Membership Inference Attacks on Federated Learning Optimisation Protocols

PIM MULDER, University of Twente, The Netherlands

Previous works on Membership Inference Attacks (MIAs) have done plentiful research on the effect of different hyperparameters of a Federated Learning (FL) system on the susceptibility to MIAs [2, 8, 11, 23, 31–33, 35, 36]. In this work, we go one meta-step higher to investigate the effect the optimisation method itself has on the MIA accuracy and AUC. We apply white-box Membership Inference Attacks (MIAs) on these protocols to show that FedAdam [27], FedNAG [34], and a version of FedNL [29] are more susceptible to MIAs than FedAvg [22] because of an increased generalisation error, despite an overall lower empirical error. We find this is because alternative protocols have more overconfidence than FedAvg, which results in clearer distinctions in membership. Furthermore, we tailor a new attack to these protocols called the Ancillary Attack. This attack relies on the ancillary variables that the alternative protocols use. The update to the ancillary variables shows different patterns for member than for non-member data and so can be used to improve an MIA. We modify the attack of Nasr et al. [23] by adding a component that takes this update as input. Though this component further decreases the stability of the attack model, we show that it helps to improve the prediction of membership. We run experiments with and without this component on the victim model architectures of ResNet [12] with 18 and 34 layers, smaller Fully Connected Networks (FCNs) and on logistic regression trained for CIFAR-10, CIFAR-100 [17], Purchase100 and Texas100 [31] in a cross-silo federated setting [14]. We show that even with a federation of 100 clients, a client can successfully attack systems optimised with alternatives to FedAvg. Additionally, when using the ancillary attack component, they can further their success.

Additional Key Words and Phrases: Membership Inference Attacks, Federated Learning, Federated Optimisation, Ancillary Attack

1 INTRODUCTION

Since 2023, Machine Learning (ML) has been going through many innovations that have made the technology significantly more accessible to the public. With the increased use of this technology, the problem of the collection of sensitive data for the creation of intelligent models has slowly been surfacing. Federated Learning (FL) [22] seems to provide a solution to prevent the collection of sensitive data by requiring clients to train an ML model locally to then aggregate the local models at a server to train a collaborative model in a series of rounds.

Although minimal, the clients still transmit some information about their training data. For that reason, a common security audit [5] is to verify if an attacker cannot succeed in a Membership Inference Attack (MIA) [31]. The purpose of an MIA is to infer from a given model if a victim sample has been included in the training data. For this, another ML model is trained to find the overconfidence the victim model has in members compared to non-members and to find other possible overfitting patterns.

In recent years, much work has been done to find the effect batch size [2, 8], model size [8, 32, 36], amount of epochs [2, 8, 11], learning rate [2] and regularisation [23, 31, 33, 35, 36] has on the accuracy of membership prediction. From this research, it has become apparent that, in general, the more a model is able to generalise, the more resilient it is to MIAs. However, current work has not investigated the effect the optimisation protocol itself has on the accuracy of membership prediction.

As Aggarwal [1] states, alternatives to standard gradient-descent [28] perform well in ill-conditioned optimisation problems. Since the local optimisation problems of the clients within a federation are inhomogeneous, the alternatives are an attractive strategy. We investigate if using these alternatives carries or mitigates any additional privacy risk that comes from FL. The alternatives that we consider are the FedAdam [27], FedNAG [34] and a simplified FedNL [29] protocol as well as standard FedAvg [22].

The FedNAG and FedNL protocols require the clients to transmit more information than just the updated model parameters to improve their optimisation technique. We examine if these ancillary variables can be used in an attack and whether the increase in model prediction accuracy weighs up to the increase in leakage. For this, we developed an Ancillary Attack that leverages this information. We examine the chance of success for an attacking server and client in four cross-silo FL applications. In FL, a federation either consists of many clients with few data or few clients with much data [14]. These are called cross-device and cross-silo applications, and we investigate the latter.

We run the Ancillary Attack on models trained on the CIFAR-10 and 100 [17], Purchase100 [31] and Texas100 [31] datasets. For CIFAR-10 and 100, we attack a ResNet [12] of 18 and 34 layers, and for Purchase100 and Texas100, we attack a simple Fully Connected Network (FCN) and a logistic regression model. We train each architecture with FedAdam, FedAvg, and FedNAG. We also train logistic regression using FedNL.

