Differential Privacy for Human Activity Recognition

Master Thesis

presented by
Maximilian Hopp
Matriculation Number 1471618

submitted to the
Computer Vision Group
Prof. Dr. Möller
University of Siegen

March 2025

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

Abstract

Gradient Inversion (GI) is a problem that emerged after the development of the Federated Learning (FL) algorithm. In FL a neural network is jointly trained through multiple devices by sharing gradients with a central server [24]. From these gradients, the input of the original model can be reconstructed in certain cases. In this thesis, the focus in GI is shifted to Human Activity Recognition (HAR) with focus on a sensor setting, as research in this field mainly used image datasets where the original image and label were reconstructed. [14, 41] We evaluated the performance of existing label reconstruction attacks for different settings of the FL-algorithm, model, training stage, batch size, and sampling strategy. The two HAR datasets WEAR [4] and Wetlab [37] are used for sensor input. The effectiveness of the commonly used defense strategy differential privacy [27] is implemented through gradient clipping and noise addition to the gradients. The main focus is on the FedSGD algorithm, but preliminary results for the FedAVG algorithm are also provided. The findings suggest that in HAR sensing, label reconstruction is significantly affected by the number of classes, sampling method, and distribution of the classes in a dataset. In particular, there are substantial differences in leakage between the WEAR and Wetlab datasets. The latter seems also to influence the necessary magnitude of clipping and noise to make the gradients robust against attacks. Individual subjects showed a difference of up to 10% in the accuracy of label reconstruction, possibly due to the different distribution of their data.

1.1 Problem Statement

The growth in popularity of neural networks over the last decade resulted in an interest to reduce computational effort. Therefore, the Federated Learning (FL) algorithm was created, where multiple devices train a single neural network together by sharing their gradient updates with a central server [24]. Different modalities such as Vision and HAR then adapted their model training into a FL setting in hopes of achieving improved results [19]. As the HAR setting for sensor data was not considered until now, this thesis now focuses on this problem [46]. The goal in HAR sensing is to predict human activities based on data provided by sensors or smartwatches for example accelerometer data [4]. Another benefit is that single users can help train a complex model with a limited amount of data. Especially HAR can benefit from the FL paradigm because data is often recorded by sensors or smartwatches. These devices can potentially participate in the algorithm on the fly while collecting the data. For FL it was believed that the data of a user remains private because only gradient updates are shared with other users. Thus, the data would be safe, but recent research results have shown that raw data, such as images and labels, can be reconstructed from gradient updates in certain cases and a set of assumptions. [14] The leakage of the labels often is used to enable performance improvements in further attacks such as data reconstruction [14, 52], membership inference attacks [42] and attribute inference attacks [25].

1.2 Contribution

In this master thesis, the following contribution will be made: Analysis of privacy vulnerabilities for two Human Activity Recognition (HAR) state-of-the-art (SOTA) architectures under various settings for a HAR sensing scenario using FL-algorithms. The implementation is a combination of the Temporal Action Localization (TAL) repository, the breaching repository [1] and self-written code. In the evaluation GI-attacks on the labels of a NN batch are simulated in different settings of the FedSGD and FedAVG FL-algorithms. The settings differ in model architectures, datasets, model training stage, and data sampling. The evaluated datasets consist of sensor data collected by smartwatches. For training stages, the possibility of an untrained and trained model is considered. In the data sampling step, the methods $sequential$, $balanced$, $unbalanced$, and $shuffle$ are used. Lastly, options are evaluated on how to make the different architectures and their gradients less vulnerable to GI-Attacks are evaluated. The defense in this thesis is formed by the common approach of Differential Privacy (DP), that uses gradient clipping and

[1] https://github.com/JonasGeiping/breaching

noise addition to conceal the original gradients as much as possible. To summarize this, the main contributions are as follows:

1. Evaluating label reconstruction attacks on the FedSGD algorithm FedSGD and FedAVG for different models, datasets, batch sizes, sampling strategies, attacks, and privacy settings in HAR sensing.

2. Proposing a new attack Greatest Common Divisor Bias, that uses the bias gradients to estimate the impact of labels on the gradients in a different way than done in [13].

3. Providing preliminary results about the performance in the FedAVG setting and lessons learned for future research

This thesis consists of 5 chapters. In chapter 1 the problem statement, contribution and related work are presented. The necessary theoretical background of the thesis is introduced in Chapter 2. A closer look at the methodology is given in Chapter 3. An overview of the experimental setup and results of the evaluation are described in Section 4. Lastly, the results are concluded in chapter 5 and a look at possible future work is provided, mainly focused on experiments that did not fit the time frame of the thesis together with an overview of problems that remain unsolved.

1.3 Related Work

1.3.1 Federated Learning (FL)

A FL-algorithm can be used to train a machine learning model through multiple devices while allowing the training data to remain private. [24] Another advantage is that independent users can train a strong model together without the need to be connected and the requirement of vast amounts of data. [50] It can also be useful to split the computation cost onto multiple devices. Especially HAR is cut out for FL, because data is often recorded by multiple devices or sensors. The survey on FL by Li et al. [19] focuses on human sensing and shows that extensive research for HAR in FL is carried out. Topics like privacy and security, communication costs, setup, optimized FL, data usage and heterogeneity are investigated and try to improve the framework. Three popular variants of this algorithm are horizontal FL, vertical FL and federated transfer learning [50]. In horizontal FL the users share a similar feature space / classes, but posses data that are mostly differing. The vertical FL setting can be interpreted as the counterpart because in this method the users vary in their feature space but hold similar data.

For federated transfer learning the setting is changed, as here the feature space and data are mostly different between users. As a means of overcoming this challenging setting, the concept of transfer learning is integrated into the FL approach [47]. An overview of the three FL algorithms is showcased in 1.1 where it can be seen how users, data samples and feature space vary in each variant.

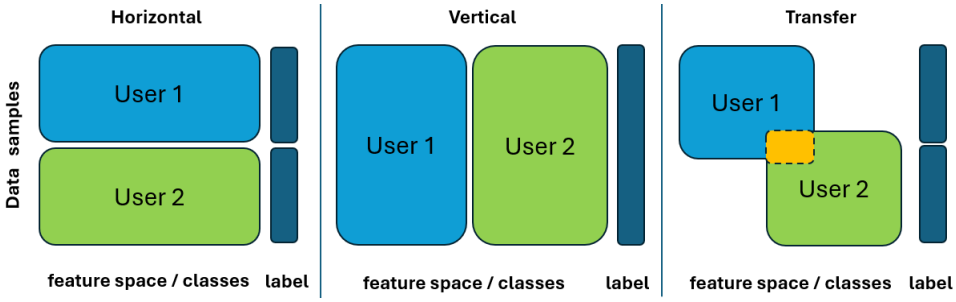


Figure 1.1: Federated learning algorithms: Horizontal (left), Vertical (middle), Transfer (right) inspired from [22]

In the GI literature the horizontal FL approach is commonly used and will also be the focus of this thesis. [14, 48, 52, 10, 41, 23, 13] The basic idea of FL is to have a central server that provides a global model to users who participate in the training process. On the user side, the actual model training is executed, and the resulting gradients are shared with the server. After receiving the gradients, the server updates the global model and reissues it to all users. This process can be further divided into Federated Stochastic Gradient Descent (FedSGD) and Federated Averaging (FedAVG). For FedSGD the users run a single local update of the ML model and send the gradients to the server. On the server side, the gradients are averaged and used to update the model [24]. In FedAVG the local SGD steps by a single user are increased and the resulting gradients are averaged on the server side. Both variants will be covered in this thesis.

1.3.2 Gradient Inversion (GI)

It was assumed that the FL-algorithms keep the training data of the participating users private because only the gradients are shared with the central server. Contrary to this assumption, a variety of research showed potential risks of privacy leakage because original data, original labels, or information such as membership of data in a training batch were successfully reconstructed from gradients in certain settings.

The discussion of data reconstruction from gradients was started in [28, 29] where the authors used a training batch of size 1 in a simple NN to demonstrate this possibility. Zhu et al. [56] introduced an attack that allowed the reconstruction of the original data from the calculated gradients of the FL process. The idea is to generate dummy data as input for a NN model and to iteratively optimize the distance between the gradients of the dummy data and the gradients that have already been leaked. The euclidean distance and L_BFGS [20] used in [56] was exchanged for cosine similarity and the Adam optimizer in [14]. There, it was also shown that the attack works on untrained and trained deep and shallow models. Another finding of Geiping et al. [14] was that the input to FL-Layer is always reconstructable in any network architecture. Multiple factors such as activation and loss functions, optimizer and batch size can influence the costs and effectiveness of a GI-attacks as shown in [44]. In [32] the convergence of the attack was improved by improving the initialization process of the dummy data. An analytic attack was introduced in [12] where the FL-Layer can be used to reconstruct the input data. The authors also showed the possibility of using this attack in Convolutional Neural Network (CNN)s. The third attack style is a recursive (closed form) attack as in [55]. Here, the data is reconstructed by solving a sequence of linear equation systems in a recursive manner. As noted in [41], these attacks mostly neglect the extraction of the original labels. The extraction of the original labels was used in the attack Deep Leakage from Gradients (DLG) by [56] but only as part of the optimization process itself. An improvement of the attack called Improved Deep Leakage from Gradients (iDLG) is presented in Zhao et al. [52] because the original work often produces wrong labels, works only with batches ≤ 8, needs a phase in the training process where the model is sensitive to weight initialization, careful hyperparameter selection, and sometimes still does not converge. The improved version in [52] uses the insight that gradient weights of the last layer in a NN classification model have a negative value if their respective label is present in the input data. However, this is valid only for a batch with a single sample. An extension to arbitrary batch sizes is proposed in a short paper by Wainakh et al. [40]. Further analysis of the weight gradients is also done by Wainakh et al. [41] where the idea in [40] is introduced as the new attack Label Leakage from Gradients (LLG) and is based on two new properties. One being the insight that if a weight gradient of the last layer in a NN model is negative, the corresponding label must be found in the input batch with arbitrary size. However, a negative activation function is required. As second new property they prove that for an untrained model, the magnitude of a gradient $g_i = \mathbf{1}^T \cdot \nabla \mathbf{W}_L^i$ is roughly proportional to the number of appearances of label i in the input batch. They used this fact to create an algorithm that uses the impact of a single label on the gradient to extract the original labels.
The LLG algorithm is also used in recently published attacks as a baseline for com-

parison [13, 51]. In [13], a shift of focus to the bias gradients of the last NN layer happens, here the bias gradients are used to execute an algorithm with a style similar to the LLG attack from [41].

An attack called Instance-wise Label Restoration from Gradients (iLRG) uses approximations of recovered class-wise embeddings and post-softmax probabilities to create linear equations of the gradients which are solved through the Moore-Penrose pseudoinverse algorithm and yield the number of label occurrences. [23]
Another new algorithm called Gradient Bridge (GDBR) is introduced by [51] where they build gradient bridges from layer-wise gradients without having to rely on the last layer of the model, like most of the previous mentioned attacks. They follow the flow of the gradients in the lower layers of the model and build equations to derive the original labels.

Based on the recently conducted survey by Li et al. [19], there are works to be found that combine HAR sensing and DP [21, 31, 45, 35, 49] but no research specifically focuses on label reconstruction for this part of HAR. The goal of this thesis is to close this gap and show the differences in label reconstruction compared to vision-based settings.

1.3.3 Protecting privacy in FL environments

A popular method to restrict the possibility of gradient leakage is to use Differential Privacy [11], which has been adjusted for use in NN models and FL in several works. [1, 15] In this defense method, the shared gradients are first clipped and then combined with noise. The downside of DP is the potentially worse accuracy of the neural network. [27]
More lightweight defense approaches that have less influence on the training process have also been proposed, such as the increase of the batch size. [56, 40, 13]
Another option would be to remove the bias in the last layer, as proposed by [36]. This would completely shut down some of the previously presented attacks such as LLBG and iLRG.
Security defenses like secure aggregation can also be integrated as defense measures but are computationally expensive and can still be broken under specific conditions, as shown in [43]. Another popular method is gradient compression, where the gradient coordinates below a predefined threshold are set to zero. This was applied in [30, 53, 41, 13], but will not be used in this thesis due to time constraints.

Chapter 2

Theoretical Background

For the theoretical background, the topics connected to GI-attacks will be introduced. This chapter covers the relevant parts for this thesis regarding the topics Human Activity Recognition, Neural Networks, Federated Learning, Gradient Inversion and Differential Privacy.

2.1 Human Activity Recognition

The research in the domain of HAR is focused on recognizing human activities from collected data. In this section HAR will be introduced by explaining the aspects of recognition models, neural networks, and challenges. In this thesis, the HAR setting is restricted to sensor data. The main goal for inertial-based HAR is to predict activities based on one or multiple sensor streams. A sliding window is applied to the sensor stream that takes a certain time frame from the stream. In this thesis, it is one second or 50 rows of the sensor stream. One sample is represented by $[1sec.xAxes]$. This sample is then put into a neural network which predicts the activity that was performed in this time frame. It is also possible to predict multiple samples T at the same time, adding another dimension to the input: $[Tx$ $1sec.x$ $Axes]$. [5]

2.1.1 Recognition Models

The NN-models in HAR sensing face a different task than the models in the vision community because they have to use time series data as input. Consequently, the SOTA models in HAR sensing are not the same as in the vision community. In [54] rules for designing HAR models have been proposed:

1. Enhancement of Extraction of local temporal context

2. Different sensor modalities should be treated unequally
3. Multi-modal fusion
4. Global temporal information extraction
5. Appropriate reduction of temporal dimension

Two SOTA-HAR models are the DeepConvLSTM [26] and the TinyHAR [54] model. They will be introduced in the next section 2.2.

2.2 Neural Network

A Neural Network (NN) is a Machine Learning (ML) algorithm that is trained on example data and aims to make predictions on new data. The architecture of a neural network is made up of so-called layers where the first layer serves as the input layer for the data. After the input layer, the NN uses hidden layers to propagate the data through the system. For the hidden layers, a variety of variations are used, some of the most prominent ones are Fully Connected, Batch Normalization, ReLU, LSTM, Dropout and Convolution. Most of the layers in a NN contain weights W that can be used to influence the calculation and output of a layer.
In the following, two popular and well-known HAR models are described, which will also be used for the evaluation.

2.2.1 DeepConvLSTM

DeepConvLSTM is an architecture used for HAR tasks and was introduced by [26], it is designed to extract discriminative features automatically from the input and then model temporal dependencies based on these features. It starts of with four consecutive convolution and recurrent layers and passes the result to a LSTM layer. For generalization purposes, a dropout layer is added after the LSTM and finally a FL-Layer is used to make the final prediction.

2.2.2 TinyHAR

The TinyHAR model can be seen as a special form of DeepConvLSTM and was proposed by [54]. It uses five different parts in its architecture: a convolutional subnet, cross-channel interaction, cross-channel fusion, temporal information extraction, and temporal information enhancement.

The convolutional subnet is the first step in the model, which is responsible for taking in the raw data and extracting local features from it. Through the convolution layers, the features are not only extracted but also fused with each other.

The convolutional layers all posses batch normalization [17] and nonlinearity with a Rectified Linear Unit (ReLU). [54].

A transformer encoder block [39] is used for learning the interaction of sensor channel dimensions at each time-step and uses a scaled dot-product self-attention layer and a NN consisting of two FL-Layer to propagate the data forward. The importance of every sensor channel is determined by the attention layer by comparing the similarity of a sensor channel with the other sensor channels. Each sensor channel passes through FL-Layer to further fuse the features of the channels. [54].

A cross-channel fusion of features can be achieved with a FL-Layer. TinyHAR also uses a LSTM layer for extracting temporal information. As this step is done after the Cross-Channel Fusion, the features across sensor and filter dimensions are already fused. [54]. The temporal information enhancement step has the goal to assess the relevance of all the features in the complete sequence to recognize an activity.

2.2.3 Challenges

HAR faces different challenges than other modalities like Image Recognition due to the different goal of what is to be recognized. In the following the challenges presented by Bulling et al. [8] are summarized.

The task starts with defining clearly what activities are to be recognized. The WEAR dataset shows this in the definition of the jogging activity, where it is divided into 4 sub classes: skipping, rotating arms, sidesteps, butt-kicks. Another challenge is the possible class imbalance because certain activities might be occurring more frequently than others. In [3] this is explained based on behavioral monitoring over an extended time frame, here activities like sleeping and working will occur more often than short irregular actions like drinking.

The third challenge is the annotation of the ground truth labels of the training data. In HAR annotation is presumably more tedious than for images because it either needs to happen while recording the data or afterwards by going over the raw sensor data. [8] For datasets like WEAR, that consists of sensor and video data, it can be less of a problem by overlapping the data for the annotation, thus not having to look through the raw sensor data. Other settings like the previously mentioned behavioral monitoring mostly will not have this option for longer recording times due to privacy reasons.

The fourth HAR specific challenge presented by [8] is data collection and experiment design. When the paper was published, there was no joint effort in the HAR community to collect general purpose datasets of human activity. The design of a HAR experiment is also challenging because the sensors should have minimal influence on the execution of the activity, preparation, and conduction of the ex-

periment. It is time consuming and costly because of the need for participants, experimenters, and equipment. [8]
Sensors itself create another hardship because of the variability in their characteristic. The device itself can have hardware errors and sensor drift, or it can be affected by other things like changing operating temperatures or the movement of the device while performing activities.

2.3 Federated Learning

In Federated Learning (FL) a neural network model is trained by multiple devices at once. This is done by a central server that collects gradient updates $g_k = \nabla F_k(w_t)$ from users and applies the update to its own model by averaging the gradients $w_t + 1 \leftarrow w_r - \eta \sum_{k=1}^{k} \frac{n_k}{n} g_k$. As described in the related work section 1.3, the data was assumed to be private because only g_k is shared with the server, but the original data can actually be reconstructed from g_k in certain cases. In the following, the two FL algorithm FedSGD and FedAVG will be introduced in more detail.

2.3.1 Federated Stochastic Gradient Descent

The Federated Stochastic Gradient Descent (FedSGD) algorithm is used in FL to train a global model via gradient updates ∇W from the clients. On the server side the gradients from all users are averaged, used to update the global model directly and the resend it to the clients. The logic for FedSGD consists of the parameters: C as number of clients participating in an update round, learning rate η, k clients, E local update steps, model w_t, minibatch size B and gradients g_k. In FedSGD E is fixed at 1. Clients compute their individual gradients $g_k = \nabla F_k(w_t)$ on their local data by performing one step of gradient descent. The gradients are then sent to the server, aggregated and the global model is updated by $w_t+1 \leftarrow w_r - \eta \sum_{k=1}^{k} \frac{n_k}{n} g_k$. This update could also be done by averaging the models of the users and then using the averaged model as a global model. [24]

2.3.2 Federated Averaging

The Federated Averaging (FedAVG) algorithm does multiple iterations on the client devices and then sends the new model weights to the server, where the weights from all clients are averaged. The server then uses the averaged weights to update the global model and distribute the new global model to the clients. In FedAVG the parameters and the update logic as explained in FedSGD 2.3.1 are used but the

parameter E is not 1 but > 1 i.e. $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$ is executed E times by each participating user. [24]

2.4 Gradient Inversion Attacks

In a federated learning setting a range of attacks on this framework is possible to extract different types of information. These attacks can be summarized under the generic term Gradient Inversion (GI) because they are heavily dependent on inverting the gradients of the neural network most of the time. This section describes the process of label reconstruction and showcases how the attacks evaluated in this thesis execute their attack. A more detailed explanation of the attacks is given in 3.5.1. In addition, important terms for GI will be introduced, which will be used throughout this thesis.

