. Therefore, the non-IIDness in feature space can only use *estimates* of the distributions unless we synthetically generate data according to the Gaussian distributions we choose.

To still calculate the non-IIDness in the feature space, we use a proxy distribution instead of the original one. This proxy is a translation from the datapoints to a more Gaussian-like distribution, which is translated using the Scikit-learn PowerTransformer. By doing this, we overcome the complexity of calculating the optimal transport over a non-Gaussian distribution. This allows us to calculate the Wasserstein distance between the two proxy distributions using the formula in Section 3.3.1. We want to have the distance between the exact distributions to calculate the non-IIDness in feature space, but the actual distribution is unknown for real-world datasets. Therefore, as we are only interested in the relationship between the success rate of the Membership Inference Attack and the non-IIDness in the feature space, an estimate of this distance will do. Furthermore, we argue that a relationship between the proxy distributions and the attack success rate can be extended to a relationship between the original distributions and the attack success rate of an MIA.

Defining a closed metric for the non-IIDness in feature space has proven difficult, as many factors are at play when defining the non-IIDness in the feature space. As we only use estimations based on our datapoints, the number of datapoints plays a significant role in calculating the proxy distribution. Furthermore, the number of features impacts the calculation: what is a good strategy to aggregate the distributions whilst we only calculate

the Wasserstein distance between two distributions? A simple averaging over the distances might lose outliers, whilst picking the maximum distance between two distributions may make the non-IIDness as large as one outlier.

The lack of a closed metric can become challenging, especially when generating synthetic data. When generating a classification problem with a feature non-IIDness of 20, one may multiply all data points by a factor to increase the feature non-IIDness while still having the same distribution for the target model to train on.

Because of these limitations, we use non-IIDness only in feature space in specific circumstances. We only use this non-IIDness within the same 'central' dataset, meaning that we only use it within the same dataset. We look into the impact of partitioning a 'central' dataset differently but refrain from using this metric to compare different datasets where the same dataset cannot be obtained when adding all partitions together.

Chapter 4

Experiment Setup

We implemented our attacks using the Pytorch[1] framework for the implementation of the machine learning models, together with the Flower[2] framework for the federated training of the model by the different clients. The training was done on a compute node with four Intel Xeon E7-8890 v4 processors and 2 TB of memory. As this was done in a Slurm[3] cluster, not all processors and memory were used at all times.

4.1 Datasets

In addition to generated synthetic datasets, which will be discussed in Section 5, we also used existing datasets for verification. First, we use the Texas100 dataset to verify the implementation of our attack model so we can compare our results with the Nasr et al. [14] attack, which we implemented.

Next to this dataset, we also use two publicly available federated datasets. Both datasets are collected in real-world situations and already have a natural split in the data. Therefore, we can test our attack on real-world federated data, which has mostly been done up to now with artificially split 'regular' datasets, such as the Texas100 dataset.

Texas100. The Texas100 dataset [18] consists of over 67,000 data samples collected from the Texas Health Department. It consists of discharge data from the hospitals and has 100 different labels. This dataset is not for Federated Learning and should thus be partitioned to be used for Federated Learning. Then, we will compare our results with those of Nasr et al. as a benchmark.

Students. The student performance dataset [2] consists of around 650 student achievements in secondary education of two distinct Portuguese schools. Data attributes include student grades and demographic, social, and school-related features. The task for this dataset is to predict whether a student passed or failed the course based on these features. As there is a high correlation between the intermediate grades and the final grade, as noted by [6], we opt not to include this feature. The categorical features are one-hot encoded before training.

Heart Disease. The heart disease [1] dataset consists of a total of over 900 data samples regarding heart diseases in four hospitals across the world. In most regular ML tasks, only

[1] https://pytorch.org/
[2] https://flower.ai
[3] https://slurm.schedmd.com/

the Cleveland dataset is used, whereas we will use all four hospitals' datasets. The goal of the model is to predict whether heart disease is present or not based on the fourteen different features related to health. The categorical features are one-hot encoded before training. As we keep the number of partitions the same in all our experiments, namely two, we split this dataset into two datasets, each having two hospitals. The splits we use are Long Beach (VA)-Switzerland (CH) and Cleveland (CL)-Hungary (HU), chosen to have more diverse results in the non-IIDness in the label space.

4.2 Target model

We use a target model for all aforementioned datasets consisting of five fully connected layers activated by a ReLU function. The training is, as mentioned, done using the Flower library and aggregated using Federated Averaging, FedAvg. All clients have the same number of datapoints for the synthetically generated datasets contributing to the target model. For the real-world datasets, this is not necessarily the case. Therefore, we use weighted aggregation of the models, where the number of datapoints of the client determines the weight.

4.3 Attack model

To attack the target model, we define our attack model to be the same as Nasr et al. [14]. Where Nasr et al. used multiple rounds of training for the target model as input for their attack model, we opted to use only the last round of training. We chose this solution to save computation time in our research. The attack still works, as it can be seen in Nasr's results that the last rounds contain the most helpful information for the MIA. However, this choice impacts our performance on our attack slightly, but this is not a problem for our research, as we are mainly interested in the difference in our attack performances based on the non-IIDness between client datasets. In Section 6, we show a benchmark of our implementation against Nasr, in which we show the impact on the performance of our choices, together with an overview of the differences.

In our setup, we only work with two different clients. We define one as the attacker and the other client as the victim. The attacker sets up the attack model and uses his training dataset and validation dataset to train it, where the ratio of members/non-members of the training dataset is set to 50/50. Then, after each epoch of training by the attacker, the resulting model is tested using data from the victim with the same member/non-member ratio. In this way, we simulate how an attacker could perform on the victim's data, even if he has not seen any of it yet. We train the attack model for 30 epochs, as Nasr has shown that after 30 epochs, the results stabilise.

At first sight, simplifying the number of users and the number of training rounds of the target model used in the attack model might make it look like the MIA has been reduced from federated learning to regular machine learning research. In our setup, the attacker simply only cares about the final model, where the model updates are one of the advantages of the MIA in federated learning. However, we argue that our research is still unique to the federated learning setup, as we specify one client as the attacker and one as the victim. The attacker already has more knowledge about the dataset than any third party in this scheme would have: it knows its share of datapoints in the training dataset of the final model, and it knows its share of datapoints from its validation dataset to not

be in the training dataset of the final target model. Therefore, it has a better information position for the MIA than any party in a 'regular' machine learning MIA would have.