With our experiments, we show that these learning methods increase the overall performance of a federated model, but with that, the susceptibility to MIAs increases due to an increase in the generalisation error. Furthermore, we show that ancillary variables can be used to increase the MIA performance both for an average increase and an increase in performance when targeting clients that are outliers to the distribution, and so are more vulnerable. Finally, we show that in a small cross-device setting, these attacks are even possible by a federation client.

2 BACKGROUND

2.1 Federated Learning Protocols

With Federated Learning (FL), a federation of K clients collectively tries to fit a mapping function f parameterised by θ between the input data $\mathbf{x}$ and target outputs $\mathbf{y}$ under the orchestration of a server. This target mapping is what is used to best predict a result; hence, we speak of optimising the parameters that lead to good predictions. In each global round, the server selects a fraction C of the clients to perform a local optimisation round. These clients form that round's

Author's address: Pim Mulder, p.w.mulder@student.utwente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

participation set S_i. To these clients, the server transmits the model θ_i, which the clients use as initial parameters.

During a local optimisation round, the selected clients $k \in S_i$ use their local dataset $\mathcal{D}_k = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{m}$ to minimimise a loss function $\ell(f_\theta(\mathbf{x}), \mathbf{y})$. They do so until the local convergence criteria are fulfilled, typically after passing some amount of local epochs E. The client transmits their updated model $\theta_{i+1}^{(k)}$, and their amount of data $n_k = |\mathcal{D}_k|$ to the server. The server takes the weighted average of the received models w.r.t. the size of the dataset corresponding to the received parameters to obtain the new aggregate θ_{i+1}. This is repeated until some global convergence criteria are fulfilled, e.g., after some amount of global rounds have passed.

Depending on the optimisation protocol, the server and client may vary in the information transmitted. In our experiments, we consider the following protocols that may be encountered in a cross-silo FL setting [14].

2.1.1 *Federated Averaging (FedAvg) [22].*

Designed by McMahan et al. [22]. To minimise the loss function, the selected clients apply the stochastic gradient descent (SGD) [28] algorithm with a learning rate η and transmit their updated model to the server:

$$\theta_{i+1}^{(k)} = \theta_i - \eta \sum_{(\mathbf{x},\mathbf{y}) \in B} \nabla_{\theta_i} \ell(f_{\theta_i}(\mathbf{x}), \mathbf{y})$$

Where B is a batch of samples from $\mathcal{D}_k$. The server combines the results as follows and returns the updated model:

$$\theta_{i+1} = \sum_{k \in S_i} \frac{n_k}{\sum_{k \in S_i} n_k} \theta_{i+1}^{(k)}$$

2.1.2 *Federated Adam (FedAdam) [27].*

A subclass of the FedOpt protocols [27] and increases the adaptivity of FedAvg by eliminating the static aggregation at the server. To naturally apply adaptive aggregation, the clients transmit the delta $\Delta_{i+1}^{(k)} = \theta_{i+1}^{(k)} - \theta_i$ instead of the parameters; the server transmission remains as described.

One of the most popular [1] Machine Learning (ML) optimisers, Adam [16], forms the inspiration for FedAdam. FedAdam maintains a first and second-order momentum at the server, with weight decays β_1 and β_2 and the weighted average Δ_{i+1}. Respectively:

$$m_{i+1} = \beta_1 m_i + (1 - \beta_1)\Delta_{i+1} \qquad v_{i+1} = \beta_2 v_i + (1 - \beta_2)\Delta_{i+1}^2$$

The momenta are initialised at zero. As a result, momenta are initially biased to zero. To account for the bias to that zero, we adjust the momenta before updating:

$$\hat{m}_{i+1} = \frac{m_{i+1}}{1 - \beta_1^{i+1}} \qquad \hat{v}_{i+1} = \frac{v_{i+1}}{1 - \beta_2^{i+1}}$$

We update the parameters in line with Adam, using a small regularisation term ϵ and a global learning rate η_g:

$$\theta_{i+1} = \theta_i + \eta_g \frac{\hat{m}_{i+1}}{\sqrt{\hat{v}_{i+1}} + \epsilon}$$

2.1.3 *Federated Nesterov Accelerated Gradient (FedNAG) [34].*

An attempt to prevent client drift [15] in FL by incorporating Nesterov Momentum [24]. Client drift is the process of clients separately moving away from globally optimal parameters due to inhomogeneous local optimisation problems. Momentum [26] maintains the information of previous updates via a velocity term u_i scaled by a friction γ. Consequently, clients will account for each other's updates, finding a better solution quicker. The server maintains this velocity and transmits it with the parameters θ_i to the selected clients. The client returns the updated velocity with the updated parameters and optimises using the following:

$$u_{i+1}^{(k)} = \gamma u_i - \eta \sum_{(\mathbf{x},\mathbf{y}) \in B} \nabla_{\theta_i + u_i} \ell(f_{\theta_i + u_i}(\mathbf{x}), \mathbf{y})$$

$$\theta_{i+1}^{(k)} = \theta_i + u_{i+1}^{(k)}$$

The server takes the weighted average of the momentum as it would for the parameters.