2.4.1 Problem Setting

The main goal of this thesis is to evaluate the possibility of GI attacks in a HAR setting but we also want to keep our focus on simulating a realistic scenario that could occur in practical settings. Our setting focuses on FL where a NN model is trained by multiple clients *C* with the FedSGD algorithm. The central server *S* receives the gradients from the clients and uses them to update the global model. The shared gradients are assumed to be correct, i.e. they were calculated on correct data and labels by the clients. For the attacks we focus on the weight ∇W_L and bias ∇b_L gradients of the classification layer in the NN models, as shown in 3.2. The leakage of the labels itself is not the most severe problem but the enabling to use further attacks like data reconstruction [14, 52], membership inference attacks [42] and attribute inference attacks [25]. For example in data reconstruction attacks that are optimization based, reconstructed labels allow one to directly optimize for the data and use the labels to ease the optimization process [14].

2.4.2 Threat Model

As threat model the idea of a honest-but-curious server is widely used in the GI literature. [14, 48, 52, 10, 41, 23, 13] The basic idea is a server that operates in the intended way of FL but is also interested in reconstructing information from the data, i.e. the gradients it receives from the participating users. There are also threat models that consider altering the client models to get access to more data as in [43]. We use the same definition as [41] to provide a more detailed description of our threat model, it is defined by the adversary access point, mode, and observation.

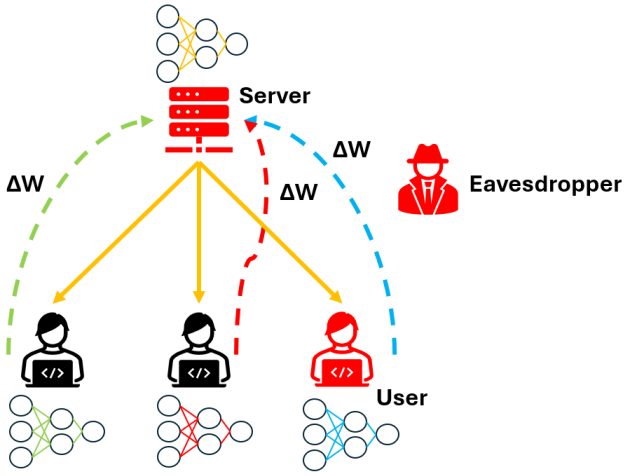


Figure 2.1: Threat model with 3 access points: compromised user, curious server, eavesdropper. Inspired by [41].

Access point When we talk about GI-Attacks, it needs to considered where the Access Point (AP) of the attacker is. The AP is the place where the observer obtains information that it must not have access to and can use to perform a GI-attack. In a FL setting multiple APs can be defined. As shown in 2.1 the AP can be on the client side, the server, or in the form of an eavesdropper between user and server. This thesis focuses on the server acting as AP.

Mode The mode aspect refers to the attacker and how he influences the FL process. Most of the time, a passive behavior is assumed, where the goal of the attacker is to simply extract knowledge from the gained information. Most of the GI-attacks choose this mode and do not interfere with the system in any way. [14, 48, 52, 10, 41, 23, 13]

Observation In [41] the observation parameter in a GI setting is described as the amount of information the attacker can observe. They divide the observed information into three classes:

1. Shared gradients
2. White-box model
3. Auxiliary knowledge

The shared gradient case means that the attacker only has access to the calculated gradients of a user device. For the white-box model, the knowledge of the attacker is extended by the model architecture and parameters. If additional knowledge is part of the process, the attacker also has access to an auxiliary data set. It contains data samples that have the same classes as the training data of the users. The authors argue this as a realistic setting because many datasets, which are used for initial training of a NN, are publicly available.

2.5 Defense

As mentioned in the related work section, there are multiple approaches for defending against GI-attacks. In the following, we briefly recapitulate the different possibilities, explain which technique is used in this work and show how the defense mechanism tries to achieve a better protection.

Techniques As described in the related work section, the following defense mechanisms can be used in FL settings:

- Gradient Compression (GC)
- Secure Aggregation (SA)
- Differential Privacy (DP)
- Lightweight Defenses

When applying GC the goal is to scale down the gradient by replacing single coordinates with 0 if they exceed a threshold. [30] The SA approaches allows to aggregate the user gradients before sending them to the server, thus the server has no access to the individual gradients. [7] In this thesis we focus on the application of Local Differential Privacy (LDP) which is a subclass of DP.
LDP is a defense mechanism that is applied on the user side before sharing the gradients with the central server. In a realistic setting, a privacy budget ϵ determines the amount of noise added in the training process of a neural network. This thesis applies the approach of Gat et al. [13] where no privacy budget needs to be retained because the label reconstruction is executed on single gradients that are not part of a training process. First, the gradients are clipped and afterwards a noise value is added to the clipped gradient to further disguise the original gradient. The clipping of the gradients is done by normalizing the vectors to a L_2-norm of ρ if the norm of the gradient exceeds a defined threshold. Noise is added to the gradients by sampling a value from a Gaussian distribution that has a mean of $\mu = 0$ and a

predefined standard deviation of σ. Another option are lightweight defenses like removing the bias from the classification layer. [36]

Chapter 3

Methodology

In this chapter the methodology and implementation of all relevant components for the HAR sensing setting are explained which includes the datasets, NN-models, FL-algorithms, GI-attacks and defense with focus on DP.

3.1 Repository Overview

The repository is a combination of new code together with implementations of existing repositories. The implementation of the models is taken from the TAL repository and is explained in section 3.3. Federated Learning is implemented with new code for FedSGD, while FedAVG is executed by the breaching repository. The two components are discussed in Section 3.4. For the dataset creation, the logic from the TAL repository is adapted and the sampling methods balanced and unbalanced are added. The datasets are explained in Section 3.2. Many label reconstruction attacks are already implemented by the breaching repository, more recent attacks like iLRG and LLBG are integrated by taking the code from their original implementation, lastly GCDB is integrated into breaching as a new repository. An overview of the attacks is provided in section 3.5.1.

3.2 HAR Datasets

In HAR the datasets can consist of different types of data. For example, the WEAR dataset [4] provides not only sensor data but also video data. As described before, in this thesis only sensor data is considered in the evaluation because GI-attacks focusing on image data have already been extensively researched. [14, 23, 13] The amount of sensors used for recording the activities also often differ from one dataset to another. For example, the WEAR dataset used 4 smartwatches (placed

on wrists and ankles) to record the activities, while the Wetlab dataset used only one wrist worn sensor. [4, 37] For the evaluation, we take two datasets into consideration, which are also used in the TAL repository. [1]

WEAR The WEAR dataset is a collection of recordings of sport activities by different participants. Every participant had to perform a set of 18 activities that can be grouped into 3 categories: running-, stretching-, and strength-based tasks. For the recording, the task was to perform every activity for at least 90 seconds. The activity is not necessarily performed continuously for 90 seconds, but can have breaks. This recording setup leads to a balanced dataset with respect to the activities. Only the NULL-class, the time where no activity is performed, is much more prominent in the dataset. The number of classes for the dataset is the 18 activities plus the NULL class, resulting in 19 classes overall. This is also visualized in diagram4.15 in the evaluation. At the time of evaluation, the dataset contained 18 recordings of different subjects executing the activities. The data was recorded with 4 Bangle.js smartwatches [2], one placed on each wrist and ankle. The four wearable sensors recorded 3D accelerometer data at 50 Hz and a sensitivity of $\pm$ 8g. Each subject recorded is saved in a .csv file that includes the subject number, accelerometer data, and corresponding label for each timestamp. Every smartwatch provides 3 dimensions x, y and z of acceleration data i.e. one timestamp is defined by 12 dimensions when using 4 smartwatches as seen in 3.1. [4]

Wetlab The Wetlab dataset is a collection of recordings of 8 activities performed in a Wetlab. The test subjects were tasked to extract DNA from an onion and tomato. The activities are cutting, inverting, peeling, pestling, pipetting, pouring, stirring, and transfer. The dataset thus has 9 different classes for the dataset, the mentioned activities plus the NULL class. Instruction steps on how to perform this extraction were given to the participants through Google Glass. There was no time limit, resulting in time differences between subjects in the range of 18 to 45 minutes. This results in a more unbalanced dataset compared to WEAR. The dataset contains 22 subjects and is recorded using a wrist-worn 3D acceleration sensor. As in the WEAR dataset the sampling rate of the sensor is 50Hz but the sensitivity is $\pm$ 4g. The data is saved in a .csv file the same way as the WEAR files but containing only 3 dimensions of acceleration data because only one sensor was used for the recording as seen in 3.1. [37]

[1] https://github.com/mariusbock/tal_for_har

[2] https://mariusbock.github.io/wear/

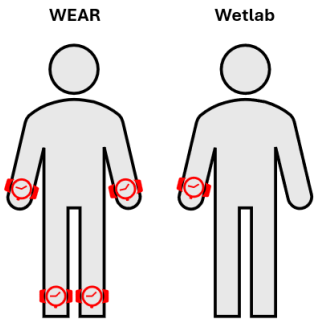


Figure 3.1: Sensor placement (red) in the data collection process for WEAR (left) and Wetlab (right) inspired from [4]

3.3 HAR Models

In the evaluation, the DeepConvLSTM [26] and TinyHAR [54] model are used. The implementation and settings of the two models are taken from the TAL repository. For the DeepConvLSTM 64 convolution kernels with a size of 9 are used. There are 128 LSTM units and 2 LSTM layers. The dropout is set to 0.5. The TinyHAR model consists of 40 convolution kernels for the WEAR dataset and 20 kernels for Wetlab. The kernels have a size of 9 and overall 4 convolutional layers are used. The dropout is set to 0.5. In diagram 3.2 the red X marks the point where the gradients are taken, which is the classification layer for both DeepConvLSTM and TinyHAR. The different inputs of the two datasets are also shown and are the same for both models. Based on the number of input samples, the models predict a class for each sample. Afterwards the results can be taken to evaluate metrics with them, for example accuracy.

3.4 Federated Learning Algorithms

In the breaching repository the two FL algorithms FedSGD and FedAVG are implemented. For this thesis, the FedSGD algorithm is simulated with a new implementation for HAR. The user can adjust a range of parameters like the number of local updates, number of data per local update, local learning rate, and whether the server knows about these hyperparameters.

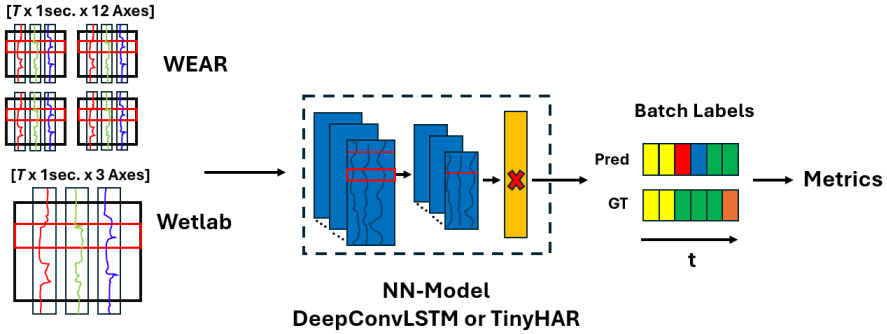


Figure 3.2: In both models the gradients are taken from the classification layer (red X). Figure adapted from [5]. The input provided by the datasets is the amount of samples T, the time frame and the number of axes. The model calculates an activity prediction for every sample T.

3.4.1 FedSGD

For the FedSGD experiments, a new script that uses the label attacks from the breaching repository, the additional added attacks and the new GCD attack was created. In FedSGD only one local update is performed on the user side. Afterwards, the calculated gradients are immediately sent to the server, where they are aggregated by the server. The script simulates this concept for a HAR Sensing setting. A simulation of multiple users was not considered i.e. the iteration of the dataset was performed in sequential order for FedSGD because it would not make any difference, as the users do not influence each other. The FedSGD algorithm 1 starts with sending the initial model of the server to all users. Afterwards, the users run the one local update on their local data and send the resulting gradient to the server. After receiving all updates, the server averages them and updates the global model. This concludes one update round and the server can distribute the new global model again and begin a new update round. In this thesis it is assumed that the server aggregates the gradients i.e. access to the single user gradients is assumed.

Algorithm 1 Federated Learning SGD

Init: Global model m, Number of users n
Send m to all users u
for $i = 0$ to n **do**
    Compute gradient g_i of one user u_i
    Send g_i to server
end for
Server updates m with aggregated value of all g

3.4.2 FedAVG

The breaching repository differentiates between two kinds of FedAVG. In the setting "Multi-User Steps", one user executes multiple SGD steps on the local data and averages the resulting gradients before sending them to the server. In the "Multi-User Aggregate" setting, a number of users do multiple SGD steps on their local data and average the gradients. The averaged gradients are then sent to the server, where they are aggregated. The FedAVG algorithm 2 starts with the initial model sent to all users by the server. Afterwards, the users run the number of local updates on their local data and send the difference in parameters between the initial model and the model after training to the server. After receiving all updates, the server averages them and updates the global model. This concludes one update round and the server can distribute the updated model again, to start a new round. The difference calculation step could also be done by the server, in that case the user sends the complete updated model and the server handles the difference calculation. The breaching repository uses the difference calculation approach.

Algorithm 2 Federated Learning AVG

Init: Global model m, Number of users n, Number of local updates l
Send m to all users u
for $i = 0$ to n **do**
    for $j = 0$ to l **do**
        Do local update step j for user u_i
    end for
    Compute parameter difference d_i of initial m and m after the updates
    Send d_i to server
end for
Server updates parameters of m with averaged differences d from all users

3.5 Gradient Inversion

In the evaluation, the following label reconstruction attacks were used, which we will explain in this section:

1. Analytic (Only for FedSGD)
2. Yin (Only for FedSGD) [48]
3. iDLG (Only for FedSGD) [52]
4. LLG / LLG* [41]
5. iLRG [23]
6. Bias Corrected [3]
7. LLBG [13]
8. GCDB [4]
9. Baseline: Random and EBI [13]

3.5.1 Label Reconstruction Attacks

Analytic The analytic attack in the breaching repository takes all negative values from the bias gradient of the last NN layer and constructs a list of valid classes of them. Based on the classes in this list, one label is added to the list of reconstructed classes. This only allows reconstructing batches with unique values truthfully. Multiple occurrences of a label can not be detected. If the number of valid classes is smaller than the number of data points, the missing number of labels is filled up with random labels as seen in line 6 of Analytic.

Algorithm 3 Analytic

1: **Input:** Bias gradients g, Number of data points n
2: ▷ *Works only with unique labels*
3: Init: $labels$ [ ]
4: Identify: $valid\_classes$ where bias gradients $g < 0$
5: Add: labels += valid classes
6: Random: Fill up with random labels until n is reached
7: **Output:** $labels$

[3] Found in the breaching repository
[4] Newly proposed attack

Yin The Yin attack was proposed in [48] and focuses on the extraction of labels analytically using the weight gradients of the classification layer. The algorithm takes the minimum values along the columns of the weight gradients. This way the algorithm is able to identify which labels are present in the input data. If a negative value is found in a column, the corresponding label is added to the list of reconstructed labels. However, this does not allow an assumption about how many labels of the respective class were used.
This leads to an important restriction of the attack, namely the need for non-repeating labels in a batch. For the vision based setting in [48] this restriction was less problematic because the evaluation dataset was ImageNet with over 1000 classes. For batches with a small size, that keeps the chance of duplicates at a minimum, and this attack will work as intended. For the HAR sensing setting however, this will often be problematic because the datasets have a small number of classes in general, as can be seen with our evaluation datasets WEAR and Wetlab. Additionally a ReLU layer needs to be in front of the classification layer or another nonlinear layer with positive output.

iDLG The Improved Deep Leakage from Gradients (iDLG) attack was introduced by [52] as an enhancement of the original Deep Leakage from Gradients (DLG) attack by [56]. In the original work the extraction of the original labels was not considered, only the reconstruction of the original data by optimizing the distance between dummy gradients and real gradients was done. Zhao et al. focused their approach on extracting the ground truth labels first and use them for improving the reconstruction process. They analyzed the weight gradients to show that the ground truth label must posses a negative value assuming that non negative activation functions have been used.
Their label reconstruction is generalized for a one sample batch, thus the attack only works for a maximum of one data point. This was extended to arbitrary batch sizes in [41].