4.4 Evaluation Metrics

To evaluate our attack, we use the same metrics as those of Nasr: attack accuracy and true/false positive rate. Furthermore, we use the advantage metric, as used by Humphries.

Attack Accuracy. The attack accuracy is defined as the portion of correctly predicted members and non-members in our test dataset divided by the total number of datapoints in this dataset. We use an equal distribution of members and non-members in our test dataset, just like in our training dataset.

True/False positive rate. The true positive rate (TPR) and false positive rate (FPR) are used as an extra metric to compare the impact of different values and forms of non-IIDness on the MIA.

Membership advantage. As used by Humphries et al. [6] and defined by Yeom et al. [20], we will be using the membership advantage, Adv, as an extra metric. This metric is defined by $\texttt{Adv} = \text{TPR} - \text{FPR}$, which results in 0 if the attack randomly decides the membership of datapoint z and in 1 when the attack always guesses the membership of z correctly.

Chapter 5

Data generation

As elaborated upon in Section 4, we will use synthetically generated datasets next to already existing ones. Where other related works, for example, Lin et al. [10], have used a Dirichlet distribution to partition existing datasets into non-IID label distributed datasets, we also focus on non-IIDness in feature space. Next, a Dirichlet distribution does not always immediately converge to the desired non-IIDness and might introduce quantity skewness, where this is not desired. Next, we want to zoom in on the impact of solely one property, where the other property stays stable, which requires a lot of control of the data generation. For these reasons, we have opted to introduce our own data generation process with the help of the scikit-learn library.

5.1 Generating non-IIDness in the label space

To keep the non-IIDness in the feature space stable whilst gradually modifying the non-IIDness in the label space, we generate our synthetic data from scratch. In our implementation, we first focus on the label counts per partition. Our method can generate these counts for any arbitrary number of partitions, but we will focus on the setup with only two partitions as we focus on this setup in our metrics.

First, based on the number of datapoints per client and the number of classes, we set the counts per label equal for all partitions and classes. This starting setup equals a non-IIDness in the label space of 0.0, meaning it is an IID distribution. Next, to increase the non-IIDness, we switch datapoints between clients, where we prefer to select the labels that have a count closest to the equal, starting, split. From this point on, we keep repeating the switching of datapoints until we reach a calculated non-IIDness in the label space sufficiently close to the value we want. In our setup, we calculate with an offset of 0.025 from the target non-IIDness. The algorithm is added as Algorithm 1.

When the target non-IIDness is acquired, the next step is to create the dataset. We do this using the `make_classification`[1] function of scikit-learn. This function generates a random n-class classification problem by creating clusters of points normally distributed about vertices of an n-dimensional hypercube, where n is the number of informative features. Then, it assigns an equal number of clusters to each class. We use specific parameters such that the standard deviation around the centers of feature clusters stays low. This results in a low non-IIDness in feature space, as the distance between the datapoints stays low. Then, using our collected label counts per partition, we sample the desired data points per partition without replacement from the generated synthetic dataset. The result

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

Algorithm 1 Generate non-IID counts in label space

Initialize `counts_per_partition` and set random seed.
for each client **do**
    Divide datapoints equally across the classes and store them.
end for
Set `previous` labels to avoid swapping the same labels repeatedly.
while `label_skewness` not in `target range` **do**
    Calculate current `label_skewness`.
    Select two random clients.
    if `label_skewness` $<$ `target range` **then**
        Find labels closest to the average distribution for both clients.
    else if `label_skewness` $>$ `target range` **then**
        Find labels furthest from the average distribution for both clients.
    end if
    Swap a small number of datapoints between the two clients for these labels.
    Update `previous` labels to track the swaps.
end while
Return `counts_per_partition` and `label_skewness`.

is a partitioning with the required non-IIDness in the label space whilst having a low non-IIDness in the feature space. The implementation of our algorithm can be found in the appendix in Section 10.

5.2 Generating non-IIDness in the feature space

The next step is generating a non-IID dataset in the feature space. Here, we again base our method on the `make_classification` function of scikit-learn. Generating this dataset relies on the measurements of non-IIDness in the feature space, as explained in Section 3.3.3. We discussed that our metric does not satisfy all situations and can only be used in a limited number of setups. We, therefore, keep the number of data points, partitions, and features the same in all setups.

Then, to generate the synthetic data, we make use of the number of clusters per class input of the `make_classification` function in scikit-learn: we let the algorithm generate a number of clusters per class equal to the number of partitions we want. Then, we assign one of the generated clusters to a partition, such that all partitions have, for all classes, a different center of the cluster.

In our method, we introduce a new variable as a function parameter: the intra-class separator. This separator defines the distance between the centers of the clusters of the same class, thus introducing non-IIDness for the features. By varying this intra-class separator, we can control the non-IIDness in the feature space for all our experiments.

This method keeps as many parameters as possible unchanged over the different experiments. The unchanged parameters are the number of clients, number of datapoints and number of features. Also, we stick to a specific random seed to be able to repeat experiments. Furthermore, we set the already existing *inter*-class separator defined in `make_classification` to a sufficiently large number so that we do not have to change this in later experiments. Moreover, we aim to keep the bounds the same for all features over the experiments, meaning that the minimum and maximum values of a feature do not change (too much) over different experiments.

By having these safeguards and enough distance between the levels of non-IIDness between experiments, we can draw an overall conclusion on the effect of the non-IIDness in feature space on the success rate of an MIA.

Chapter 6

Results

As aforementioned in Section 3.3.2, we started with our own metric for calculating the non-IIDness in the label space. After completing the experiments, Gutierrez et al. [3] published their work with a better proposal for calculating the non-IIDness in the label space, so we opted to transform our results to their metric. Therefore, in this section, not all datapoints are distributed equally over the spectrum of non-IIDness.