2.1.4 *Federated Newton Learning (FedNL).*

For logistic regression, we also use an adapted version of FedNL [29]. Our adaptation still uses the Newton method to find the parameters for which the loss function is minimal. However, we recompute the exact Hessian for each round instead of computing the approximation using the Frobenius norm. The resulting optimisation step is:

$$\theta_{i+1}^{(k)} = \theta_i - H^{-1} \nabla_{\theta_i} \ell(f_{\theta_i}(\mathbf{x}), \mathbf{y})$$

Where the gradient and Hessian are computed over the complete dataset at once. In this protocol, the server sends the initial parameters to the clients, who respond with their Hessian and gradient. The server weightily averages both over the amount of data per client, computes the inverse of the Hessian, and finds the model update as the client would. Since the computation of the Hessian and its inverse is intense, this method is only a feasible protocol for shallow neural networks. Therefore, we only use it for logistic regression.

Game 1 Membership inference security game [30, 36], with an optimisation algorithm $\mathcal{T}$ and a distribution $\mathbb{D}$ from which n samples are included in training. The challenger randomly selects a dataset from the distribution and trains on the union of that subset and z_0 or z_1 dependent on the outcome of coinflip b. The adversary $\mathcal{A}$ can tell if z_0 was included in training if they can consistently predict b.

Input: $\mathcal{T}, \mathbb{D}, n, \mathcal{A}$

```
1: $S \sim \mathbb{D}^{n-1}$
2: $b \sim \{0, 1\}$
3: $z_0 \sim \mathbb{D}$
4: $z_1 \sim \mathbb{D}$
5: $\theta \leftarrow \mathcal{T}(S \cup \{z_b\})$
6: $\tilde{b} \leftarrow \mathcal{A}(\mathcal{T}, \mathbb{D}, n, \theta, z_0)$
```

2.2 Membership Inference Attacks

A Membership Inference Attack (MIA) is an attack that aims to determine whether a given victim data sample was part of the victim model's training dataset or not [31]. It is one of the most basic attacks [30] and for that reason is often used to audit data privacy [5]. Game 1 gives the security game for a MIA [30, 36]. The training algorithm is secure for the distribution if the adversary cannot predict b better than a random guess, i.e. the probability of a correct answer is 0.5.

To make the attack more relevant for various applications, many works alter Game 1 to create attackers of different strengths:

2.2.1 Black and white-box. A black-box attacker has querying access to the victim model to obtain $f_\theta(\mathbf{x})$ of any $\mathbf{x}$. A white-box attacker knows the full model architecture f, its parameters θ and the hyper-parameters that were used to obtain f_θ. They can access any intermediate computation of the model architecture and compute its full gradients.

2.2.2 Passive and active mode. A passive attacker can only observe the computations that would be made regardless of their attack. An active attacker may alter its computations and messages to affect other clients' computations. Depending on whether the attacker is the server or a client, this may mean the malicious server can single out clients, or malicious clients may send updates to let other clients reveal more about their data. Logically, a malicious server is more powerful.

3 RELATED WORK

3.1 Membership Inference Attacks

Shokri et al. [31] coined Membership Inference Attack (MIA) and, with that, showed how it might work. The intuition is that many ML models overfit to some degree: Modelling more complex relations than necessary to get better results on the training data at the cost of decreased performance on other data. They capture this phenomenon by training many 'shadow models' on datasets similar to the victim dataset. Shadow models have a similar architecture to the victim model and, for that reason, should show similar overfitting patterns. These patterns include overconfidence or lower loss on member data, as shown in Appendix A. A final model, the attack model, is trained to predict membership by finding these patterns in the shadow models based on the input data, prediction and target label: $(\mathbf{x}, f_{\theta_{\text{shadow}}}(\mathbf{x}), \mathbf{y})$. In the final attack, the threat model should predict the membership of input data based on the prediction of the victim model.

Yeom et al. [35] formally show the positive relation between overfitting and MIA advantage and confirm models are more vulnerable the more they overfit. Besides, they show larger models are more susceptible to MIAs, regardless of generalisation performance.