LLG / LLG* In [41] a new attack with the name Label Leakage from Gradients (LLG) was introduced. The attack focuses on exploiting the last layer of a neural network through its weight gradients. The authors used two properties related to weight gradients to justify their attack approach. For the first property they showed that from

$$\nabla W_L^i < 0,$$

the existence of label *i* in a training batch can be followed, where

$$\nabla W_L^i$$

is the weight of the last layer of a NN. In the second property they used the gradient

$$g_i = \mathbf{1}^T \cdot \nabla W_L^i$$

to show a relation of the gradient magnitude to the label *i*. For an untrained model, the number of samples in a training batch that resemble the label *i*, is almost proportional to g_i. They introduce a parameter *m* which is the impact that a single label of class i has on the value of g_i. The algorithm for the basic LLG attack is displayed in 4. Two variations of the LLG algorithm are LLG* and LLG+. In LLG* the attacker has additional information about the model architecture and parameters. For LLG+ an additional auxiliary dataset is used by the attacker. In this thesis only the standard LLG and LLG* are considered, as they provide a more realistic setting for HAR. Creating an auxiliary dataset for LLG+ is not as simple in HAR as for other modalities because datasets are expensive to create and not as freely available. [8, 6]

Algorithm 4 LLG

Input: Weight gradients g, Number of data points n
▷ *Stage 1: Compute sum of weight gradients*
Init: $weights$ from the gradient g; $labels$ [ ]
Compute: Single $weight$ gradient as sum of every class $weightgradient$
Identify: Add labels of $weightgradients < 0$ to $labels$ ▷ *Stage 2: Extract labels*
while length$(labels) < n$ **do**
    Select label where g is minimum
    Append label to $labels$
    $g = g - m$
end while
Output: $labels$

iLRG In [23] an attack named Instance-wise Label Restoration from Gradients (iLRG) was proposed, that tries to solve a system of linear equations to extract the labels of batch-averaged gradients. The algorithm consists of three steps: (1) The class-wise embeddings are reconstructed by calculating a quotient of two weight and bias gradients that are taken from the classification layer. (2) This quotient is then given to the classification layer to calculate the probabilities after applying the softmax function. (3) After gaining the probabilities, a linear equation system is created and solved with the Moore-Penrose inverse because the system is not in a square shape. The result is the number of occurrences of each label in the batch. The pseudocode is shown in 5. For this thesis, we took the implementation from

the original repository [5] and integrated it into the breaching repository. Ma et al. state that the attack is designed for untrained models and is performing worse for trained models because a couple of necessary approximations are anticipated to not hold anymore in the trained scenario, which are often true for the untrained case. One is that the distribution of embeddings e needs to be uniform and condensed over a specific sample class B_i of the batch B. This allows to replace them with the arithmetic mean of the class. [23] The second approximation is that gradient row i of a batch averaged gradient needs to be mostly determined by samples from class i in the batch. [23] Lastly, a third approximation assumes that the average post-softmax probability from a number of samples from class j is similar to one created by j-class average embedding. [23] Additionally, a classification model with softmax activation is necessary. [23]

Algorithm 5 iLRG

Input: Bias gradients g, Number of data points n
Init: Probabilities (uniform distribution), coefs [ ], values [ ]
Compute: avg_bias as mean of $bias\_per\_query$
for every class **do**
    Construct coefficients
    Construct values based on bias
    Construct equation from coefficients and values
end for
Solve equation system with pseudo-inverse
Filter negative results
Round results
Construct labels from results
Output: $labels$

LLBG The Label Leakage from Bias Gradients (LLBG) attack was recently introduced by Gat et al. [13] and takes advantage of the bias gradients of the last NN layer for label extraction. They present two versions for the attack, one for untrained models and another for trained models. Due to time constraints, we only present and use the first version in this thesis. Furthermore, the version for trained models requires information about the model prediction accuracy, but we decided to provide the same amount of information to all attacks. The untrained model LLBG version consists of two stages. In the first stage, all labels that are guaranteed to be in the input batch are added to the list of reconstructed labels. Each added

[5] https://github.com/BUAA-CST/iLRG

label means that the corresponding bias gradient is reduced by an estimated impact m, similar to the LLG attack. In the second stage, the bias gradient with the lowest value is selected and the corresponding label is added to the reconstructed list. Afterwards, the gradient value is increased by the impact m. If the selected gradient still has the lowest value, it remains selected otherwise the algorithm switches to the minimum bias gradient. This process is repeated until the number of data points is reached, which needs to be known by the attacker. The LLBG attack is basically the LLG attack adapted for bias gradients. For this thesis, we took the implementation from the original repository [6] and integrated it into the breaching repository. The authors also used the breaching repository for their implementation and experiments.

GCDB The Greatest Common Divisor Bias (GCDB) attack is also based on the assumptions from [41, 13], where the magnitude of the gradients is proportional to the number of occurrences of a label in a training batch. These assumption led to the idea to extract the mentioned impact parameter m from the bias gradients of the last NN layer. [41]
In the process of this thesis, new ideas were tested to estimate m. This attack leverages the relationship between the single components of the bias gradient. We subtract every component once with every other component in the gradient and the search for the minimum value. If there are two single components that differ only by one occurrence in their number of labels, the difference between them should be close to m. Using this basic idea of estimating m in GCDB, choosing the correct difference is done by taking all calculated differences and run a search for the greatest common divisor (gcd). The gcd value is then used for m and the labels are estimated they same way as in LLG and LLBG [41, 13].

[6] https://github. com/nadbag98/LLBG

Algorithm 6 GCDB

1: **Input:** Bias gradients g, Number of data points n
2: ▷ *Stage 1: Compute bias and valid classes*
3: Init: $bias\_per\_query$ from the last gradients of g; $labels$ [ ]
4: Compute: avg_bias as mean of $bias\_per\_query$
5: Identify: $valid\_classes$ where $avg\_bias < 0$
6: Calculate GCD candidate for every class and use as m
7: Use $m = \sum average\_bias[valid\_classes]/n$ in case of GCD estimate > 0
8:
9: ▷ *Stage 2: Extract labels*
10: **while** length$(labels) < n$ **do**
11: Select label l with minimum avg_bias
12: Append l to $labels$
13: avg_bias[idx] $- gcdCandidates[label]$
14: **end while**
15: **Output:** $labels$

Bias Corrected The Bias-Corrected attack is a work in progress approach, which can be found in the breaching repository and is similar to the approach of the LLG-attack. [41] For the attack the mean of the bias gradients of the last layer are used. For every valid class, a label of the respective class is added to the list of extracted labels. A valid class is a class that has a bias gradient below zero. Similarly to the impact m in LLG, an impact is estimated in Bias Corrected by dividing the average gradient by the number of data points. This completes the first stage of the algorithm. In the second stage, the lowest bias gradient is selected and added to the extracted labels list. Afterwards, the bias of this specific gradient is increased by the impact. This process is repeated until the number of data points is reached. Consequently, it is necessary to know the number of data points in a batch for this attack, just like in the LLG attack, otherwise the algorithm would not know when to stop. The implementation of this attack is the same as in EBI but without the sorting in line 7.

EBI The Empirical Bias Impact (EBI) baseline that is used in [13] to extract labels from bias gradients tries to estimate the impact of a label empirically. In that regard, it is similar to the LLG attack but uses the algorithm of LLBG. The authors used this approach as an ablation attack, to showcase the superiority of their impact estimation compared to a simple empirical method. For the estimation

of the impact the following term is used:

$$m = \frac{1}{B} \sum_{i;\, \beta_i < 0} \beta_i$$

where B is the batch size, β is the complete bias gradient $\nabla_b L$ and i is used to indicate a specific component of the gradient. The term also constrains the components used with the condition $\beta_i < 0$. The implementation is basically the same as the BC attack, but it sorts the valid classes by magnitude before adding the to the label list (line 7 in EBI).

Algorithm 7 EBI

1: **Input:** Bias gradients g, Number of data points n
2: ▷ *Stage 1: Compute bias and valid classes*
3: Init: $bias\_per\_query$ from the last gradients of g; $labels$ [ ]
4: Compute: avg_bias as mean of $bias\_per\_query$
5: Identify: $valid\_classes$ where $avg\_bias < 0$
6: Sort: $valid\_classes$ based on $bias$ values in ascending order
7: **for** $i = 0$ to $\min(n, \text{length}(valid\_classes)) - 1$ **do**
8: Append the i^{th} smallest valid class to $labels$
9: **end for**
10: Compute impact $m = \sum average\_bias[valid\_classes]/n$
11: Adjust: $avg\_bias[valid\_classes]$ by subtracting m
12:
13: ▷ *Stage 2: Extract labels*
14: **while** length$(labels) < n$ **do**
15: Select label l with minimum avg_bias
16: Append l to $labels$
17: $avg\_bias[\text{idx}] - m$
18: **end while**
19: **Output:** $labels$

Random Baseline As baseline we use random guessing. It is important to note that the performance of the random guess in this setting depends on the number of classes of the evaluated dataset. The WEAR dataset consists of 19 classes, resulting in a $1/19$ chance of guessing the correct label. For the Wetlab dataset, with just 9 classes, this chance is $1/9$. However, this is not the performance that can be expected in our setting because the label distribution does not have to be in the same order as the labels in the original data. The following example will illustrate this:

Consider a small sample batch with the labels $[1, 2, 3]$. For the case where the baseline predicts the labels $[2, 3, 1]$, this result still yields an accuracy of 100% because the other label attacks can also not reconstruct the original labels in the correct order.

In table 3.1 the different attacks are listed together with the gradients they use, preferred batch size, additional knowledge and general assumptions. All gradients mentioned are from the classification layer of a NN model. Most of the attacks need a proportional scaling of the gradients in accordance to the labels, that is why no attack is specifically designed for trained models.

Attacks	**Gradients**	**Batch Size**	**Knowledge**	**Assumptions**
Analytic	Bias	Arbitrary	N	Unique Labels
iDLG	Weights	1	N	-
Yin	Weights	Arbitrary	N	Unique Labels
iLRG	Bias	Arbitrary	N	Untrained Models, Embedding approx.
Bias-Corrected	Bias	Arbitrary	N	Untrained Models
GCDB	Bias	Arbitrary	N	Untrained Models
LLBG	Bias	Arbitrary	N	Untrained Models
LLG	Weights	Arbitrary	N	Untrained Models
LLG*	Weights	Arbitrary	N	Untrained Models, Model Knowledge
Random	-	Arbitrary	N	-
EBI	Bias	Arbitrary	N	-

Table 3.1: Overview of all attacks used in the thesis. Showing gradients used in every attack, intended batch size, necessary knowledge for correct execution and additional assumptions to guarantee good results. N is the Number of Data Points.

3.6 Sampling

Following the evaluation by Zhang et al. [51] with five different sampling methods: random, uniform, single, subclassed and imbalanced, we use the 4 strategies: $sequential$, $shuffle$, $balanced$ and $unbalanced$. Due to time constraints the subclassed method is not used but the sequential method could be interpreted as a special case because it also never contains all classes.

Sequential The $sequential$ sampling runs over the provided data in the correct order. We replace the single sampling of [51] with this method because in the

WEAR and Wetlab dataset the activities do not change as often for a maximum batch size of 100. This means that the sequential method effectively replaces the single sampling.

Shuffle In the $shuffled$ approach, the entire subject is iterated in random order, which is the same as the random sampling in [51].

Balanced Our uniform method is called $balanced$ in this thesis and has the objective of sampling the same number of labels for every class as good as possible. If that is not possible, random labels are chosen.

Unbalanced The $unbalanced$ sampling takes 50% of the data from one class A, 25% from another class B and the last 25% are sampled randomly. This sampling strategy is also used in [41, 13]

3.7 Defense

The defense in our evaluation consists of Local Differential Privacy (LDP). It is implemented through gradient clipping and noise addition. Due to time constraints, the noise addition is fixed at a value of 0.1. For clipping, the values 1.5, 1.0 and 0.5 will be evaluated. The reason for fixating noise but to vary clipping is that a higher noise will further degrade the success of label reconstruction attacks. For clipping, it is not as easy to estimate when a point is reached where it has a sufficient influence on the attacks. In [13] the same approach is used for applying LDP because there is no training process of the model and the label attacks are only evaluated on single batches. As a result of this, there is also no privacy budget ϵ that needs to be retained. That is the reason why this thesis follows the same approach for all settings. When clipping is applied, the magnitude of the overall gradient is computed and scaled down to the clipping value afterwards the noise value is applied. The algorithm is shown in 8. The actual implementation is taken from the breaching repository and used for both FedSGD and FedAVG.

Algorithm 8 Differential Privacy

1: **Input:** gradient g
2: Norm: Build global norm gn of g
3: Clipping: Clip gn based on clipping value
4: Restore: gn is restored to format of g
5: Noise: Add noise to g
6: **Return:** gradient g

Chapter 4

Experimental Evaluation

4.1 Setup

In the following the experimental setup is explained, focusing on the data and models used, the FL-algorithms, the evaluation metrics and the baselines.

4.1.1 Data and Models

For the evaluation, the WEAR and Wetlab datasets are used. The WEAR dataset can be predicted fairly reliable with a $F1$ score slightly higher than 80% in different settings, as seen in [5]. As an opposing dataset, the Wetlab dataset was selected because it only achieved $F1$ scores around 35% in different settings in [5]. In this way, it can be checked if the prediction performance has a potential influence on the accuracy of leaked labels.

4.1.2 FL-Algorithms

FedSGD The FedSGD algorithm is simulated by simply running one local training step and then passing the resulting gradient to the label reconstruction attacks. In practical settings, an aggregation of the gradients from all users is executed on the server side. We assume that the server is responsible for this aggregation and thus has access to all individual gradients. Keeping this in mind, we do not simulate multiple users but simply run over the entire WEAR and Wetlab dataset. For every created batch of size (100 or 10 or 1) x 50 x (12 or 3) the gradient is computed and the label reconstruction executed.

FedAVG As previously mentioned, additional parameters are introduced in FedAVG, namely:

1. User: Local Updates, Multi User Aggregate
2. Number of local updates: 2, 5
3. Number of users: 2, 5

Local Updates is a setting where again only one user is participating in a single update round, but the training step runs more than 1 local update before sending the gradients to the server. The number of local updates is the corresponding value but is also used in the Multi User Aggregate configuration. Two different settings are considered, where the number of local updates are 2 or 5. Multi User Aggregate is basically the Local Updates setting but with multiple users participating in an update round, where the gradients now are aggregated from all users before giving them to the server. This setting is tested with 2 or 5 users.

4.1.3 Metrics

For the evaluation the Label Number Accuracy (LnAcc) metrics are used on different granularity levels. This granularity levels are subject level, attack level and Batch Size (BS) level where subject level are the results for a single subject of a dataset, attack level are the results for one attack over all subjects of a dataset. The BS level results of an attack at the BS level are calculated on all subjects of a dataset for a single batch size but including all sampling strategies. A second evaluated metric is the Label Existence Accuracy (LeAcc) and can be found in the appendix B because the results of LnAcc are of greater interest.

Label Existence Accuracy The LeAcc metric is used to indicate how well a label leakage attack assessed the present dataset classes in a batch. It counts the amount of correctly identified classed and returns a percentage value based on the number of classes of the dataset.

$$CorrectPredictedUniqueLabels/NumberOfUniqueBatchLabels = LeAcc$$

Label Number Accuracy The LnAcc metric is used to reflect the accuracy of the reconstructed labels in a batch. It counts the amount of correctly identified labels and returns a percentage value based on the number of labels in the original batch.

$$CorrectPredictedLabels/BatchSize = LnAcc$$

4.1.4 Baseline

The baselines used are the random distribution that was described in the label attacks section (3.5.1) and the also previously described EBI attack.

4.2 Experiments

The covered experiments involve various settings for the parameters:

1. Model architecture: DeepConvLSTM, TinyHAR
2. Dataset: WEAR, Wetlab
3. Batch size: 1, 10, 100
4. Data sampling: Balanced, Unbalanced, Shuffle, Sequential
5. Training stages: Trained, Untrained

In the following, one specific combination of these parameters will be referred to as a run. For example, one run would be the setting consisting of "DeepConvLSTM, WEAR, 1, Balanced, Untrained". In table 4.1 an overview of the number of runs for the three settings is listed, accumulated resulting in 960 different runs. In the FedSGD experiments, we iterate over the entire dataset, since there is only a single batch per label reconstruction call.
In the FedAVG runs the aforementioned parameters: users, number of local updates, and number of users are used. As the number of runs increases tremendously with each additional parameter, the FedAVG setting is restricted to a batch size of 100. This still allows for helpful insights because it is the most realistic setting in terms of batch size. Even for HAR sensing settings, where the datasets are way smaller than image datasets, a batch size of 1 or 10 is unlikely. If differential privacy is also applied, the additional parameters noise and clipping are added, again increasing the number of run configurations.

	FedSGD	FedAVG Local	FedAvg Mult
Standard	96	64	128
DP	96	192	384

Table 4.1: Number of runs for each setting. FedAVG runs are higher in number because of the additional parameters $localUpdates$ and $numberOfUsers$. The FedSGD runs with differential privacy are restricted to batch size 100. All FedAVG runs are restricted to batch size 100.

4.2.1 Settings and Caveats

Dataset creation settings Following the previous description of the WEAR and Wetlab dataset in 3.2, these settings are used for the FedSGD evaluation. Every data sample consists of 50 data points each with 12 (WEAR) or 3 (Wetlab) values i.e. a single data/label sample is of size $50\text{x}12(3)$. The window overlap is fixed at 50. In FedSGD every dataset is completely evaluated, the subjects are not mixed. This means for the shuffle, balanced and unbalanced sampling method only data from a single subject recording is combined.
FedAVG in the configuration with multiple local updates and one user follows the same approach and calculates the results for every subject separately.
In the FedAVG setting with multiple users, not the complete dataset is used, but three different selections of individual subjects. Based on the results of the FedSGD experiments, the five subjects with the worst and least leakage were chosen for evaluation, as well as another setting containing 5 users with average leakage.

Model Training Hyperparameter settings As this thesis combines the setting of HAR sensing with GI-attacks, there are plenty of hyperparameters to keep in mind, which have a potential influence on the results. In the following, the parameters used in the model training process are listed and explained. The training was carried out through the TAL repository with hyperparameters as shown in 4.2.

Hyperparameter settings					
lr	seed	epochs	sampling rate	window size	window overlap
0.0001	1	100	50	50	50

Table 4.2: Hyperparameter settings for HAR model training

The learning rate was set to $1e^{-4}$ and weight decay to $1e^{-6}$, the optimizer in use is Adam. All trained models were run for 100 epochs and with the Leave One Subject Out (LOSO) method, so that each subject has an individual model. For the evaluation of the GI attacks the saved checkpoints were loaded for each subject and the dataset initialized with the same configuration.

Caveats A number of caveats have to be considered while reading the results presented in this thesis. The different experiments were carried out on a student PC belonging to a chair of the University of Siegen and the OMNI-Cluster [1] provided by the University of Siegen. As time and resources were limited, the experiments

[1] https://cluster.uni-siegen.de/

could only be done once and the hyperparameters were mostly fixed. For experiments of batch size 1, with the same model and same training stage but different sampling techniques, a similar result is expected because only the order of the labels should have changed. However, the scores showed small differences in averaged results, which can be attributed to the random factor in some attacks. The FedAVG results are preliminary results because FedSGD is evaluated in more detail, to show the influence of the model, training, dataset and sampling strategy in HAR for a simpler setting. The trained models were all taken at 100 epochs runtime although the best performance of the model could be at an earlier stage and possibly have a bigger influence on the label reconstruction attacks.

4.2.2 Baseline Experiments

The presentation of the experiments will start with the random baseline and the EBI attack.

Random In 4.1 the baseline of the random attack can be seen for WEAR and Wetlab across the three batch sizes 1, 10 and 100. The results in 4.1 are averaged over all sampling strategies, models and training stages. For a BS of 1 the results are centered around $1/NumberOfClasses$ as expected beforehand.
As BS increases to 10 and 100, the random baseline LnAcc also increases because the label occurrences of every class are more evenly distributed for each batch. At BS 100 the difference in the number of classes has evened out almost completely because the BS exceeds the number of classes for both datasets by a large margin.