6.1 Benchmark

Starting our work, we first reproduce the results of the membership inference attack by Nasr et al. [14], using the Texas100 dataset. As we build upon their setup of the MIA, we want to verify and benchmark our implementation to confirm that we have implemented the attack correctly. As mentioned earlier, we have simplified several parameters that were used by Nasr, resulting in a lower accuracy than the original implementation. The key differences between our implementation and the implementation by Nasr are, first of all, the number of clients. Where we limit ourselves in this research to a two-client setup, Nasr used a four-client setup. Furthermore, we only used the last round of training, whilst Nasr also used four intermediary rounds as input for the attack. Last, where Nasr used 4,000 member datapoints and 4,000 non-member datapoints for training, we only used a fraction of the datapoints in each category: 132 datapoints. When evaluating our setup, we still achieve an attack accuracy of 58.34% against 62.4% of Nasr by a passive local attacker. Therefore, we conclude that our implementation, even though simplified, is implemented correctly so that we can conclude the relation between data imbalances and the attack success rate of an MIA. Table 6.1 shows an overview of the differences in the setup.

Implementation	# participants	# member/non-member train	# member/non-member test	# Observed epochs	Accuracy	Epochs
Our research	2	133/133	133/133	1	58.3%	30
Nasr et al.	4	4,000/4,000	4,000/4,000	5	62.4%	100

TABLE 6.1: Overview of differences between Nasr's and our setup.

6.2 Non-IIDness in label space

Having established that our implementation is correct, we continue with our first experiments, looking into the impact of non-IID properties on the MIA in federated learning. In this case, we start with the non-IIDness in the label space.

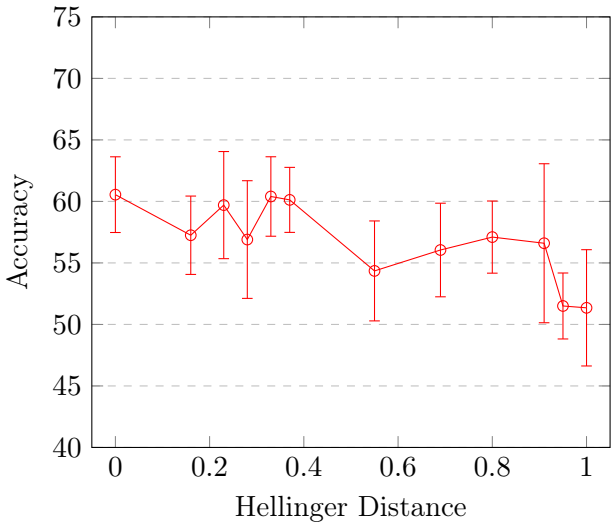


FIGURE 6.1: Attack accuracy for degrees of non-IIDness in label space

In Figure 6.1, we show the average accuracy within one standard deviation for given degrees of non-IIDness in the label space. The average accuracy is calculated over the accuracy of epoch 30 of the experiment, where we repeat all experiments ten times.

In this figure, we first note deviations in the line. Where we start with an accuracy of around 60%, we later see a lot of variation until 0.4. The reasoning, we argue, is that for every experiment, we generate a truly new synthetic dataset with a given skewness. Doing this increases the randomness factor, resulting in a more robust conclusion. We see, however, that for the first datapoints to have a more stable trend, we ideally would run the experiment more often. On the other hand, by adding the standard deviations over the runs, we can still draw conclusions from these first datapoints.

Another outlier in the figure is the sudden accuracy drop between a Hellinger Distance of 0.90 and 1.0. In this figure, we have added an extra datapoint at 0.95 to look into this phenomenon further. We see here that, with only a low standard deviation, the drop in accuracy already occurs between a Hellinger Distance of 0.90 and 0.95. We will focus more on this sudden drop in accuracy in Section 6.3.

Next to accuracy, we specified the Advantage (Adv) metric, followed by the True Positive Rate (TPR) and False Positive Rate (FPR). These different metrics have been plotted in Figure 6.2. Here, we see the same trend for the advantage and accuracy.

6.3 Cut-off point

As mentioned in Section 6.2, we are interested in the significant drop in accuracy when reaching a Hellinger Distance between 0.90 and 0.95. As we are near the maximum non-IIDness in label space, the attacker only has a few datapoints with the same label as the target. As datapoints of every label have a different distribution, the attacker has only a few datapoints with which it should estimate the original distribution of the datapoints with the same label as the client. Without enough datapoints in this distribution, the attacker cannot train its model accurately on datapoints in the training dataset, of which it does not know an estimated distribution. As a result, the attacker assumes that datapoints distributed according to the label it has almost no information on are not part

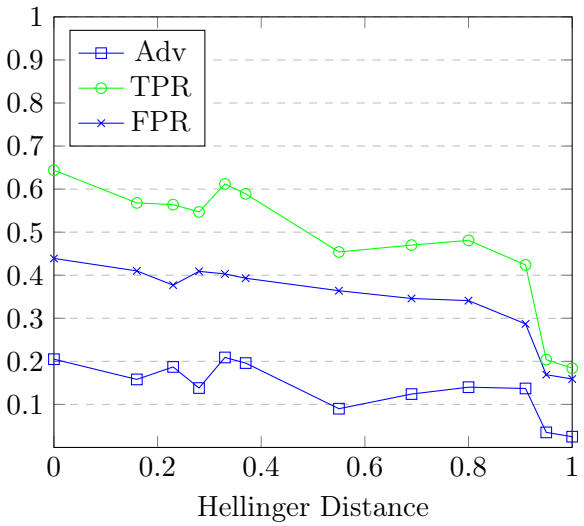


FIGURE 6.2: Adv, TPR and FPR for degrees of non-IIDness in the label space.

of the training dataset. We can see this phenomenon in Figure 6.2, where we can zoom in on the Advantage, True Positive Rate and False Positive Rate. We see the decrease mainly in TPR rather than the FPR. This means that between a Hellinger Distance of 0.90 and 0.95, the attacker predicts datapoints that are members of the training dataset to be non-members.