The attack of Nasr et al. [23] also shows this for their white-box attack. Their attack leverages its white-box access by considering not only the confidence of the prediction but also the loss value, the value of each activation layer and the value of the gradients of each parameterised layer. The idea is that the loss and gradient are closer to zero for members than for non-members. They say not all activation values necessarily contribute to a better attack model. The earlier layers tend to extract simpler features relevant to all possible samples. In contrast, the information of the later layers is shown to improve the attack model better since these are more prone to overfitting.

Because the attack model of Nasr et al. [23] takes more input data than just the loss and target label, it requires more than a standard binary classifier. For this, they propose an alternate architecture which forms the basis for our Ancillary Attack model architecture. We explain this and Nasr et al. their architecture in Section 4.

With their attack, they do not need to train many shadow models. Instead, they compute all the needed values on the victim model. This attack significantly improves the results of Shokri et al. [31] and reports an average membership prediction accuracy improvement of 5% or 3%pt.

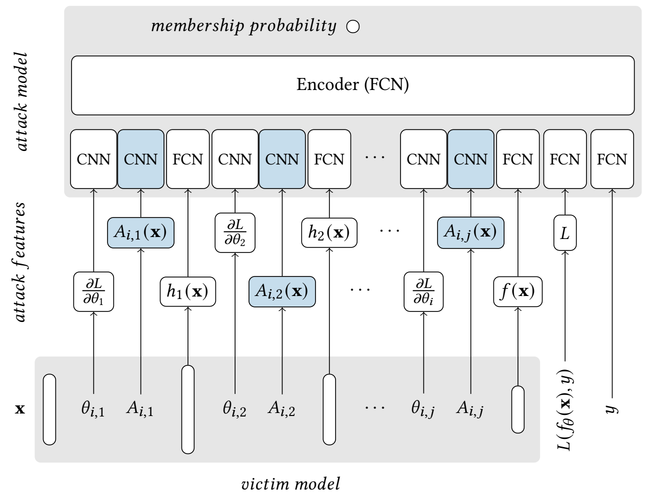


Fig. 1. The ancillary attack model architecture. $\theta_{i,j}$ is the jth layer of the ith iteration of the victim model, $A_{i,j}$ is the jth layer of the ith iteration of the ancillary variable. In blue are the components and information added to the model of Nasr et al. [23] to use the ancillary variables.

3.2 Attack Success Interpretability

Many works on MIAs make use of balanced attack accuracy [11, 19, 23, 31–33, 35]: the average prediction score on a balanced dataset of members and non-members. Carlini et al. [5] vouch for an alternate interpretation of the success of an attack. They argue that the accuracy may unreasonably increase if a model is great at predicting non-membership but can hardly predict membership. Therefore, they recommend reporting the Area Under the Curve (AUC) of the True-Positive Rate/False-Positive Rate (TPR/FPR) tradeoff characterised by a Receiver Operating Characteristic (ROC) curve along logarithmic axes. With logarithmic axes, the resulting figure focuses on the parts of the data that are susceptible to MIAs and so on the degree to which a model can predict membership rather than non-membership.

4 ANCILLARY ATTACK

Ancillary variables are the variables used to improve the optimisation: The velocity u for FedNAG and the Hessian for FedNL. As a general notation, we use A_i to mean the ancillary variable at round i.

Ancillary variables are often used for two purposes: Either they are used to ensure the cross-iteration optimisation information other than the model updates is not discarded by, for example, keeping track of the average direction of the last few updates scaled by some decay. Alternatively, they are used to ensure one optimisation iteration obtains much more information than they would otherwise. Because of this role in the learning process, they so should display smaller update steps for member data than for non-member data when the model has completed training, similar to the gradient.

With that, we propose to further the thought of Nasr et al. [23] by also accounting for communicated 'Ancillary' variables. The architecture of the attack model of Nasr et al. consists of several components for the loss, label, activation and gradient value(s) that result from the victim model prediction on the victim sample. The loss, label and activation components are a Fully Connected Network (FCN) with one hidden layer of size 128 and a Rectified Linear Unit (ReLU) activation function. The FCN has an output vector size of 64 and a 20% dropout probability.