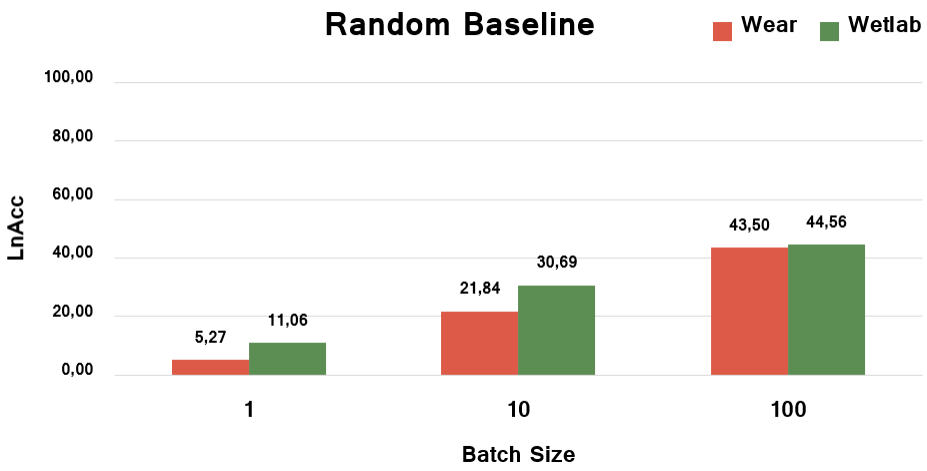


Figure 4.1: Random baseline for WEAR and Wetlab across different batch sizes

EBI In 4.2 the baseline of the EBI attack can be seen for WEAR and Wetlab across the three batch sizes 1, 10 and 100. The results in 4.2 are averaged over all sampling strategies, models and training stages. In contrast to the results of the random baseline, this attack starts with a perfect LnAcc for BS 1 but gradually decreases as the BS increases. This allows to make the assumption that the empirical estimate of the impact m loses accuracy with increasing BS.

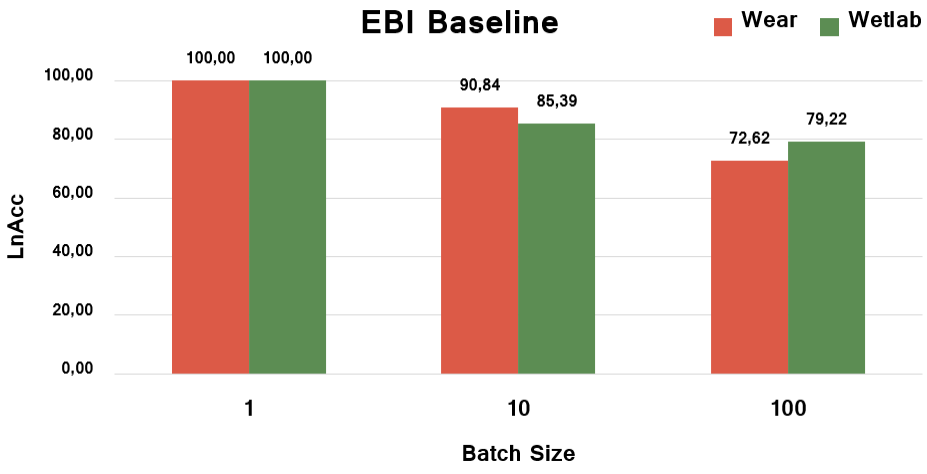


Figure 4.2: EBI baseline for WEAR and Wetlab across different batch sizes

4.2.3 Influence of model training

This section shows how the training stage of the model influences the performance of an GI-attack. In the diagrams 4.3, 4.4 all runs with an untrained or trained model and the same dataset are averaged. In addition, only runs with a BS of 100 are taken into account.

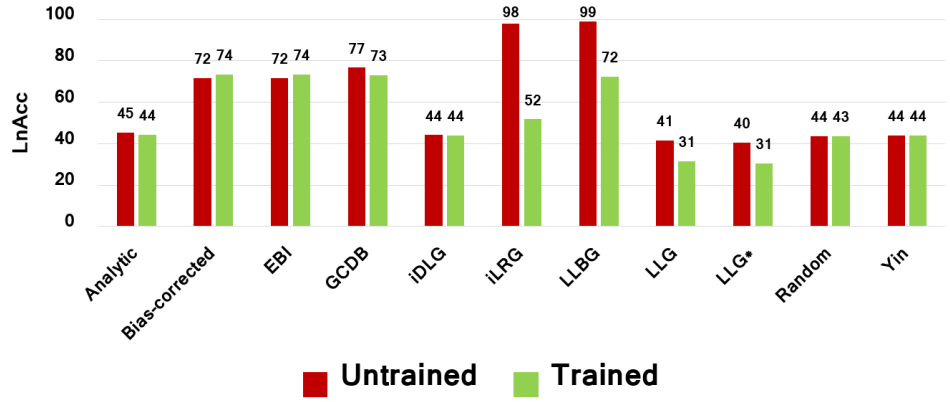


Figure 4.3: WEAR results for runs with trained and untrained model. The averaging is done for runs with batch size 100.

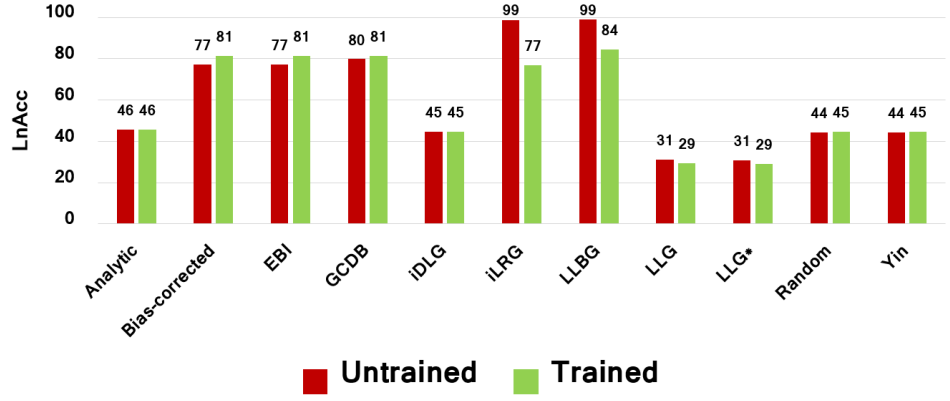


Figure 4.4: Wetlab results for runs with trained and untrained model. The averaging is done for runs with batch size 100.

The attack histogram shows a similar shape for both datasets when using an

untrained model. In both cases, an almost perfect score can be achieved with the iLRG and LLBG attack. For the bias impact attacks BC, EBI and GCDB the results are similar with GCDB performing $2,7\%$ better in Wetlab and 5% better in WEAR. The Yin, Analytic and iDLG attacks are made for settings where the batch consists of a unique set of labels or restricted to one sample. For batches with more data points than classes in the corresponding dataset, the labels exceeding the number of classes are reconstructed randomly. This fact is reflected in both the untrained and trained results, as the performance is close to the random baseline. The LLG and LLG* attack perform below the random guess for both trained and untrained models. The decline in accuracy for iLRG attacks on trained models is expected, this was also shown by the creators [23]. Unexpected is the result for the trained Wetlab setting 4.4, where the attack still remains at a reasonable accuracy around 77%.

4.2.4 Varying Batch Size

Changing the batch size of the data samples used in the NN model does have an influence on the performance of label reconstruction attacks, as shown in works like [41, 13]. This section will show how attack performance changes for batch sizes of 1, 10 and 100. For the sake of keeping the evaluation in a reasonable scope, all runs are grouped by batch size and dataset. Afterwards an average LnAcc and LeAcc value is calculated over all runs for every attack.

Size 1 In 4.5 the performance of the attacks for a batch size of 1 is shown, on the left side the WEAR dataset is displayed, and on the right side of the bars the Wetlab results.

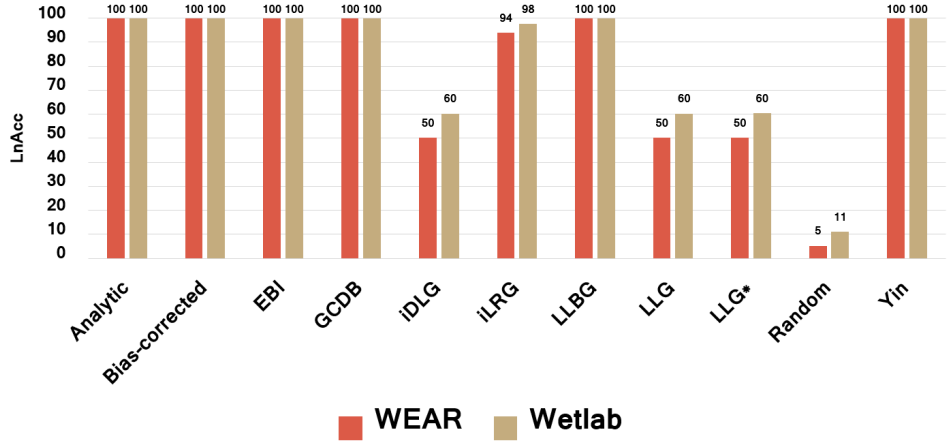


Figure 4.5: Results for all runs of WEAR and Wetlab where BS is 1.

The results for BS 1 are mostly as it could be expected, for example analytic attacks like Analytic and Yin extract labels perfectly. Bias gradient attacks also deliver perfect results for both datasets and the pseudoinverse attack also performs well, but not perfect, with 93% in WEAR and 97% in Wetlab. The most surprising result for this setting is the bad performance of iDLG, LLG and LLG*. They yield results around 50% for WEAR and 60% for Wetlab. What is striking is the fact that all attacks with bias gradients work better than ones with weight gradients.

Size 10 In 4.6 the performance of the attacks for a batch size of 10 is shown, on the left side the WEAR dataset is displayed, and on the right side the Wetlab results. Compared to BS 1 every LnAcc value decreased significantly in both datasets except the random baseline. Especially Analytic and Yin drop by more than 50% because they are not designed for bigger batches with multiple occurrences of the same label.

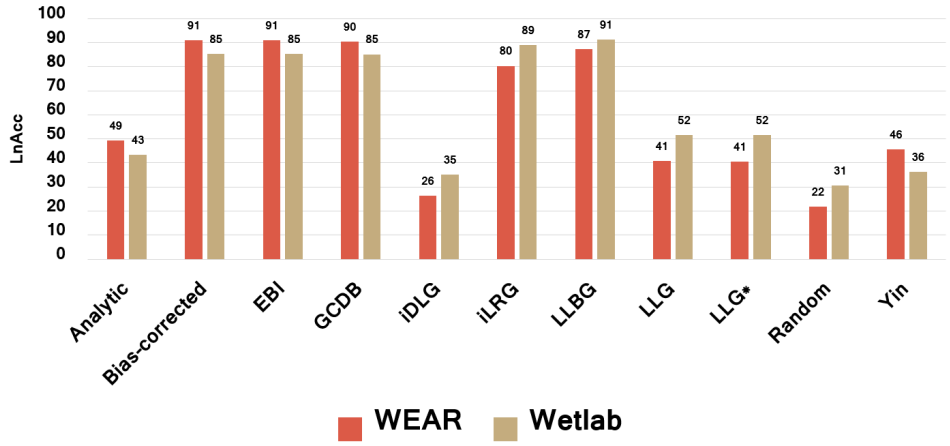


Figure 4.6: Results for all runs of WEAR and Wetlab where BS is 10.

Size 100 In diagram 4.7 the performance of the attacks for a batch size of 100 is shown. For size 100 the trends of the previous size do not continue, as Wetlab now shows the most leakage for the top five attacks. In addition, the accuracy for LLBG improved by 1% and iLRG only lost 1%. Although the batch size increased by factor 10, these two attacks remained stable in their accuracy. The other bias gradient attacks on the other hand, tanked a lot of their LnAcc score, decreasing around 15% for WEAR and 5% for Wetlab. Here, the trend of Wetlab being more prone to leakage is shown once again.

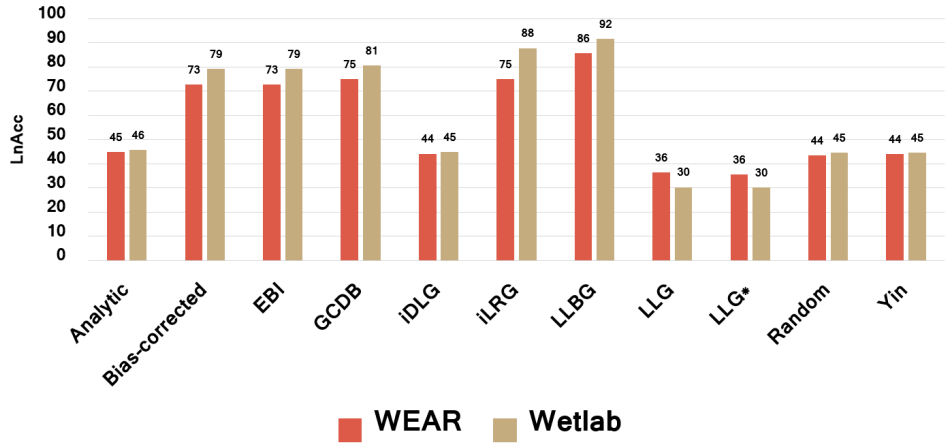
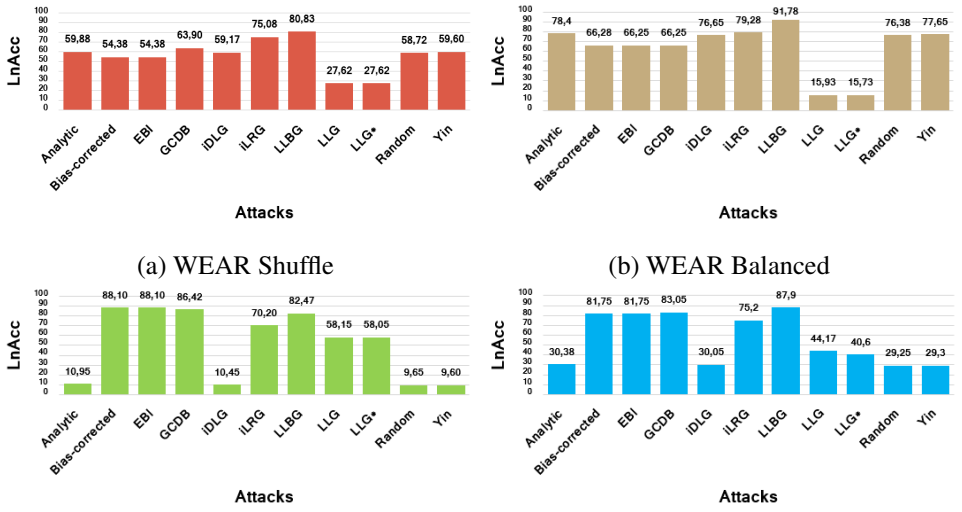


Figure 4.7: Results for all runs of WEAR and Wetlab where BS is 100.

4.2.5 Influence of Label Distribution

In this section the influence of the sampling method on the attack performance is discussed. The four chosen sampling methods in the experiments are sequential, balanced, unbalanced and shuffled. For each sampling method, a separate plot is shown, once for the WEAR dataset 4.8 and once for Wetlab 4.9. For the plots, all runs with the same sampling method, the same dataset and BS 100 are selected and the average LnAcc for every attack on all these runs is calculated.

WEAR The WEAR dataset shows significant differences in the performance of nearly all attacks, depending on the applied sampling method on the underlying batch. For attacks that have a random aspect like Analytic, iDLG and Yin the accuracy is close to the random baseline for sequential sampling. This comes from the fact that there is a high number of labels of one class in a sequential batch. This increases for the other sampling methods, where the class occurrences are more evenly distributed. Especially for the balanced batch it exceeds 75% and therefore the LnAcc of the bias gradient attacks except LLBG. The shuffled sampling for WEAR also has a high accuracy for attacks with a random component, reaching almost 60% and again outperforming most attacks except LLBG, iLRG, GCDB. The reason is probably the relatively evenly distributed number of classes in the underlying subjects of the dataset.



(a) WEAR Shuffle

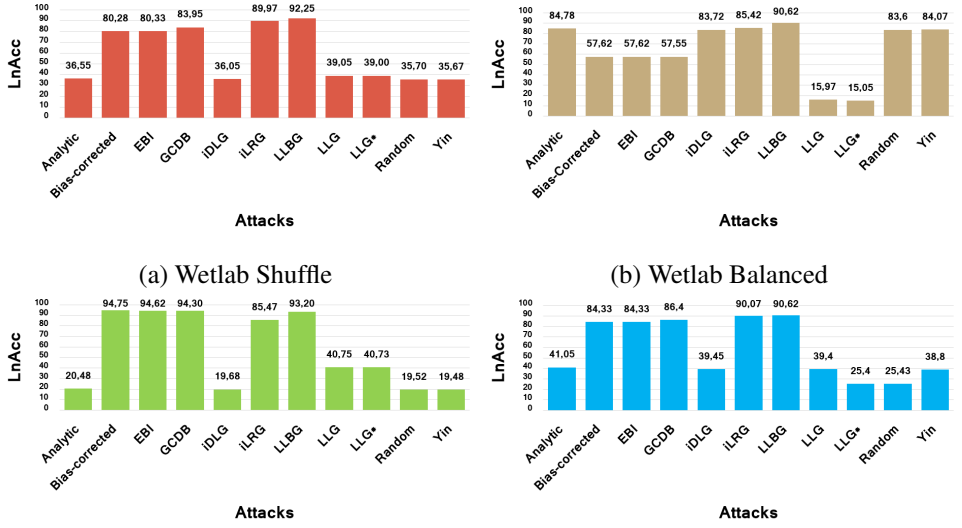
(b) WEAR Balanced

(c) WEAR Sequential

(d) WEAR Unbalanced

Figure 4.8: Results of different sampling methods for WEAR with batch size 100

Wetlab The LnAcc score for different sampling methods also changes while using data from Wetlab, as seen in 4.9. For most of the 4 histograms the scores are similar. The b, c, and d histograms show a distribution similar to WEAR but still yield some important differences. In sequential sampling, the LLBG attack is outperformed by 1% compared to the other three bias attacks BC, EBI and GCDB. Shuffle sampling also shows better results in the Wetlab evaluation for these three attacks, as they all increase by at least 20%. On the other hand, the performance for balanced sampling decreases nearly 10%.



(a) Wetlab Shuffle

(b) Wetlab Balanced

(c) Wetlab Sequential

(d) Wetlab Unbalanced

Figure 4.9: Results of different sampling methods for Wetlab with batch size 100

4.2.6 Models

Another aspect for comparison are the two different HAR models and the question if leakage is higher for a simpler model like TinyHAR compared to the more complex DeepConvLSTM architecture.