6.4 Non-IIDness in feature space

Next to the label space, we also look into the effects in the feature space. We have used a broad spectrum of non-IIDness in the feature space, in which we keep the non-IIDness in the label space to a minimum. In Figure 6.3, we show our results in terms of the accuracy of the MIA. All datapoints are averaged over five repeated experiments.

In this figure, we see that, in general, the non-IIDness in the feature space does not fluctuate much, with the exception of an outlier at a non-IIDness in the feature space of 1,000. From this, we conclude that there is no significant impact from solely the non-IIDness in feature space, as measured by our metric, on the success rate of a membership inference attack in our synthetic datasets. Our next step is to examine the impact of the imbalance in feature space in real-world datasets.

6.5 Impact in real-world datasets

After our results on our synthetic datasets, we now focus on some datasets collected in real-world federated settings. Table 6.2 showcases our results for different methods of partitioning the real-world datasets. The original partitioning uses the original split in which we collected a dataset, and the uniform partitioning removes the original split and uniformly assigns datapoints randomly to the clients. The equivalent label partitioning also randomly assigns datapoints to the clients but does this according to the same counts per label for all clients, thus keeping the non-IIDness in the label space equal.

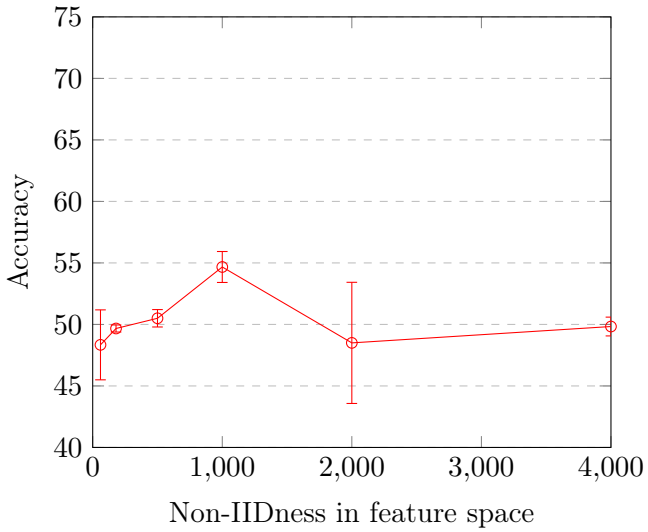


FIGURE 6.3: Attack accuracy for degrees of non-IIDness in feature space

First, also shown in Figure 6.4, we see that the MIA for the Students dataset and for the Heart VA-CH dataset the uniform partitioning performs better in accuracy than the original split. Furthermore, for Heart CL-HU, we see the accuracy stays the same. When linking these results with the non-IIDness of said datasets in the label space, we see that the 0.207 difference in Hellinger Distance. This results in an accuracy difference of 6.81 percent points for the Students dataset. Furthermore, we see a difference in Hellinger Distance of 0.186 for the Heart VA-CH dataset, resulting in an accuracy difference of 3.81 percent points.

To ensure this impact in accuracy is correlated to the non-IIDness in the label space, we also created an equivalent label partitioning to the original split, which, in each case, performs worse than the uniform split. In all cases, this equivalent label partitioning has a significantly lower non-IIDness in feature space than the original split, which for the Students and Heart CL-HU results in approximately the same accuracy as the original partitioning. However, the equivalent label partitioning results are significantly lower than the original split for the Heart VA-CH dataset. The fact that the Heart VA partition has more than 75% missing values for multiple columns, whilst the Heart CH partition has this only for one column, might impact this.

Dataset	Partitioning	Non-IIDness Label	Non-IIDness Feature	Accuracy	Acc. STD	Advantage
Students	Original	0.213	2.19e4	50.30	± 1.031	0.011
	Uniform	0.006	1.97e2	57.11	± 3.467	0.130
	Equivalent Label	0.213	4.64e3	50.89	± 1.458	0.042
Heart VA-CH	Original	0.191	1.84e41	47.75	± 1.500	-0.050
	Uniform	0.005	1.14e10	51.56	± 2.135	0.016
	Equivalent Label	0.191	1.59e13	40.60	± 1.817	-0.192
Heart CL-HU	Original	0.095	2.55e40	49.66	± 0.676	-0.007
	Uniform	0.016	1.29e3	49.66	± 0.676	0.000
	Equivalent Label	0.071	2.35e3	49.49	± 1.014	-0.010