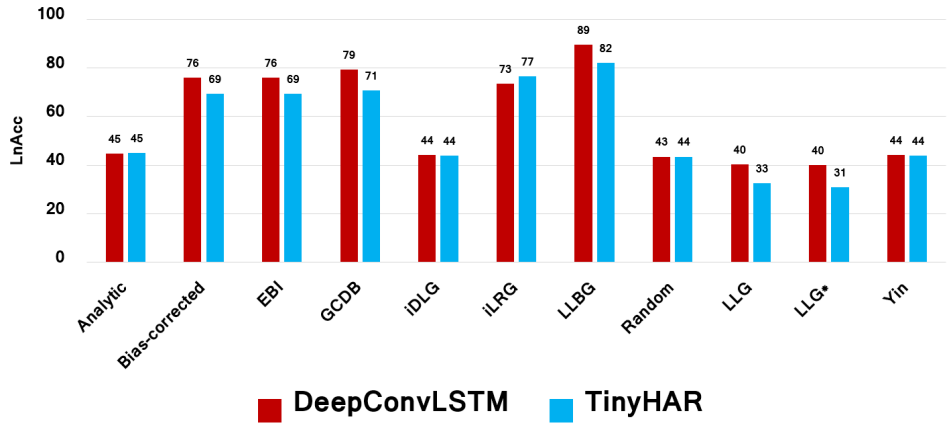


Figure 4.10: Model performance of DeepConvLSTM and TinyHAR for WEAR

The diagram 4.10 shows no difference between the models for attacks with random aspect, namely Analytic, iDLG, Yin and the random baseline. This also happens in the Wetlab diagram 4.11. In both diagrams, the top 5 attacks perform better for DeepConvLSTM with the exception of iLRG, where both times TinyHAR has a higher LnAcc score. A surprising difference can be seen for the LLBG attack, as the performance for both models in Wetlab only deviates by 1% but for WEAR it is a 7% difference. Another interesting observation is the performance of the LLG/LLG* attack, in both datasets the DeepConvLSTM has a higher LnAcc score. As the two attacks rely on weight gradients, this indicates a higher leakage risk of weight gradients in the DeepConvLSTM. For both cases, the score still is below the random guess, so the relevance of this is questionable and needs to be further investigated. In general, it can be concluded that a more complex architecture like the DeepConvLSTM does not show a worse leakage compared to the simpler TinyHAR model.

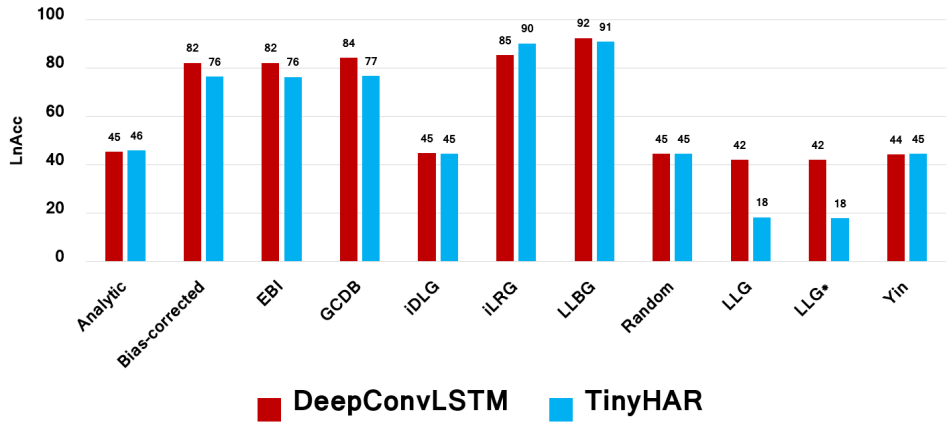


Figure 4.11: Model performance of DeepConvLSTM and TinyHAR for Wetlab

4.2.7 Subject Deviations

Previously the evaluation did not focus on individual subjects but averaged them for the LnAcc score. This section will catch up on this and show some of the deviations that occur between the subjects. For the WEAR dataset, 18 subjects are compared with each other and for Wetlab 22 subjects.

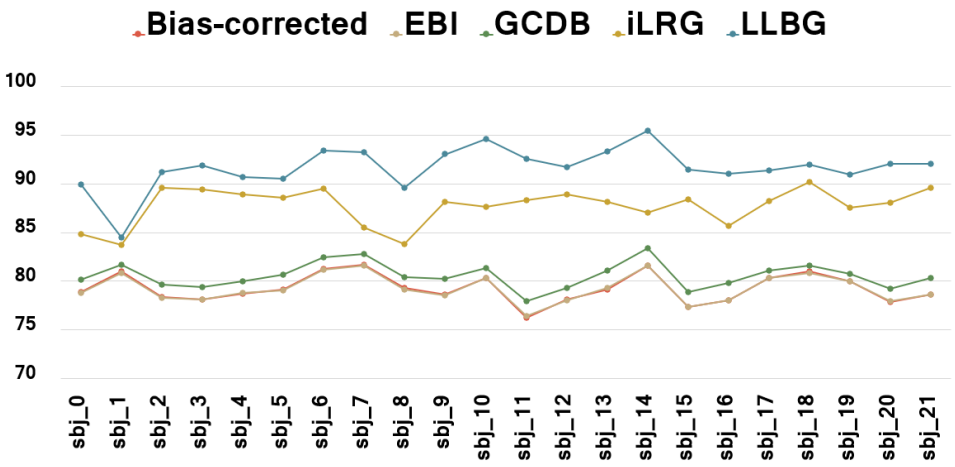


Figure 4.12: Performance of Wetlab subjects for BS 100. LLBG has the best performance for all subjects, sometimes varying strongly. Subject 1 performs about 10% worse than subject 14.

In 4.12 the 22 subjects of Wetlab are shown in a line diagram, where each line symbolizes an attack. The best attack is LLBG, as it outperforms all other attacks for each subject. Surprisingly, there is a high variation of the LnAcc score between a few subjects. Subject 1 has the lowest leakage at roughly 85%, but subjects 10 and 14 show very high leakage around 95%, a difference of 10%. A comparison of the CSV files of the best and worst subject shows that their number of labels vastly differs in the NULL class, see diagram 4.13. This leads to the assumption that a more unbalanced distribution in the subject data could be the reason for a worse LnAcc average score.

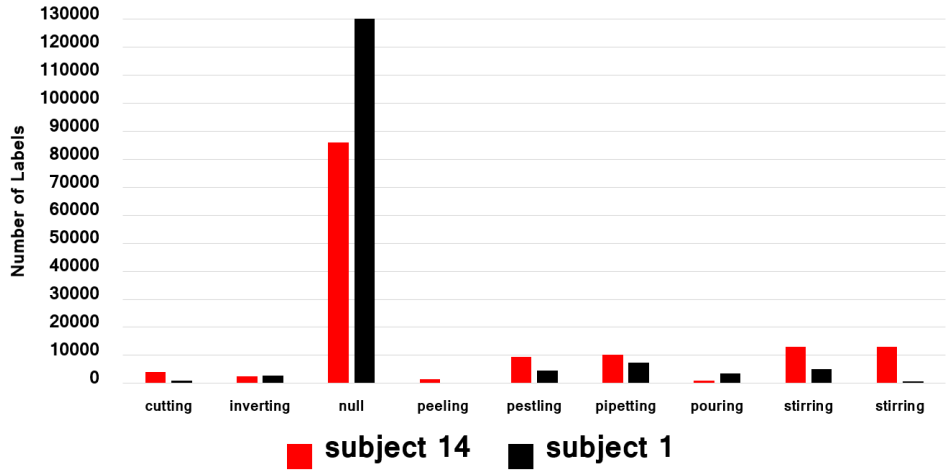


Figure 4.13: Label distribution for best and worst performing subject in Wetlab. Subject 1 has no peeling data because it was skipped in the recording.

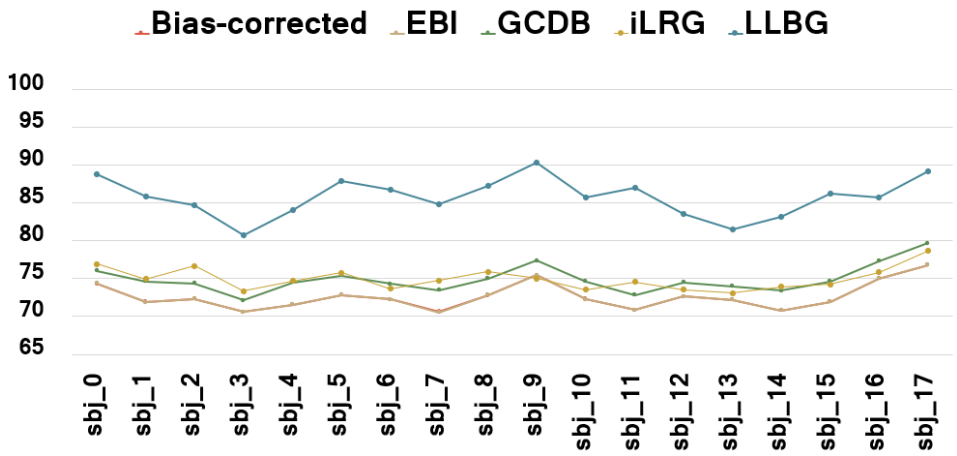


Figure 4.14: Performance of WEAR subjects for BS 100. LLBG shows the best results for every subject. The performance between different subjects can vary a lot. The difference between subject 9 and 3 is roughly 10%.

In diagram 4.14 the 18 subjects of WEAR are shown in a line diagram, where each line represents a different attack. For WEAR the best attack is also LLBG and is again the best for every subject. Surprisingly, all attacks besides LLBG show a

comparable curvature and perform similarly for most subjects, in terms of which subjects perform good and poor for the individual attack. For bias gradient attacks, BC, EBI, GCDB this is expected, as the attacks are very similar. The iLRG attack on the other hand uses a completely different approach, but is mostly comparable to the GCDB curve. In LLBG the deviations between some subjects are nearly 10%, for example, subject 9 roughly at 90% LnAcc and subject 3 only slightly above 80%.

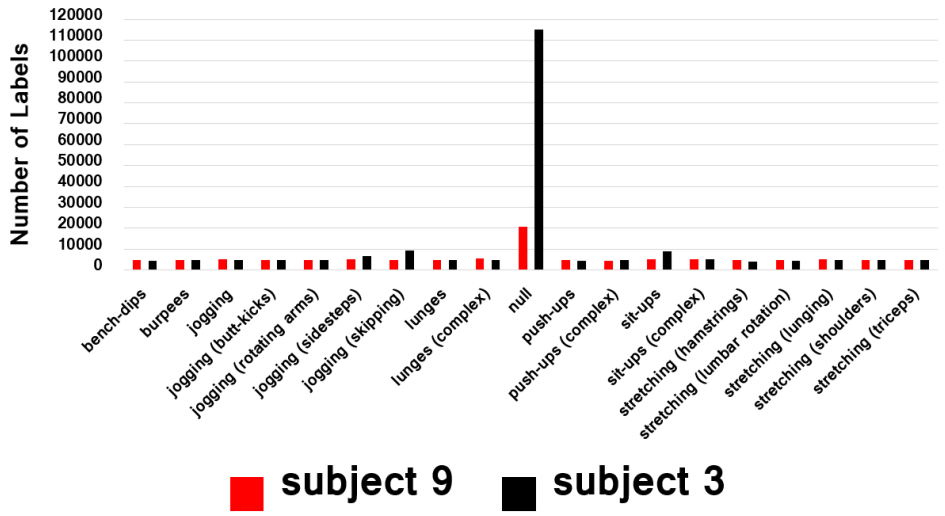


Figure 4.15: Label distribution for best and worst performing subject in WEAR

In diagram 4.15 the comparison of subject 3 and subject 9 from WEAR confirms the assumption of the Wetlab subject comparison. Here, the best performing subject has a much lower number of labels in the NULL class, compared to the subject with the worst result. This could indicate that highly unbalanced data in a single subject is beneficial in limiting label leakage. A closer look at the runs shows that, especially when the model is trained, the results vary drastically between the subjects. In Appendix B the $F1$ scores, which represent a metric that reflects the performance of an NN model, of the four subjects examined are shown. For WEAR, the $F1$ scores have a high variance B.3, but in Wetlab they are similar B.6. This contradicts the training stage as a possible reason for the large difference in LnAcc score.

4.2.8 Defenses

For the protection of the gradients in the FedSGD setting, the clipping and addition of noise to the original gradients were evaluated. Due to time constraints, the noise value is limited to 0.1. The clipping threshold is tested for the values 0.5, 1 and 1.5. An visualization of the different clipping stages and the results without differential privacy is provided for both datasets in the diagrams 4.16 and 4.17.

For the WEAR gradients, a clipping of 1.5 is already sufficient to push every attack on or below the score of the random baseline. The LnAcc of the LLBG attack drops below the baseline when clipping is set to 1. Using a clipping of 0.5 pushes the scores even further down, to 36% for LLBG and 34% for the rest of the attacks.

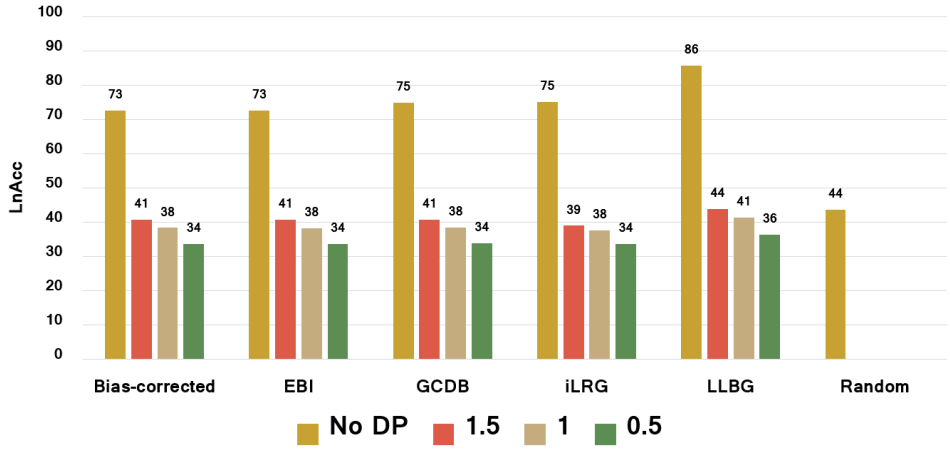


Figure 4.16: Differential Privacy through clipping and noise for WEAR. Noise is fixed to 0.1 and clipping is tested for 0.5, 1 and 1.5. The results show that a clipping of 1.5 is already sufficient to protect the WEAR gradients.

The gradients in the Wetlab evaluation show a higher resistance to different clipping values and noise addition. Even for a clipping value of 0.5 the LLBG and iLRG attack do not drop below the random guessing score of 45. Higher clipping with 1.5 and 1 do not push any of the attacks below the random guess. The jump in accuracy when applying 1.5 clipping still manages to achieve significantly better privacy than without DP because the LnAcc of all attacks drops by around 30%.

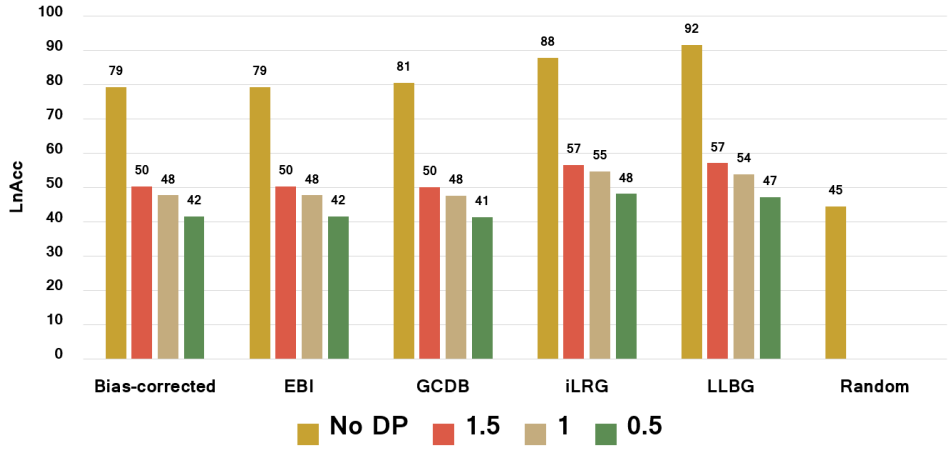


Figure 4.17: Differential Privacy through clipping and noise for Wetlab. Noise is fixed to 0.1 and clipping is tested for 0.5, 1 and 1.5. The results show that even a clipping of 0.5 is not sufficient to push the LnAcc of all attacks below the random baseline for the Wetlab gradients.

4.2.9 Preliminary Results

FedAVG Due to time constraints and the extensive evaluation for FedSGD the evaluation for FedAVG had to be cut short. Not the whole dataset is evaluated in the Local Updates setting, but only one entire training step is simulated. In the Multi User Update setting also just parts of the datasets are used, in order for the runs to finish early enough. For every user a different subject from the dataset is used. The table 4.3 shows the three combinations of subject. One consisting of the subjects with the most leakage, another with subjects that showed average leakage and the last with subjects that showed the least leakage.

	WEAR	Wetlab
Low	$14, 1, 7, 1*, 2$	$10, 15, 7, 4, 9$
Average	$11, 5, 6, 4, 12$	$1, 20, 0, 17, 16$
High	$17, 9, 0, 13, 16$	$6, 2, 18, 5, 19$

Table 4.3: Subject combinations for FedAVG Multi User Update evaluation. *In the Low combination of WEAR subject 1 is used twice. Low are the subjects with the least leakage. Average are the subjects with average leakage and High are the subjects with the most leakage in the FedSGD setting.

Local Updates The local updates configuration is evaluated for 2 and 5 local update steps. Additionally the noise and clipping evaluation is done in the same way as in the FedSGD setting. All runs are evaluated for batch size 100 and every dataset subject is iterated separately.

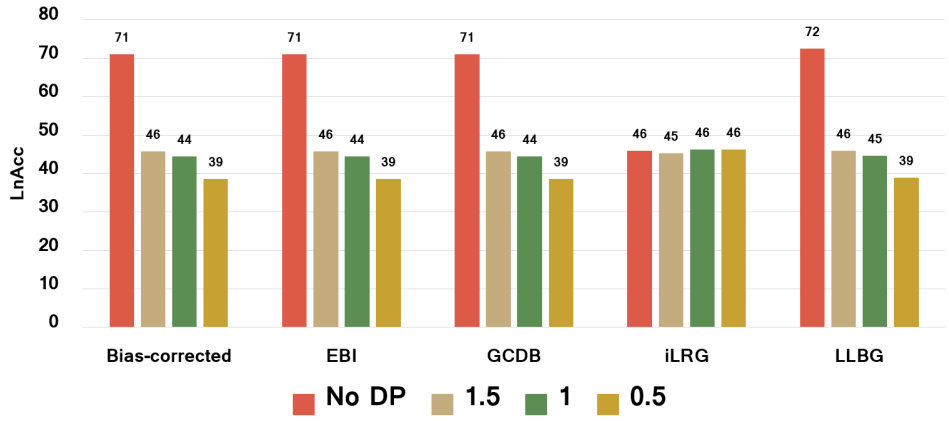
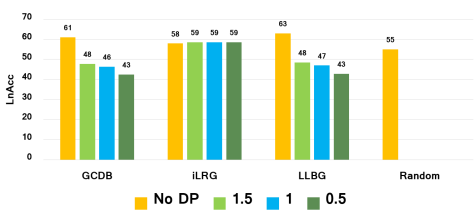


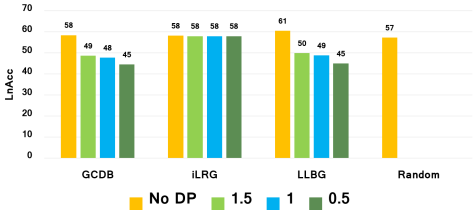
Figure 4.18: FedAVG with 2 local updates and 1 user. Averaged over all models, datasets, training stage and sampling strategies.

There was an error when evaluating another configuration setting with 5 local updates, thus we only focus on the setting with 2 local updates. For the bias attacks the LnAcc is above 70% and drops to random guessing for all clipping stages. The iLRG attack is performing on random guessing level for all DP stages. This is a first indication of problems with the attacks in the FedAVG setting and will be further investigated in the Multi User Update paragraph 4.2.9.

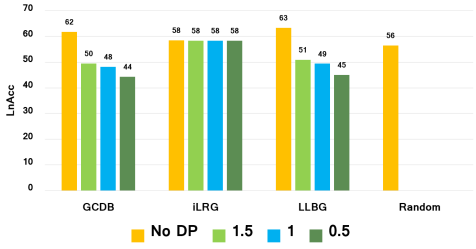
Multi User Update The Multi User Update configuration is evaluated for 2 and 5 local update steps and for 2 and 5 users. Furthermore, the noise and clipping evaluation is done in the same way as in the FedSGD setting. All runs are evaluated for batch size 100 and restricted to the bias gradient attacks, iLRG and the random baseline. The GCDB, EBI and BC showed the same result for all configurations, so in the diagrams only GCDB is displayed.



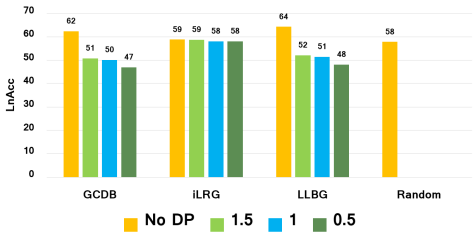
(a) FedAVG for 2 users and 2 local updates



(b) FedAVG for 2 users and 5 local updates



(c) FedAVG for 5 users and 2 local updates



(d) FedAVG for 5 users and 5 local updates

Figure 4.19: Results of different numbers of users and local updates. Averaged over all models, datasets, training stage and sampling strategies. Grouped by the clipping value and noise is fixed at 0.1.

Between the 4 different configurations, only minimal differences can be found, which indicates that the number of users and the number of local updates do not pose a significant influence. However, the results are above the random baseline only for GCDB and LLBG. The iLRG attack is only slightly above random guessing and also does not change when DP is applied. They show that some cases still deliver good results for balanced batches, but due to the high number of batches and the equal distribution of labels, this is basically random guessing. The other sampling methods also show results that indicate random guessing. After investigating the problem more closely, the two reasons for the poor performance could be identified. One is that the gradients are not directly shared but the parameter difference between the new version of the model and the initial model before the training step. The differential values are very small due to the low learning rate of 0.0001. The attacks appear to have problems with the low values, for example, with the estimation of the impact parameter in the bias attacks. A second problem is that there are too many data for a single user, the value was set to 10000 but not all data points were used in training. The label attacks still used the value 10000 for the label reconstruction, so this value needs to be decreased to the actual number of data points used in the training. In diagram 4.20 new results for the configuration with 5 users, 5 local updates are provided. Repeating all configurations was

not possible due to too little remaining time of the thesis. The results for GCDB and LLBG improved slightly compared to the performance of the same setting in diagram 4.19d while iLRG stayed the same. This indicates that more changes are necessary for the FedAVG setting compared to FedSGD. An option is to compare the performance of the user gradients before and after they are aggregated; more potential research will be suggested in the future work Section 5.2.

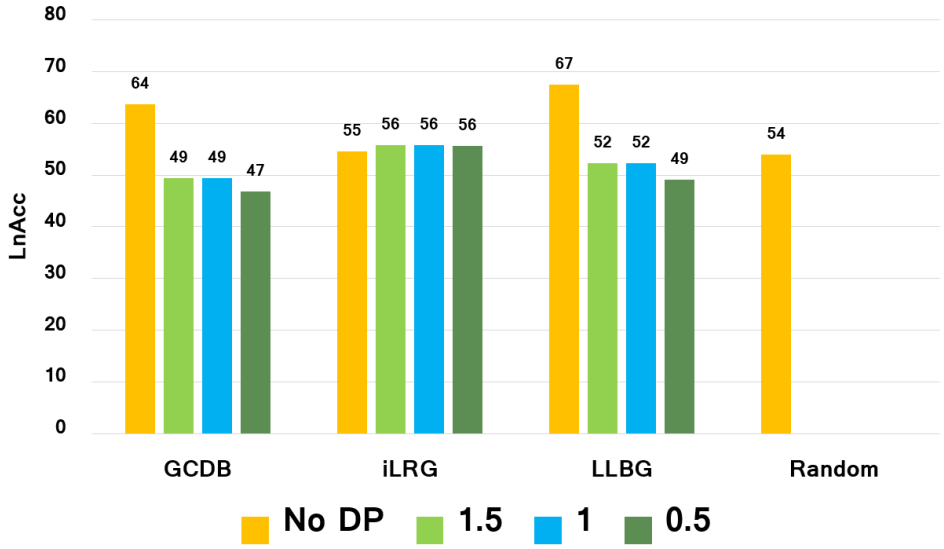


Figure 4.20: FedAVG with 5 local updates and 5 users. Averaged over all models, datasets, training stage and sampling strategies. This are the results after the learning rate and data distribution fix.

Detailed Results All results shown up until now have been averaged over multiple runs, mixing runs with different models, training stages, sampling or BS. This was a necessary step to keep the evaluation to a reasonable length. The downside to this is the artificial nature of the results, as they do not reflect the real values. More diagrams / results showing more detailed results can be found in the appendix B. A table B.1 with LnAcc and LeAcc is also provided for all runs of batch size 100.

4.3 Results

The evaluation concludes with a summary of the results previously shown and what can be deduced for the influence of the parameters model, training, dataset,

batch size, sampling and DP. For the model parameter, the comparison of DeepConvLSTM and TinyHAR did not show a higher leakage for the simpler TinyHAR architecture, but similar results. DeepConvLSTM even scored higher in general, especially for attacks that rely on weight gradients like LLG/LLG*. The training stage shows the same behavior as shown in previous works [41, 23, 13], but the trained stage performs much better for the Wetlab dataset compared to WEAR. This leads to the conclusion that the low number of classes in Wetlab leads to higher leakage even for well trained models.

For the three batch sizes 1, 10 and 100, batch size 1 shows mostly perfect results except for the weight gradient attacks iDLG, LLG and LLG*. It varies only slightly between datasets. For batch size 10 the highest scores are the same, but the best attacks for WEAR are BC and EBI while Wetlab has the best LnAcc score for LLBG. Comparing the datasets for size 100, Wetlab shows a consistently higher leakage among the top 5 attacks. This indicates that the batch size should not exceed the number of classes by a large amount. The results are heavily influenced by the sampling strategy, as a completely balanced batch for datasets with low number of classes automatically leads to better scores because of the higher chance of randomly picking the correct class label. Shuffled sampling is also dependent on the dataset it samples from, as it performs much better in Wetlab than for WEAR. The main reason is that Wetlab has a larger number of NULL class labels and fewer activities, leading to a more sequential style in shuffled batches and also a higher chance of randomly matching labels. Sequential sampling confirms this as Wetlab has a high score due to a lot of batches that mostly consist of the NULL class, while WEAR has more changing activities and thus more variation in sequential sampling leading to a lower score. The sequential style of a batch leads to an easier prediction for attacks that use the impact estimation approach because the selection does not have to jump from one class to another. If the attack has to change between classes a lot, the impact parameter has to have a high accuracy, otherwise it overlooks minor variations in the amount of labels across classes. Less variation is found for the usage of unbalanced sampling, where the scores remain close between the datasets.

This shows that the score is strongly reliant on three data aspects: number of classes, sampling strategy and distribution of the dataset itself. This characteristic is specific to HAR sensing datasets, since vision datasets have a higher number of classes and are more balanced between classes. This leads to the conclusion that the attack performances of label reconstruction papers like [41, 23, 13] are not comparable and cannot be assumed for the HAR sensing modality. Between the subjects of a dataset, differences in LnAcc up to 10% are found for averaged runs. The main differences can be found in the runs that used a trained model. The label distribution of the subjects indicates that the leakage is going down, the more

unbalanced the subject data is. This could be used as an additional privacy measure to prevent leakage as much as possible by collecting data in the worst way possible for a potential attacker.

Running the experiments with differential privacy in the form of noise addition and clipping showed that for clipping an order of 1.5 and for noise 0.1, is sufficient to protect gradients of the WEAR dataset. For Wetlab gradients iLRG and LLBG stayed just above the random baseline even for clipping of 0.5, but it is still an effective defense because all attacks dropped around 30% in accuracy. This actually is not a further validation for the assumption that a dataset with low class number has a higher leakage risk, but rather for the importance of the label distribution in the dataset. The Wetlab dataset almost always exhibits a high gradient for the NULL class, regardless of the sampling method, since this is by far the most frequently occurring label. In contrast, other classes show lower gradient magnitudes. The higher leakage in Wetlab could be because the clipping and noise values are not high enough to disguise the gradient effectively. In return, even higher clipping and/or noise would be necessary, which further degrades the accuracy of the model. Effectively, this means that the necessary amount of differential privacy in HAR sensing is connected to the data distribution of the underlying dataset.

Preliminary results for FedAVG are shown in Section 4.2.9, together with an explanation of the problems faced in this setting, an attempt to improve the performance and suggestions for possible further improvements.

Chapter 5

Conclusion

5.1 Summary

In this thesis the application of existing GI attacks together with a newly proposed attack GCDB in a HAR sensing setting was demonstrated. The implementation part includes the creation of a script that simulates the FedSGD algorithm in combination with the breaching repository. For the models, the DeepConvLSTM and TinyHAR implementation of the TAL repository was integrated. For evaluation, the two SOTA HAR models DeepConvLSTM and TinyHAR were tested, as they had not previously been used for any GI attack experiments. Likewise, the sensor data of the WEAR and Wetlab datasets were taken for evaluation in this setting for the first time. Other parameters that were considered in the evaluation are the training stage of the model, three batch sizes of the data 1, 10 and 100 as well as four different data sampling strategies: sequential, balanced, unbalanced and shuffled. This resulted in the number of run configurations seen in table 4.1. The findings for the FedSGD runs suggest that the results of label reconstruction attacks for HAR datasets are dependent on the three aspects:

1. Number of Classes
2. Sampling Strategy
3. Label Distribution in the Dataset

The newly proposed GCDB attack did perform slightly better than the other bias gradient attacks EBI and BC. This shows the validity of the approach for the impact estimation, but the LLBG attack showed the best and most stable results overall. All configurations are also evaluated, while differential privacy is applied to the gradients by clipping them and adding noise. The clipping values 1.5, 1.0

and 0.5 were tested while the noise was constantly fixed at 0.1. With differential privacy the LnAcc mostly drops below the random guess, showing that DP is an effective measure to prevent leakage in HAR sensing settings as well. Important to note is that for Wetlab a clipping of 0.5 was necessary to reach a similar LnAcc as WEAR reaches for 1.5 clipping. This strengthens the assumption that the label distribution of the dataset influences the results notably.

In the FedAVG setting, preliminary results are provided and a necessary scaling factor is identified, otherwise the bias attacks are close to random guessing. For the adjusted version, performance improved but still cannot reach the same leakage level as FedSGD. This setting needs further investigation and improvements. Some suggestions are provided in the following future work section, as well as proposals for potential research based on the findings of this work regarding the HAR sensing datasets.

5.2 Future Work

This thesis showed a first approach for attacking HAR sensing datasets with label reconstruction attacks. Due to the time constraints of the thesis, it was not possible to perform some experiments. In the following, an overview of possible future directions for the HAR sensing modality is provided.

Focusing on the FedAVG setting is a first logical step to further validate the presented results. It is especially important to show whether FedAVG can reach the same leakage amount as FedSGD or if it is possibly restricted due to the difference in gradients in FedAVG. Extending on this, a closer investigation of the gradients is necessary, as this thesis used a setting where the gradients of all users are aggregated and then provided to the server. The aggregation step could also be done by the server, in which case he would have access to the gradients of a single user. These are gradients that were computed over multiple local training steps, so basically it is similar to the local update steps scenario. This leads to another aspect of FedAVG which is the data distribution to the users, more scenarios for different parameters should be tested. For example, the amount of data each user possesses, how much of it is used in a training round, and the batch size. Additionally, the sampling method of every user was the same in this thesis. Using different sampling strategies for every user would definitely be interesting to evaluate. Changing hyperparameters for model training and dataset creation is also important, for example using another window size for the dataset.

All of the mentioned future work until now requires some sort of parameter adjustments and increases the number of evaluation results significantly, consequentially the logging should be extended and the changes to parameters need to be eas-

ier. For this purpose, more scripts should be written that handle all this and make the entire evaluation process simpler and faster. Switching gears to parameters on a different scale, other datasets and models could be evaluated as well. In the TAL repository, the Attend and Discriminate [2] model is already integrated, as well as the datasets: Opportunity [34], SBHAR [33], Hang-Time [16] and RWHAR [38], they could be easily integrated and evaluated. This would be especially interesting regarding the identified influences posed by the datasets used in HAR sensing: the number of classes and the label distribution. If analyses of these datasets can confirm these influences, this would help to consolidate the results of this work and also mean that differential privacy needs to be considered for every dataset individually, as shown for WEAR and Wetlab in this thesis.

Some attacks were not integrated due to time constraints, they should be tested in a next step. For example, the variations of the LLG [41] and LLBG [13]. The GDBR attack [51], that was released at the runtime of this thesis, also showed promising results because it could recover 80% of the labels in various settings/ sampling strategies. Also, it does not need to use gradients of the classification layer of a NN-model, which makes it distinct from the other attacks in this thesis. Another important step in label reconstruction is to extend the attacks to trained models to ensure a similar performance of the attack regardless of the model's training stage.

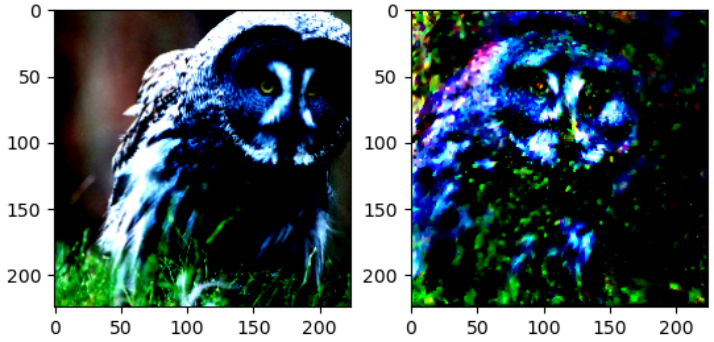
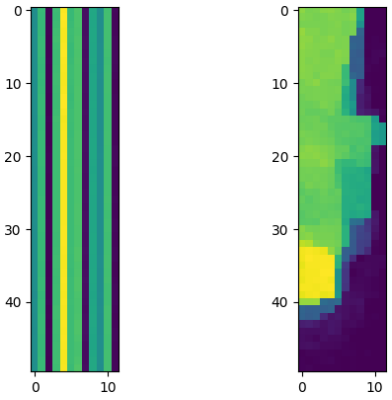


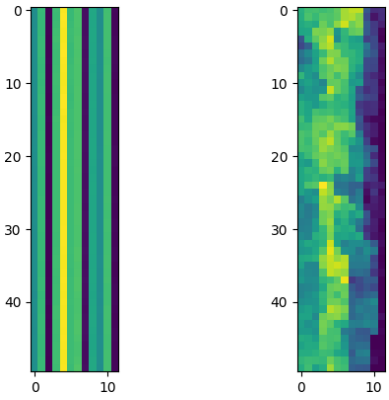
Figure 5.1: The figure shows an example for a image reconstruction, found in the breaching repository. The original image is on the left side and the reconstructed image on the right side.

In the vision community, a lot of focus has been on reconstructing the original data, usually an image. An example of such a reconstruction is shown in the images 5.1. For this example, the reconstruction worked well and the information leakage can be identified with one look at the reconstructed image. At the beginning of this

thesis the reconstruction was adapted for the sensor data from WEAR and Wetlab. Two examples of such reconstructions for a single $50x12$ sample are shown in the images 5.2. The order of the sensor data is completely lost for both examples and there is no obvious information to gain like in the image example. Optimizing this reconstruction could be another point to focus on. For accelerometer data, the information gain from a perfect reconstruction of a single still would be limited, but if it could be achieved constantly, the whole subject activity of WEAR and Wetlab would be recovered. Applying this to other types of sensors, this would enable more serious privacy breaches such as health characteristics for heart rate sensors or routines in daily life for smart home sensors. Datasets that could simulate such a setting are MIMIC-IV [18] and CASAS [9].



(a) Example 1: Reconstruction of a single $50x12$ sample.



(b) Example 2: Reconstruction of a single $50x12$ sample.

Figure 5.2: Examples of reconstructions in HAR sensing. Original sensor data on the left side and reconstructed data on the right side. Samples are from the WEAR dataset.

The complete integration of the breaching repository into the TAL repository would enable simulation of the label reconstruction over a whole training phase. It could help to identify at which point the label reconstruction performance drops to the level of the trained stage performance shown in this thesis. Furthermore, the other architectures of TAL could be used, which focus on the temporal action localization aspect of HAR instead of inertial activity recognition. For this modality, other attacks would be necessary because the output is a segment prediction in the form of $[N(activity, start, end) - triplets]$. Additionally, differential privacy should be added or a lightweight defense like the removal of the bias from the classification layer could be tested [36]. DP could be implemented using an existing

module, for example Opacus[1]. At the start of this thesis the Opacus integration was tested and easily integrated into the TAL repository. Combining TAL, breaching and Opacus would be sufficient to create a complete framework for training, attacking and defending in HAR sensing.

[1] https://opacus.ai/

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Alireza Abedin, Mahsa Ehsanpour, Qinfeng Shi, Hamid Rezatofighi, and Damith C. Ranasinghe. Attend and discriminate: Beyond the state-of-the-art for human activity recognition using wearable sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1), March 2021.

[3] Ulf Blanke, Bernt Schiele, Matthias Kreil, Paul Lukowicz, Bernhard Sick, and Thiemo Gruber. All for one or one for all? combining heterogeneous features for activity spotting. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 18–24, 2010.

[4] Marius Bock, Hilde Kuehne, Kristof Van Laerhoven, and Michael Moeller. WEAR: An Outdoor Sports Dataset for Wearable and Egocentric Activity Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(4):1–21, November 2024.

[5] Marius Bock, Michael Moeller, and Kristof Van Laerhoven. Temporal Action Localization for Inertial-based Human Activity Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(4):1–19, November 2024.

[6] Marius Bock, Kristof Van Laerhoven, and Michael Moeller. Weak-annotation of har datasets using vision foundation models. In *Proceedings of the 2024 ACM International Symposium on Wearable Computers*, ISWC '24, page 55–62, New York, NY, USA, 2024. Association for Computing Machinery.