TABLE 6.2: Results on real-world data

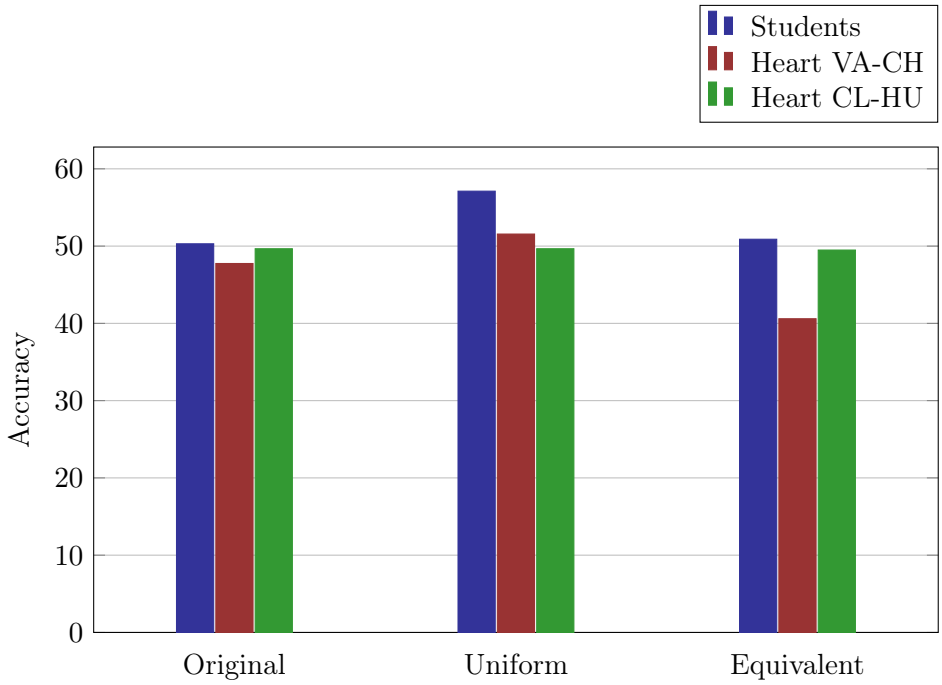


FIGURE 6.4: Accuracy of the attacks on the different partitioning methods

6.6 Impact of the attack setup

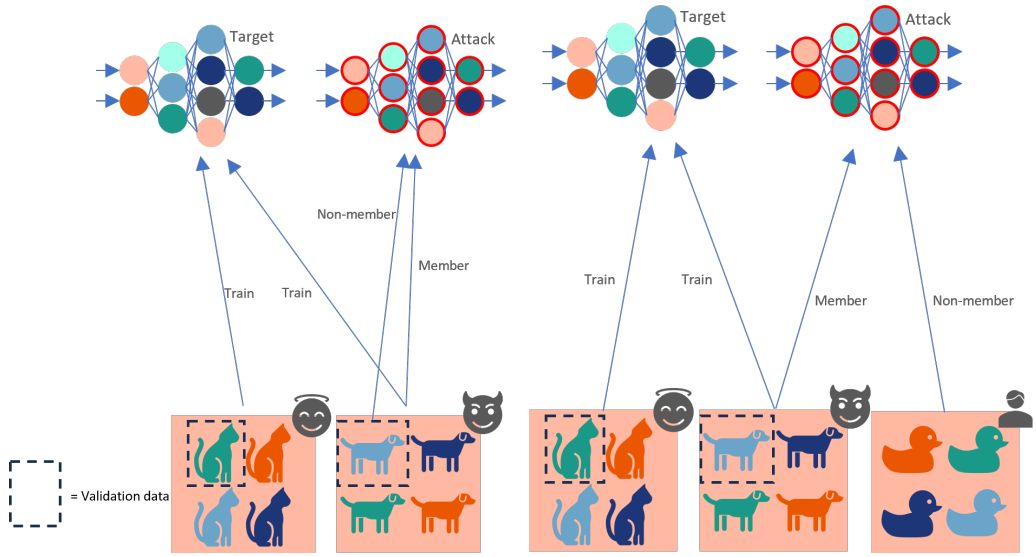
Our results conclude that IID data is more straightforward to attack than non-IID data. When looking at the research done by Humphries et al. [6], this is a surprise. First of all, they have provided significantly higher accuracies for their setup, and second of all, they claim to have the opposite effect of our research. In their case, they claim that non-IID data achieves a higher attack accuracy than IID data. Because of this interesting difference, it is worth investigating this further.

6.6.1 Differences in attack setup

Where we use the same dataset, the Heart dataset, Humphries differs in attack setup. In our setup, however, we select two parties from the four-party dataset and train a target model based on these two parties' data. Then, we use the validation data of both parties as the non-member datapoints of the target model. Humphries, however, trains the target model using data from only one of the parties and uses the remaining parties as datasets that were non-members of the target model. Our setup is illustrated in Figure 6.5a, with Humphries' attack setup in Figure 6.5b.

In these figures, we simplify the datasets into datasets with three different distributions: a dog distribution, a cat distribution and a duck distribution. We let the victim client have the cat data, the attacker client the dog data and a third party have the duck data. In our setup, we train the target model using all training data of both the victim and client. Then, from the attacker client, we use the validation data as non-member training data for the attack model. We see that the non-member data we train the attack model on, is from the same distribution as the training data of the target model.

Then, when looking into the attack setup of Humphries, we see they train the target model again using the training data of both the victim client and attacker client. But then, when training the attack model, they use the training data of the attacker as member,



(A) Attack setup we use, where we use the validation data, which is not used during training, as non-members.

(B) Attack setup Humphries uses, where they use data from a third party as non-members, possibly from a different distribution.

FIGURE 6.5: Two different attack setups used by us and Humphries

and the dataset of a third party, having a different data distribution, as the non-member training data of the attack model. The attack model is thus not trained with non-member data from the same distribution as the data of the target model.

To confirm the results of Humphries et al. and to further explore the impact of using non-member data from a third party, we perform some further experiments.

6.6.2 Results

First of all, we look into the differences in non-IIDness between the datasets we use. In Table 6.2, it is clear already that all hospitals have a different data distribution. Furthermore, we show in Table 6.3 the different non-IIDness levels for different partitioning methods of the Heart dataset, which we will use here.

Dataset partitions	Relevant for	Non-IIDness Label	Non-IIDness Feature
Heart VA v Heart CH	0% from 3rd party	0.191	1.84e41
Heart VA+CH v Heart CL	25-100% from 3rd party	0.271	4.55e8
Heart VA+CH+CL Uniform	Uniform sampling	0.069	4.07e2

TABLE 6.3: Non-IIDness levels between partitions

In our setup, we extend Humphries' setup to federated learning. We have two clients, both having part of the dataset. One client has the VA Heart dataset, and the other has the CH Heart dataset. Then, we pick the client with the VA dataset as the attacker. We select the training data of the attacker and client VA to be the training data for members. Furthermore, we select the training data of the target, client CH, to be the testing data for members. Then, depending on the setup (e.g. 0% or 100% from 3rd party), we select either all non-members to be in the validation dataset of the attacker or all non-members to be

in the dataset of a 3rd party hospital. Then, we split the non-members into a non-member training and a non-member testing dataset, where we sample without replacement.

Table 6.4 shows the results when using the different attack types. Here, we first show the 0% from the 3rd party. This is the attack model that we have used for our earlier experiments, which also shows the most real-world results. After all, we use the actual data of the victim as the test dataset, meaning we measure our attack performance on our victim. As the validation dataset was not part of the target model and was chosen randomly from the entire dataset, the validation dataset has identical data distributions as the training dataset of the target client. Therefore, we argue this is the strongest attack model for membership inference attacks within federated learning.