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.*, 46(3), January 2014.

[9] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan C. Krishnan. CASAS: A Smart Home in a Box. *Computer*, 46(7):62–69, July 2013.

[10] Trung Dang, Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, Peter Chin, and Françoise Beaufays. Revealing and protecting labels in distributed training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1727–1738. Curran Associates, Inc., 2021.

[11] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[12] David Enthoven and Zaid Al-Ars. Fidel: Reconstructing private training samples from weight updates in federated learning. In *2022 9th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, page 1–8. IEEE, November 2022.

[13] Nadav Gat and Mahmood Sharif. Harmful Bias: A General Label-Leakage Attack on Federated Learning from Bias Gradients. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*, pages 31–41, Salt Lake City UT USA, November 2024. ACM.

[14] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients – How easy is it to break privacy in federated learning?, 2020. Version Number: 2.

[15] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. Deep learning with label differential privacy. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc.

[16] Alexander Hoelzemann, Julia Lee Romero, Marius Bock, Kristof Van Laerhoven, and Qin Lv. Hang-time har: A benchmark dataset for basketball activity recognition using wrist-worn inertial sensors. *Sensors*, 23(13), 2023.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[18] Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1, January 2023.

[19] Mohan Li, Martin Gjoreski, Pietro Barbiero, Gašper Slapničar, Mitja Luštrek, Nicholas D. Lane, and Marc Langheinrich. A survey on federated learning in human sensing, 2025.

[20] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1–3):503–528, August 1989.

[21] Songfeng Liu, Jinyan Wang, Wenliang Zhang, Guangxi Key Lab of Multisource Information Mining and Security, Guangxi Normal University, Guilin, China, and College of Computer Science and Engineering, Guangxi Normal University, Guilin, China. Federated personalized random forest for human activity recognition. *Mathematical Biosciences and Engineering*, 19(1):953–971, 2021.

[22] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3615–3634, July 2024.

[23] Kailang Ma, Yu Sun, Jian Cui, Dawei Li, Zhenyu Guan, and Jianwei Liu. Instance-wise batch label restoration via gradients in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the*

20th International Conference on Artificial Intelligence and Statistics, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[25] Shagufta Mehnaz, Sayanton V. Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models, 2022.

[26] Francisco Ordóñez and Daniel Roggen. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, 16(1):115, January 2016.

[27] Ke Pan, Yew-Soon Ong, Maoguo Gong, Hui Li, A.K. Qin, and Yuan Gao. Differential privacy in deep learning: A literature survey. *Neurocomputing*, 589:127663, July 2024.

[28] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning: Revisited and enhanced. In *Applications and Techniques in Information Security: 8th International Conference, ATIS 2017, Auckland, New Zealand, July 6–7, 2017, Proceedings*, pages 100–110. Springer, 2017.

[29] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

[30] Le Trieu Phong and Tran Thi Phuong. Differentially private stochastic gradient descent via compression and memorization. *Journal of Systems Architecture*, 135:102819, 2023.

[31] Riccardo Presotto, Gabriele Civitarese, and Claudio Bettini. Preliminary Results on Sensitive Data Leakage in Federated Human Activity Recognition. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 304–309, Pisa, Italy, March 2022. IEEE.

[32] Jia Qian, Hiba Nassar, and Lars Kai Hansen. Minimal model structure analysis for input reconstruction in federated learning, 2020.

[33] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[34] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and José del R. Millàn. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, pages 233–240, 2010.

[35] Debaditya Roy, Ahmed Lekssays, Sarunas Girdzijauskas, Barbara Carminati, and Elena Ferrari. Private, Fair and Secure Collaborative Learning Framework for Human Activity Recognition. In *Adjunct Proceedings of the 2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing & the 2023 ACM International Symposium on Wearable Computing*, pages 352–358, Cancun, Quintana Roo Mexico, October 2023. ACM.

[36] Daniel Scheliga, Patrick Mäder, and Marco Seeland. Combining stochastic defenses to resist gradient inversion: An ablation study, 2022.

[37] Philipp M. Scholl, Matthias Wille, and Kristof Van Laerhoven. Wearables in the wet lab: a laboratory system for capturing and guiding experiments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, page 589–599, New York, NY, USA, 2015. Association for Computing Machinery.

[38] Timo Sztyler and Heiner Stuckenschmidt. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9, 2016.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. arXiv:1706.03762 [cs].

[40] Aidmar Wainakh, Till Müßig, Tim Grube, and Max Mühlhäuser. Label leakage from gradients in distributed machine learning. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, page 1–4. IEEE Press, 2021.

[41] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User-Level Label Leakage from Gradients in Federated Learning, January 2022. arXiv:2105.09369 [cs].

[42] Xiaodong Wang, Longfei Wu, and Zhitao Guan. Graddiff: Gradient-based membership inference attacks against federated distillation with differential comparison. *Information Sciences*, 658:120068, 2024.

[43] Zhibo Wang, Zhiwei Chang, Jiahui Hu, Xiaoyi Pang, Jiacheng Du, Yongle Chen, and Kui Ren. Breaking Secure Aggregation: Label Leakage from Aggregated Gradients in Federated Learning, June 2024. arXiv:2406.15731 [cs].

[44] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating gradient leakage attacks in federated learning, 2020.

[45] Zhiwen Xiao, Xin Xu, Huanlai Xing, Fuhong Song, Xinhan Wang, and Bowen Zhao. A federated learning system with enhanced feature extraction for human activity recognition. *Knowledge-Based Systems*, 229:107338, October 2021.

[46] Xianghua Xie, Chen Hu, Hanchi Ren, and Jingjing Deng. A survey on vulnerability of federated learning: A learning algorithm perspective. *Neurocomputing*, 573:127225, March 2024.

[47] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), January 2019.

[48] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through Gradients: Image Batch Recovery via GradInversion, April 2021. arXiv:2104.07586 [cs].

[49] Hongzheng Yu, Zekai Chen, Xiao Zhang, Xu Chen, Fuzhen Zhuang, Hui Xiong, and Xiuzhen Cheng. FedHAR: Semi-Supervised Online Learning for Personalized Federated Human Activity Recognition. *IEEE Transactions on Mobile Computing*, 22(6):3318–3332, June 2023.

[50] 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.

[51] Rui Zhang, Ka-Ho Chow, and Ping Li. Building gradient bridges: Label leakage from restricted gradient sharing in federated learning, 2024.

[52] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved Deep Leakage from Gradients, January 2020. arXiv:2001.02610 [cs].

[53] Ying Zhao and Jinjun Chen. A survey on differential privacy for unstructured data content. *ACM Computing Surveys*, 54, 01 2022.

[54] Yexu Zhou, Haibin Zhao, Yiran Huang, Till Riedel, Michael Hefenbrock, and Michael Beigl. TinyHAR: A Lightweight Deep Learning Model Designed for Human Activity Recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 89–93, Cambridge United Kingdom, September 2022. ACM.

[55] Junyi Zhu and Matthew Blaschko. R-gap: Recursive gradient attack on privacy, 2020.

[56] Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients, December 2019. arXiv:1906.08935 [cs].

Appendix A

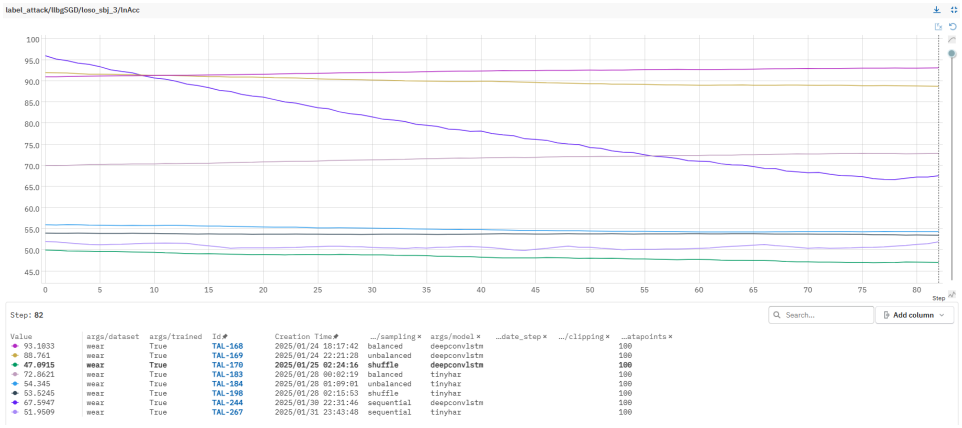
Program Code / Resources

The source code, a documentation, some usage examples, and additional test results are available at https://github.com/maxiH137/master-thesis The state of the repository as of handing in this thesis, a PDF version of this thesis and data files of all experiments/runs mentioned in the evaluation of this thesis are contained on the micro SD card that was handed in with this thesis.

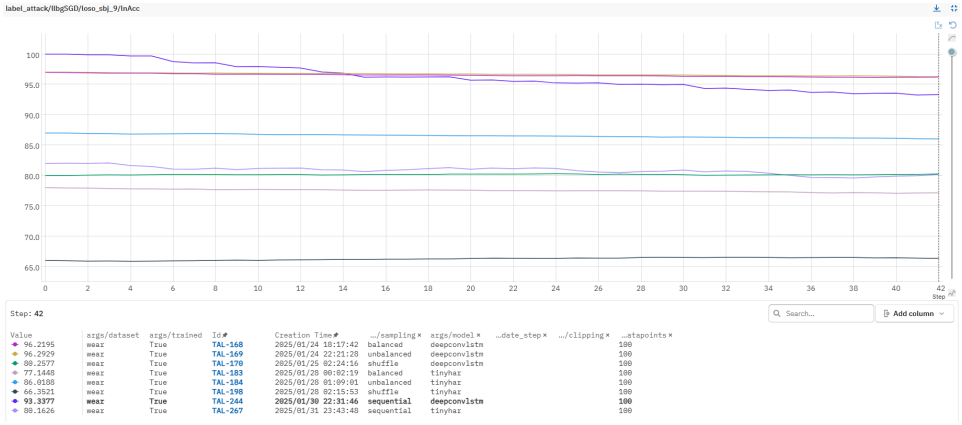
Appendix B

Further Experimental Results

In the following, further experimental results are provided such as more detailed results for different configurations of the parameters: model, training, dataset, sampling. Furthermore, a comparison of the runs of the subjects that showed the most and least leakage in each dataset is provided. Confusion matrices of different attacks for some settings are also provided.

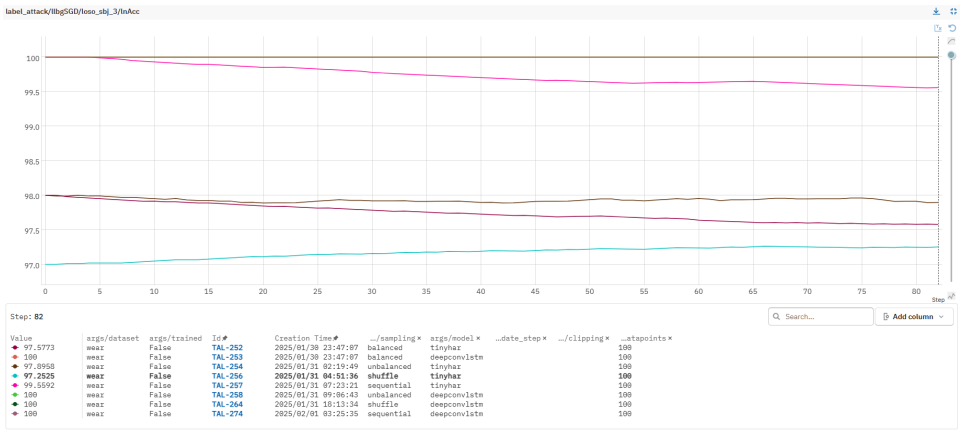


(a) WEAR subject 3

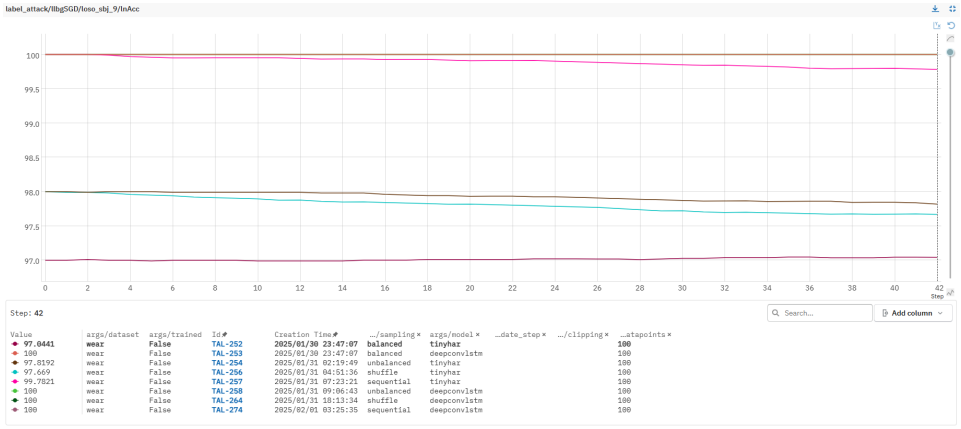


(b) WEAR subject 9

Figure B.1: LnAcc Comparison for WEAR Subject 3 and 9. Every line is one of the 16 different run configuration for model, dataset, training = True, sampling and batch size 100.

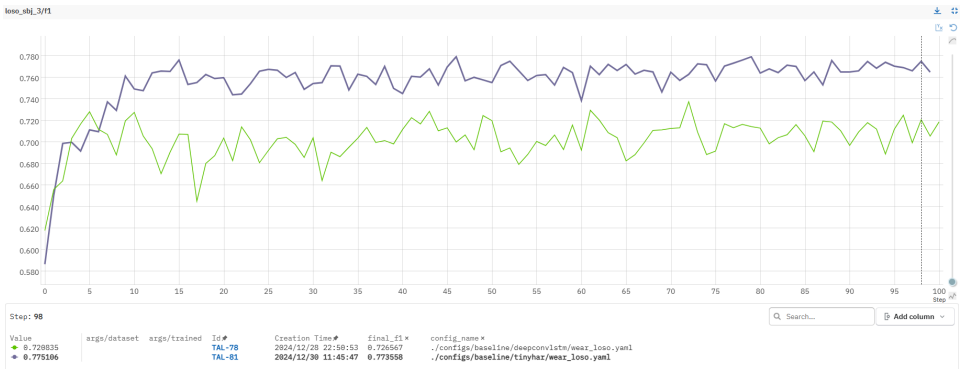


(a) WEAR subject 3

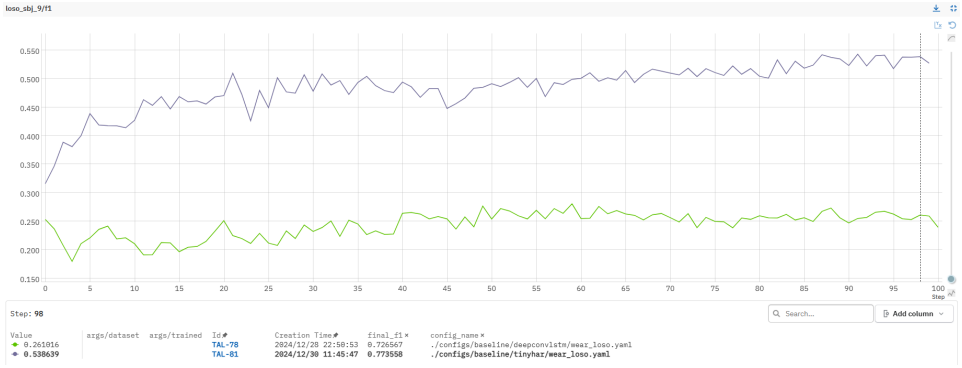


(b) WEAR subject 9

Figure B.2: LnAcc Comparison for WEAR Subject 3 and 9. Every line is one of the 16 different run configuration for model, dataset, training = False, sampling and batch size 100.

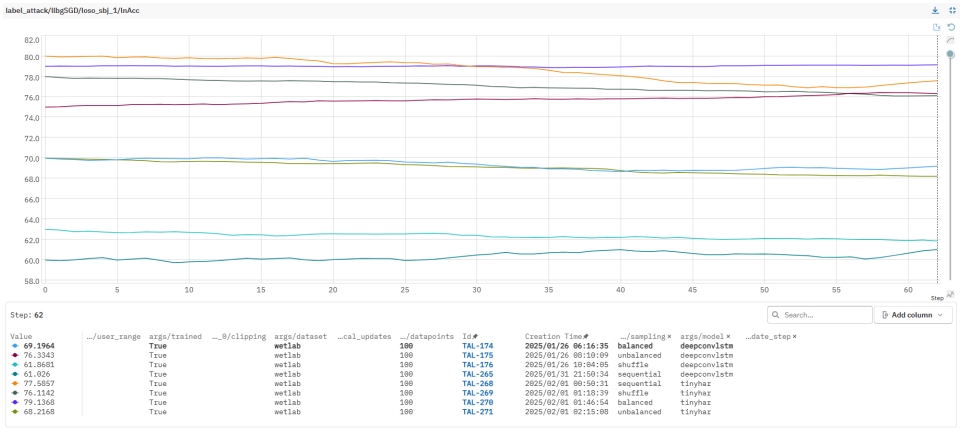


(a) WEAR subject 3

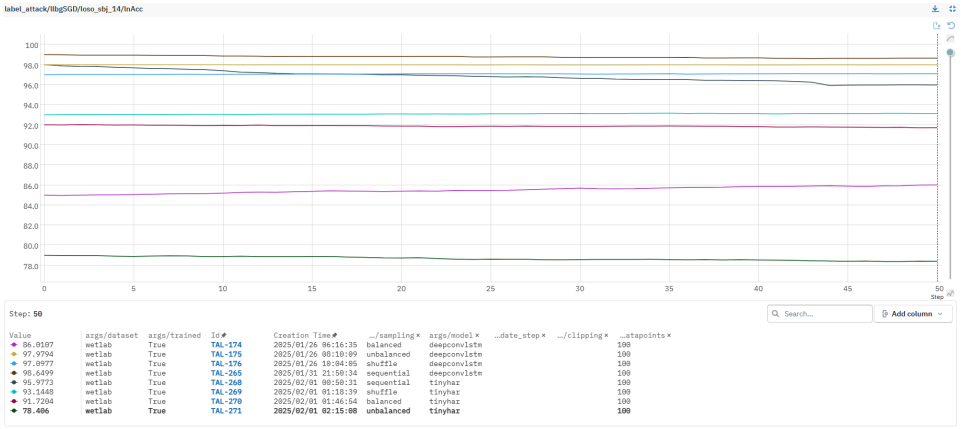


(b) WEAR subject 9

Figure B.3: IN (a) the F1 score for WEAR subject 3 is plotted and in (b) for subject 9. The green line is the DeepConvLSTM plot and the blue line is from TinyHAR.