Dataset	Partitioning	Accuracy	Acc. STD	Advantage
Heart VA-CH	0% from 3rd party	47.75	± 7.443	-0.050
Heart VA-CH	25% from 3rd party	60.71	± 7.150	0.202
Heart VA-CH	50% from 3rd party	57.78	± 1.923	0.123
Heart VA-CH	75% from 3rd party	70.24	± 2.060	0.393
Heart VA-CH	100% from 3rd party	91.23	± 6.080	0.825
Heart VA-CH-CL	Uniform sampling	52.14	± 1.610	0.041

TABLE 6.4: Results on attack setup

We also show our results when we retrieve different percentages of the non-members from a different hospital. We see an increasing advantage in all experiments, except the 50% setup. This means we can better distinguish the difference between member datapoints and non-member datapoints. Having non-member datapoints that do not conform to the same distribution as the member datapoints increases our advantage and accuracy immensely. With this increase in accuracy, however, comes the fact that this attack should be considered a less strong attack: we make it easier to perform it. Therefore, we show that not all attack setups performing an MIA are realistic, as the setup of Humphries et al. does not realistically show the accuracy of an MIA where the attacker has knowledge of the same underlying data distribution.

We argue that this conclusion we draw in the attack setup of Humphries is no surprise: it is easier to distinguish the response of a model between data distributions it has seen and it has not seen. In this setup, the non-IIDness between the partitions is measured between members and non-members, whilst in our setup, we measure the non-IIDness between the attacker and the target dataset. This difference significantly influences the conclusions that can be drawn from the gathered results. Therefore, we show that the impact of the attack setup should not be underestimated and should be carefully considered when comparing different research results.

Having shown this huge difference in attack accuracy based on the attack setup, it is problematic to simply look at the attack accuracy of a membership inference attack in federated learning. Depending on the setup used, an attack can be extremely good or purely useless. Therefore, the research community must investigate this unexpected result to agree on a realistic setup for attacks on federated learning schemes. Without such a realistic setup, we cannot tell whether machine learning deployments of federated learning instances are at risk.

Chapter 7

Related Works

Our research differs from several related works. This section will highlight the differences and similarities, grouped by topic.

7.1 Inference attacks

Different attacks exist already that infer information from a trained machine learning model. An example of such an attack is a property inference attack, which tries to infer global properties of all datapoints within the trained model. Another type of attack is the attribute inference attack, which tries to infer a specific feature of a datapoint. An example may be to infer the race, based on the age, income and occupation of a person. Such an attribute inference attack reduces to a membership inference attack [17], meaning that security against an attribute inference attack implies security against a membership inference attack. One last example, is data reconstruction. This type of attack is a very strong attack, as a successful attack would allow an attacker to reconstruct the training data. Fortunately, this attack as well reduces to a membership inference attack. All reductions and relations between the different inference attacks and the membership inference attack show the importance of focusing on the membership inference attack, as resistance against this attack provides resistance against multiple other inference attacks as well.

7.2 Membership Inference Attack

The Membership Inference Attack we have been using is the setup of Nasr et al. [14]. This setup is already an extension of the work of Shokri et al. [18], who devised a black-box setting for Membership Inference Attack. They focused on creating an attack model without having access to the original data, and it was Nasr who then implemented the first white-box attack. Implementing the white-box passive attack in a supervised setup was used as the basis for our research, which Nasr has already used to simulate Federated Learning. They have implemented the attack on a federated training of the Texas100 dataset and claim representative results for a federated learning setup. However, our research shows that the study of Nasr cannot be extended to federated learning without making assumptions about the data imbalances between clients.

7.3 Non-IIDness in Federated Learning

Where Nasr has measured the attack accuracy on multiple datasets under federated learning, they do not explicitly mention their use of IID partitioning of the dataset over the clients. Our research shows that uniform partitioning, or partitioning with a low degree of non-IIDness in the label space, yields a higher MIA performance, although this partitioning is often unrealistic. Hsieh et al. [4] have already found that skewed distribution of data labels across clients arises frequently in the real world, supported by several examples. Furthermore, they show that these skewed data labels are a fundamental problem for decentralised learning, causing significant accuracy loss.

Whilst we have not zoomed in on the relation between the training accuracy of the target model and the MIA in this research, Yeom et al. [20] have shown already that models become more vulnerable for an MIA as they overfit more.

7.4 Membership Inference Attack & Non-IIDness

The combination of non-IID data often occurring in real-world datasets and an MIA being more successful in overfitted models already shows optimism in the attack success rate. Next, Humphries et al. [6] concluded that current MIA experiments and evaluations hinge on the restrictive assumption that members and non-members are IID data samples. With their focus on the *independence* part of non-IIDness, they have shown that a distribution with dependent datapoints is even more vulnerable to an MIA than bounds set before. In our research, we add to that the impact of the *identical* distribution amongst the clients, where we find that datasets with an identical distribution are more vulnerable to an MIA than imbalanced distributions. This illustrates even more the relevance of the degree of non-IIDness to the impact of an MIA.

7.5 Calculating non-IIDness in label space

This degree of non-IIDness, of course, needs a metric. No metric exists yet for the feature space; various options are now used for the label space. Gutierrez et al. [3] have recently compared the Hellinger Distance and the Jensen-Shannon Distance as suitable options as metrics for the degree of non-IIDness in the label space. The two metrics show similar results when calculating the non-IIDness in setups with only a few clients. On the other hand, they also found that the metrics diverge a lot when using more and more clients in the federated learning setup. We only use two clients in our federated learning setup, meaning that both the Jensen-Shannon Distance and Hellinger Distance should not diverge much.

Chapter 8

Discussion & Future works

In our work, we give new insights into the impact of data imbalances on the attack success rate of a Membership Inference Attack. Furthermore, we show that different partitioning methods of datasets amongst clients can also significantly impact the success rate. We conclude that there are different setups to perform a Membership Inference Attack within Federated Learning, where we show that other methods may result in vastly different outcomes, whilst not all attack scenarios can be seen as realistic in real-world settings.

Our main contribution is showing that MIAs are not always performed in a setup that is relatable to real-world datasets, in which we show that the impact of data imbalances in label space impacts the attack success rate of an MIA. Even though Humphries et al. [6] already implemented an attack on non-IID data, we extend this implementation to Federated Learning. Moreover, we show the different datasets' impact of the non-IIDness between different clients in the federated learning protocol, for which we use a realistic attack scenario.

Still, our attack only uses two clients in this federated learning protocol: the attacker and the client. While our setup still represents federated learning, it does not show the impact of the number of clients. We argue that the most critical information is knowledge of the target's underlying data distribution, which would result in extra clients only adding extra noise in terms of non-IIDness. It would be interesting for future work to implement the extra clients and extend the metrics for non-IIDness to support calculation between multiple partitions. Furthermore, it would be interesting to look into the impact of the imbalance in the amount of data on the success rate of the MIA. Even though we do not consider it a true property of non-IID data, it still plays a role in the knowledge one has over the partitioned data.

Another limitation we find in our work is that all our synthetic generated data is sampled from a Gaussian distribution, which results in a non-IIDness that does not convert directly to real-world situations, where the data distribution is often more complicated and noisy. Furthermore, our data generation does not show the relation between non-IID data in feature space and non-IID in label space. Features have different correlations amongst each other, just as to the label. When introducing non-IIDness in either of the two dimensions, we currently do not consider this correlation. We still capture this issue by comparing results with real-world datasets and different splits but cannot compare the various degrees of non-IIDness within datasets this way.

Furthermore, we worked on a metric to measure the non-IIDness in feature space. In the real-world datasets, we estimate this non-IIDness based on our datapoints. Using this proxy distribution, we determine the degree of non-IIDness of the dataset and link the accuracy of the MIA to it. Whilst we do not see any significant effect of the degree of

non-IIDness in feature space on the accuracy of the MIA, we cannot conclude that there is no such relation in the actual distribution. It might be the case we cannot capture this relation by using our proxy distributions.

Last, as we have shown in Section 6.6, the impact of the attack setup determines the success rate of a membership inference attack in federated learning. Therefore, it is important to investigate further to find an attack setup that is realistic in real-world settings. Without one, it is impossible to tell which federated learning deployments are at risk, or not. Further research thus is needed from the community to find out under which assumptions an attacker realisticly is able to perform a good membership inference attack within federated learning.

Chapter 9

Conclusion

In this work, we show the impact of data imbalances amongst clients in federated learning on the accuracy of a membership inference attack. We do this by implementing the membership inference attack by Nasr et al. [14] and using the definitions of non-IIDness defined by Humphries et al. [6]. By generating synthetic data, we show that in federated learning, the non-IIDness in the label space significantly impacts the accuracy and advantage of a membership inference attack. Moreover, we show that for different real-world datasets, a uniform dataset split over the clients achieves a higher accuracy of the membership inference attack than for the original non-IID split. We show that related works often perform benchmarks on federated learning in balanced setups, achieving high accuracy scores, whilst real-world datasets are frequently imbalanced and do not achieve as high accuracies as balanced data. Last, we show the impact of the attack setup, where attack setups with slight differences can lead to entirely different conclusions.

Bibliography

[1] William Steinbrunn Andras Janosi. Heart Disease, 1989. URL: https://archive.ics.uci.edu/dataset/45, doi:10.24432/C52P4X.

[2] Paulo Cortez and Alice Maria Gonçalves Silva. Using data mining to predict secondary school student performance. EUROSIS-ETI, April 2008. URL: https://repositorium.sdum.uminho.pt/handle/1822/8024.

[3] Daniel Mauricio Jimenez Gutierrez, Aris Anagnostopoulos, Ioannis Chatzigiannakis, and Andrea Vitaletti. FedArtML: A Tool to Facilitate the Generation of Non-IID Datasets in a Controlled Way to Support Federated Learning Research. *IEEE Access*, 12:81004–81016, 2024. URL: https://ieeexplore.ieee.org/abstract/document/10549893, doi:10.1109/ACCESS.2024.3410026.

[4] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The Non-IID Data Quagmire of Decentralized Machine Learning, August 2020. arXiv:1910.00189 [cs, stat]. URL: http://arxiv.org/abs/1910.00189.

[5] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership Inference Attacks on Machine Learning: A Survey. *ACM Computing Surveys*, 54(11s):1–37, January 2022. URL: https://dl.acm.org/doi/10.1145/3523273, doi:10.1145/3523273.

[6] Thomas Humphries, Simon Oya, Lindsey Tulloch, Matthew Rafuse, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Investigating Membership Inference Attacks under Data Dependencies, June 2023. arXiv:2010.12112 [cs]. URL: http://arxiv.org/abs/2010.12112.

[7] Hadi Jamali-Rad, Mohammad Abdizadeh, and Anuj Singh. Federated Learning With Taskonomy for Non-IID Data. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8719–8730, November 2023. URL: https://ieeexplore.ieee.org/abstract/document/9739132, doi:10.1109/TNNLS.2022.3152581.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. URL: https://www.nature.com/articles/nature14539, doi:10.1038/nature14539.

[9] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated Learning on Non-IID Data Silos: An Experimental Study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978, Kuala Lumpur, Malaysia, May 2022. IEEE. URL: https://ieeexplore.ieee.org/document/9835537/, doi:10.1109/ICDE53745.2022.00077.

[10] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble Distillation for Robust Model Fusion in Federated Learning, March 2021. arXiv:2006.07242 [cs, stat]. URL: http://arxiv.org/abs/2006.07242.

[11] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. ML-DOCTOR: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models.

[12] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to Federated Learning: A Survey, March 2020. arXiv:2003.02133 [cs, stat]. URL: http://arxiv.org/abs/2003.02133.

[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, April 2017. ISSN: 2640-3498. URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[14] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 739–753, San Francisco, CA, USA, May 2019. IEEE. URL: https://ieeexplore.ieee.org/document/8835245/, doi:10.1109/SP.2019.00065.

[15] Jun Niu, Xiaoyan Zhu, Moxuan Zeng, Ge Zhang, Qingyang Zhao, Chunhui Huang, Yangming Zhang, Suyu An, Yangzhong Wang, Xinghui Yue, Zhipeng He, Weihao Guo, Kuo Shen, Peng Liu, Yulong Shen, Xiaohong Jiang, Jianfeng Ma, and Yuqing Zhang. SoK: Comparing Different Membership Inference Attacks with a Comprehensive Benchmark, July 2023. arXiv:2307.06123 [cs]. URL: http://arxiv.org/abs/2307.06123.