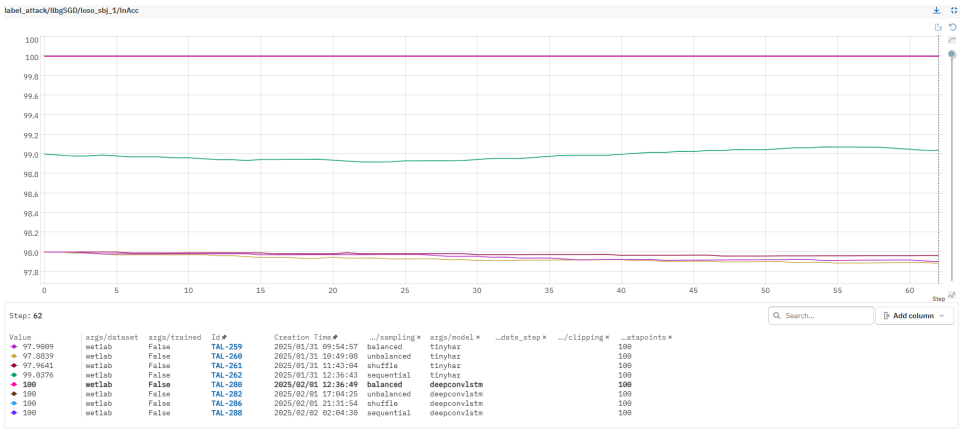


(a) Wetlab subject 1

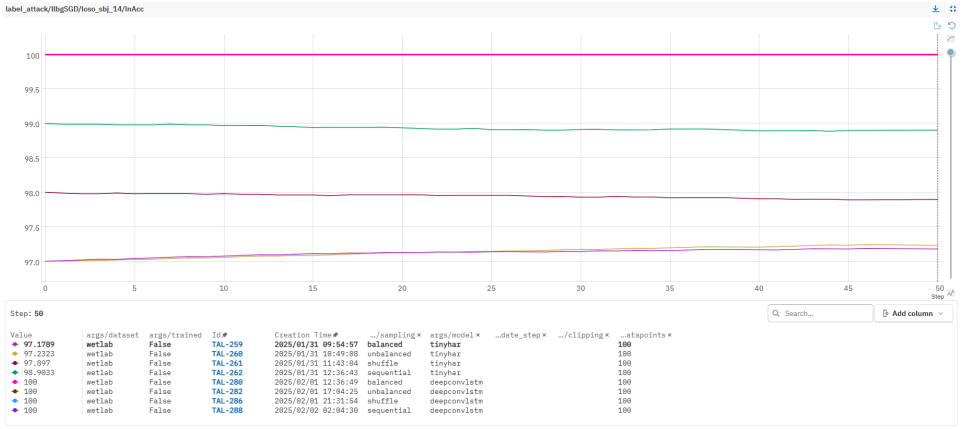


(b) Wetlab subject 14

Figure B.4: LnAcc Comparison for Wetlab Subject 1 and 14. Every line is one of the 16 different run configuration for model, dataset, training = True, sampling and batch size 100.

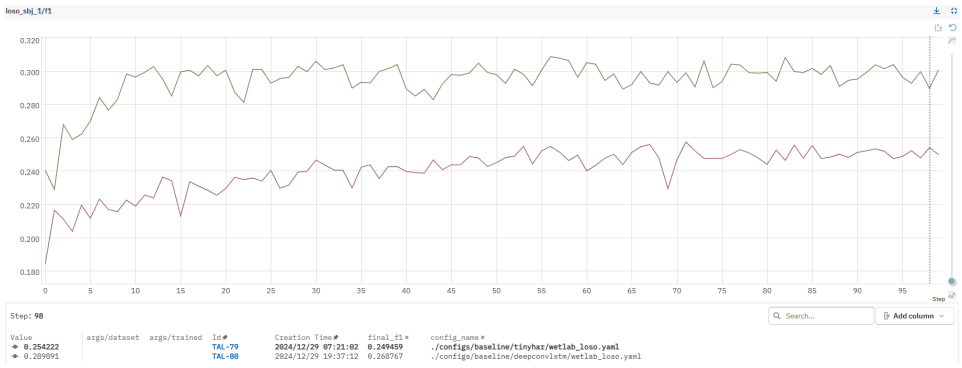


(a) Wetlab subject 1

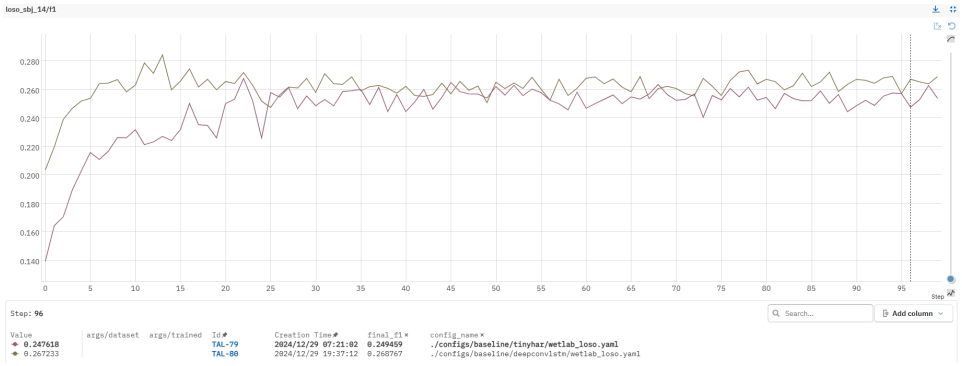


(b) Wetlab subject 14

Figure B.5: LnAcc Comparison for Wetlab Subject 1 and 14. Every line is one of the 16 different run configuration for model, dataset, training = False, sampling and batch size 100.

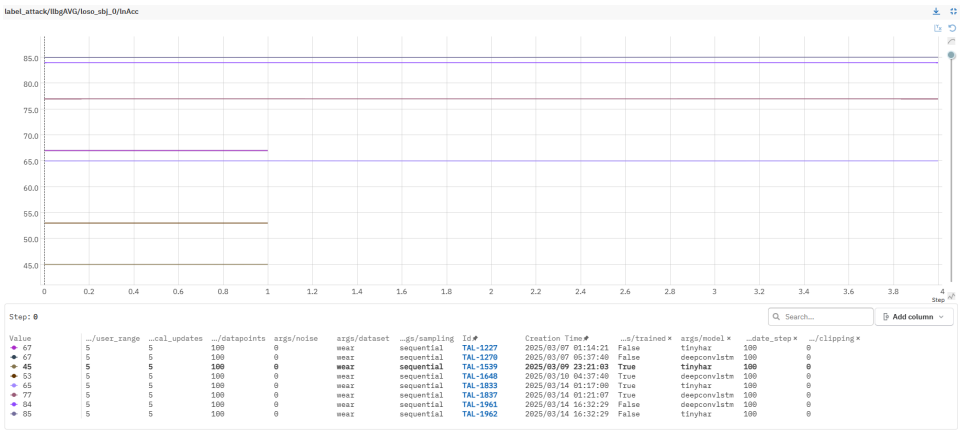


(a) Wetlab subject 1

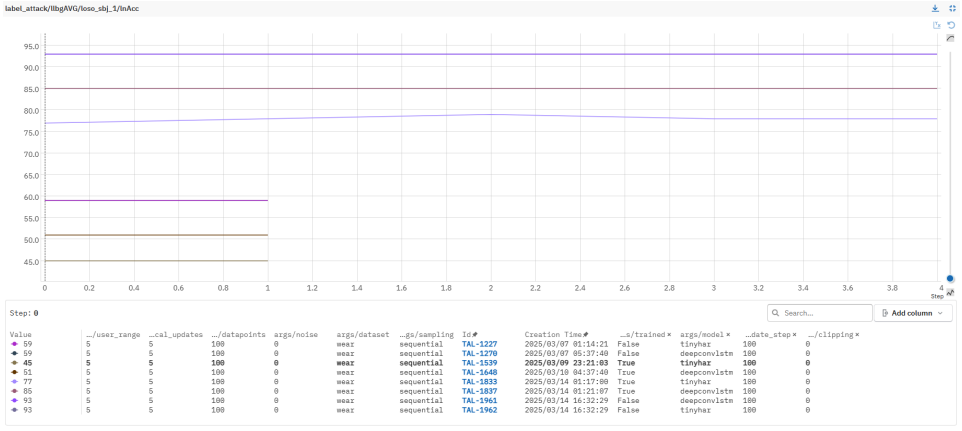


(b) Wetlab subject 14

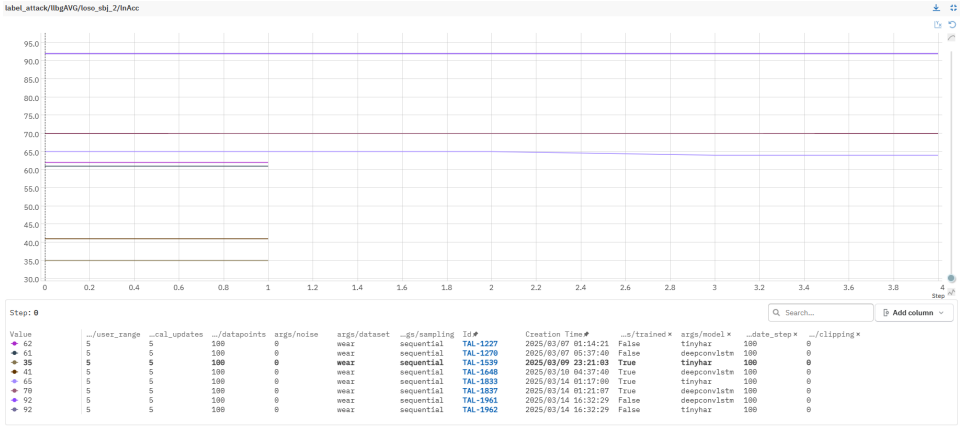
Figure B.6: In (a) the F1 score for Wetlab subject 1 is plotted and in (b) for subject 14. The green line is the DeepConvLSTM plot and the blue line is from TinyHAR.



(a) Low

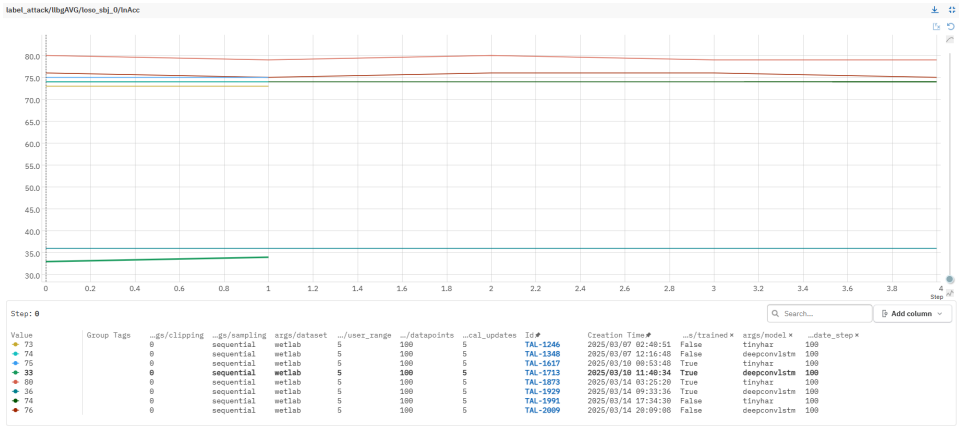


(b) Average



(c) High

Figure B.7: Comparison of WEAR runs before and after the correction attempt for the Multi User Setting. Short lines are the old runs and longer lines are the new runs. Sequential sampling improved for drastically for all three settings. Batch size is 100.



(a) Low

(b) Average

(c) High

Figure B.8: Comparison of Wetlab runs before and after the correction attempt for the Multi User Setting. Short lines are the old runs and longer lines are the new runs. Sequential sampling improved for drastically for all three settings. Batch size is 100.

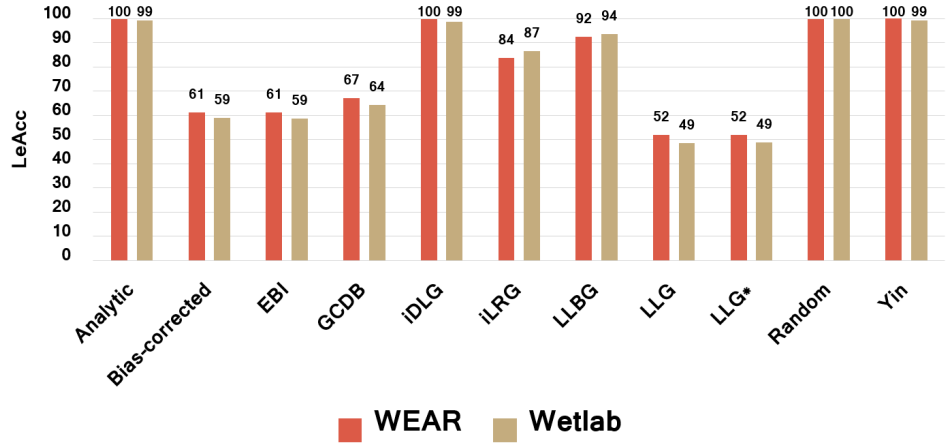


Figure B.9: The figure shows the LeAcc for WEAR and Wetlab averaged over all runs with batch size 100.

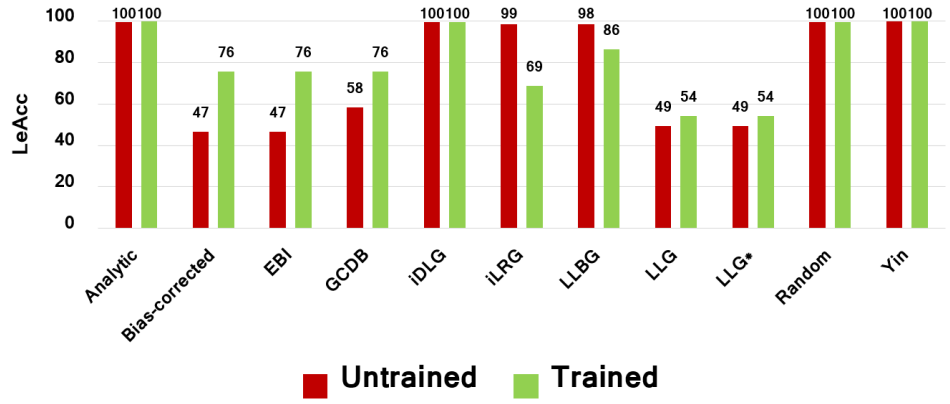


Figure B.10: The figure shows the LeAcc for WEAR runs grouped by training stage averaged over all runs with batch size 100.

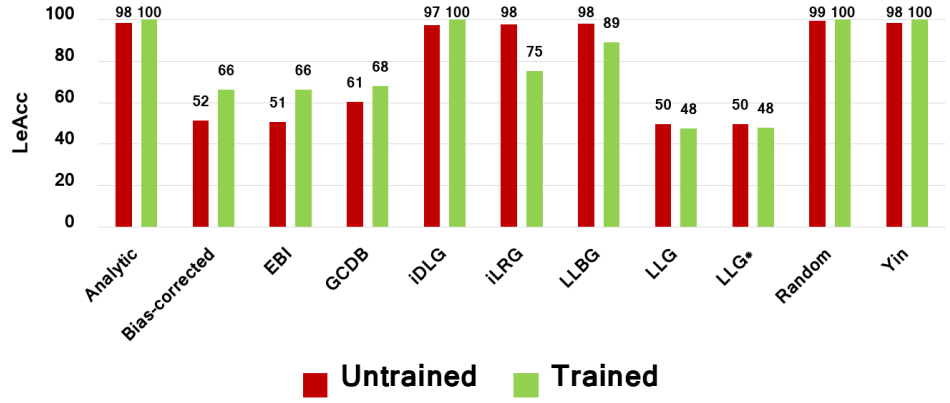


Figure B.11: The figure shows the LeAcc for Wetlab runs grouped by training stage averaged over all runs with batch size 100.

Combination	Attack	LnAcc	LeAcc
(D, Untrained, 100, WEAR, balanced)	LLBG	100.00	100.00
(D, Untrained, 100, WEAR, unbalanced)	LLBG	100.00	100.00
(D, Untrained, 100, WEAR, sequential)	LLBG	100.00	100.00
(D, Untrained, 100, WEAR, shuffle)	LLBG	100.00	100.00
(D, Trained, 100, WEAR, balanced)	LLBG	93.26	93.78
(D, Trained, 100, WEAR, unbalanced)	LLBG	90.46	89.62
(D, Trained, 100, WEAR, sequential)	LLBG	66.10	69.54
(D, Trained, 100, WEAR, shuffle)	LLBG	65.77	93.71
(D, Untrained, 100, Wetlab, balanced)	LLBG	100.00	100.00
(D, Untrained, 100, Wetlab, unbalanced)	LLBG	100.00	100.00
(D, Untrained, 100, Wetlab, sequential)	LLBG	100.00	100.00
(D, Untrained, 100, Wetlab, shuffle)	LLBG	100.00	100.00
(D, Trained, 100, Wetlab, balanced)	LLBG	78.81	84.45
(D, Trained, 100, Wetlab, unbalanced)	LLBG	88.21	82.38
(D, Trained, 100, Wetlab, sequential)	LLBG	87.01	91.53
(D, Trained, 100, Wetlab, shuffle)	LLBG	85.01	82.45
(T, Untrained, 100, WEAR, balanced)	LLBG	97.46	99.65
(T, Untrained, 100, WEAR, unbalanced)	LLBG	97.67	89.86
(T, Untrained, 100, WEAR, sequential)	LLBG	99.28	99.54
(T, Untrained, 100, WEAR, shuffle)	LLBG	97.50	98.33
(T, Trained, 100, WEAR, balanced)	LLBG	76.27	88.18
(T, Trained, 100, WEAR, unbalanced)	LLBG	63.41	80.38
(T, Trained, 100, WEAR, sequential)	LLBG	64.47	85.36
(T, Trained, 100, WEAR, shuffle)	LLBG	59.97	90.32
(T, Untrained, 100, Wetlab, balanced)	LLBG	97.47	99.99
(T, Untrained, 100, Wetlab, unbalanced)	LLBG	97.85	93.19
(T, Untrained, 100, Wetlab, sequential)	LLBG	98.95	99.790
(T, Untrained, 100, Wetlab, shuffle)	LLBG	97.84	92.35
(T, Trained, 100, Wetlab, balanced)	LLBG	86.19	97.26
(T, Trained, 100, Wetlab, unbalanced)	LLBG	76.48	93.88
(T, Trained, 100, Wetlab, sequential)	LLBG	86.78	96.56
(T, Trained, 100, Wetlab, shuffle)	LLBG	86.25	84.05

Table B.1: All runs for LLBG in setting FedSGD for batch size 100. LnAcc and LeAcc are averaged along all the subjects of the respective dataset. Values are rounded to two decimal places. In the combination T stands for TinyHAR and D for DeepConvLSTM.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Masterarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Siegen, den 17.03.2025 Unterschrift