[16] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023. URL: https://udlbook.com.

[17] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella-Béguelin. SoK: Let the Privacy Games Begin! A Unified Treatment of Data Inference Privacy in Machine Learning, April 2023. arXiv:2212.10986 [cs]. URL: http://arxiv.org/abs/2212.10986.

[18] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, San Jose, CA, USA, May 2017. IEEE. URL: http://ieeexplore.ieee.org/document/7958568/, doi:10.1109/SP.2017.41.

[19] Zuobin Xiong, Zhipeng Cai, Daniel Takabi, and Wei Li. Privacy Threat and Defense for Federated Learning With Non-i.i.d. Data in AIoT. *IEEE Transactions on Industrial Informatics*, 18(2):1310–1321, February 2022. Conference Name: IEEE Transactions on Industrial Informatics. URL: https://ieeexplore.ieee.org/abstract/document/9408373, doi:10.1109/TII.2021.3073925.

[20] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting, May 2018. arXiv:1709.01604 [cs, stat]. URL: http://arxiv.org/abs/1709.01604, doi:10.48550/arXiv.1709.01604.

[21] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, March 2021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950705121000381, doi:10.1016/j.knosys.2021.106775.

[22] Chong Zhang, Ananya Choudhury, Leroy Volmer, Johan Soest, Inigo Bermejo, Andre Dekker, Aiara Lobo Gomes, and Leonard Wee. Secure and Private Healthcare Analytics: A Feasibility Study of Federated Deep Learning with Personal Health Train. preprint, In Review, July 2023. URL: https://www.researchsquare.com/article/rs-3158418/v1, doi:10.21203/rs.3.rs-3158418/v1.

[23] Gongxi Zhu, Donghao Li, Hanlin Gu, Yuxing Han, Yuan Yao, Lixin Fan, and Qiang Yang. Evaluating Membership Inference Attacks and Defenses in Federated Learning, February 2024. arXiv:2402.06289 [cs]. URL: http://arxiv.org/abs/2402.06289, doi:10.48550/arXiv.2402.06289.

[24] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-IID data: A survey. *Neurocomputing*, 465:371–390, November 2021. URL: https://www.sciencedirect.com/science/article/pii/S0925231221013254, doi:10.1016/j.neucom.2021.07.098.

[25] Xiangfeng Zhu. Learning From Non-IID data. URL: https://xzhu0027.gitbook.io/blog/ml-system/sys-ml-index/learning-from-non-iid-data.

Chapter 10

Appendices

A. Implementation of label non-IIDness generation

```
def generate_label_skewed_counts(num_clients: int, num_datapoints_per_client: int,
                                 num_classes: int, target_skewness: float):
    counts_per_partition = []

    # Generate a label distribution of 0.0 by equally dividing the datapoints
    # over the clients.
    for client in range(num_clients):
        client_ = {}
        for label in range(num_classes):
            client_[label] = num_datapoints_per_client // num_classes
        counts_per_partition.append(client_)

    # Set the previous labels, such that we do not keep swapping the same labels.
    previous = [0, 1]

    # Calculate the label skewness of the generated data and keep adjusting the data
    # until the skewness is within the target range.
    while True:
        label_skewness = calculate_label_skewness(counts_per_partition)

        # If the calculated skewness is smaller than the target skewness, we need
        # to swap labels such that both clients get more of each other's label.
        if label_skewness < (target_skewness - 0.05):
            client_1 = random.choice(range(num_clients))
            client_2 = random.choice([x for x in range(num_clients) if x != client_1])

            # Select the labels that are closest to the middle of the distribution to
            # swap, such that both clients get a more extreme (e.g. 0 or
            # num_datapoints_per_client // num_classes) label.
            label_1 = np.argmin([abs(counts_per_partition[client_1][label] -
                        (num_datapoints_per_client // num_classes)
                        ) if label not in previous else np.inf for label in
                        range(num_classes)])
            label_2 = np.argmin([abs(counts_per_partition[client_2][label] -
                        (num_datapoints_per_client // num_classes)
                        ) if label != label_1 else np.inf for label in
                        range(num_classes)])

            # Swap at most 25 labels datapoints at once, or the minimum of the two
            # labels that a client has.
            to_swap = min(counts_per_partition[client_1][label_1], 25,
                        counts_per_partition[client_2][label_2])
```

```
            counts_per_partition[client_1][label_1] -= to_swap
            counts_per_partition[client_2][label_1] += to_swap

            counts_per_partition[client_1][label_2] += to_swap
            counts_per_partition[client_2][label_2] -= to_swap

            previous = [label_1, label_2]

        # Else, if the label skewness is higher than desired, we need to swap labels
        # from the extremes of two clients to get more towards the average each
        # client should have.
        elif label_skewness > (target_skewness + 0.05):
            client_1 = random.choice(range(num_clients))
            client_2 = random.choice([x for x in range(num_clients) if x != client_1])

            # Now, select 2 labels based where the probability is equal to the label
            # that is closest to 0 or num_datapoints_per_client // num_classes:
            # so the furthest away from the middle.
            label_1 = np.argmax([abs(counts_per_partition[client_1][label] -
                        (num_datapoints_per_client // num_classes)
                        ) if label not in previous else -np.inf for label in
                        range(num_classes)])
            label_2 = np.argmax([abs(counts_per_partition[client_2][label] -
                        (num_datapoints_per_client // num_classes)
                        ) if label != label_1 else -np.inf for label in
                        range(num_classes)])

            to_swap = min(counts_per_partition[client_2][label_1], 25,
            counts_per_partition[client_1][label_2])

            counts_per_partition[client_1][label_1] += to_swap
            counts_per_partition[client_2][label_1] -= to_swap

            counts_per_partition[client_1][label_2] -= to_swap
            counts_per_partition[client_2][label_2] += to_swap

        # If the label skewness falls within the margin of the target skewness,
        #  return the counts per partition and actual skewness.
        else:
            return counts_per_partition, label_skewness
```