The gradients are first put in a Convolutional Neural Network (CNN) component before being put into an FCN with the same architecture as before. The CNN is 1 convolutional layer with a stride of 1, a dropout probability of 20%, and a ReLU activation function. The kernel width is 1, and the height is the size of the layer's input on which the gradient is computed. The output of the CNN is flattened before it is put into the FCN. Similarly, the outputs of the FCN are flattened before all outputs are concatenated and put into the encoder. The encoder is another FCN with three hidden layers of sizes 256, 128, 64, 20% dropout probability and a ReLU activation function. The encoder output size is 1 and forms the membership prediction. To account for multiple model rounds, the input of the FCN and CNN components can be deepened to the number of captured rounds - we do not investigate this. Flattening happens over all dimensions of the component output vector.

We adjust this attack model architecture by considering the update to the ancillary variable given an ancillary variable aggregate and the input data. We note this update as $A_i(\mathbf{x})$ and have drawn a general architecture in Figure 1. Similar to the loss and the gradient, this value should be smaller for members than for non-members, as the update for the member has already been accounted for in the initial ancillary variable. Since this value has the same dimensions as the gradient, the ancillary components have the same architecture as the gradient components.

5 EXPERIMENTAL SETUP

For our experiments, we implemented an MIA workbench[1] in Python. This workbench can be used to describe or import PyTorch [25] models to train Flower [3] FL systems on different HuggingFace [20] or self-imported datasets split iid or non-iid using FedArtML [10]. The workbench can simulate FedAvg [22], FedAdam [27], FedNAG [34] and our adaptation of FedNL [29]. After the simulation is complete, it attacks as described in this section. All hyperparameters and variables mentioned can be optimised using grid search, with logging to W&B [4].

5.1 Datasets

5.1.1 CIFAR. CIFAR-10 and CIFAR-100 [17] are popular benchmark datasets to evaluate image recognition models. Both contain 32x32 coloured images. CIFAR-10 contains 60,000 samples in 10 classes, with 6,000 samples per class. CIFAR-100 contains 60,000 samples in 100 classes, with 600 samples per class. We use these datasets to experiment with attacking a more complex task. For both datasets, we simulate a federation of 100 clients where 10 clients are selected per round. The data is split iid and each client has an equal amount. To have a balanced member-to-non-member ratio, we use 30,000 samples for training, 7,500 for testing, and the other 22,500 only for the attack experiments. For both datasets, we apply random cropping with a padding of four pixels, random horizontal flipping, random rotation with a maximum of 15°, and normalisation on the training data. The testing data is only normalised.

5.1.2 Privacy Datasets. We use the Purchase100 and Texas100 datasets to evaluate simpler but more sensitive data usage with FL. The Purchase100 dataset is a derivation from the Kaggle 'Acquire Valued Shoppers Challenge' dataset[2] and the Texas100 dataset is a derivation of the Hospital Discharge Data of the Texas Department of State Health Services[3]. Both derivations are described by Shokri et al. [31].

Purchase100 contains 600 binary features representing whether a customer has ever bought a specific product. The task is to predict one of 100 shopping patterns. Texas100 contains 6,169 binary features that each represent some attribute of the patient, e.g. whether the cause of injury was drug misuse. The task is to predict one of the 100 most frequent procedures; other procedures are excluded. To have a balanced member-to-non-member ratio for both datasets, we randomly select 10,000 training samples, 2,500 testing samples, and 7,500 other samples only used for the attack experiments. In this scenario, we model a smaller federation of 10 clients where each client is selected per round. The data is split iid and each client has an equal amount. This scenario is especially relevant for hospitals or other small federations that work with highly sensitive data. We do not apply data augmentation to Purchase100 and Texas100.

5.2 Target Models

For the CIFAR-10 and 100 datasets, we use a ResNet [12] with 18 and 34 layers, respectively. For the privacy datasets, we use simple FCNs and logistic regression. We use logistic regression since sensitive applications often require simpler models to make the sensitive prediction more easily explainable or because the weights of simpler models may scientifically be more relevant as in [6]. For Purchase100, we use an FCN with layer sizes 600, 1024, 512, 256, 100. For Texas100, we use an FCN with layer sizes 6169, 2048, 1024, 512, 256, 100. Both models use a tanh activation function without dropout. All models of all datasets are trained using a cross-entropy loss function.

In Appendix B, we describe which hyperparameters we use for which simulation. For all protocols except FedNL, the clients train locally for 4 epochs. For FedNL, we use 1 local epoch. We stop global learning if the loss has not decreased by 0.1% in the last 20 rounds. In Table 1, we show the prediction accuracy on the training, testing and validation data per simulation. From this, we confirm that alternatives to FedAvg result in a better performance. Though the accuracy is not too impressive compared to the central setting [7, 9], for a federated setting, these results are in line with the original proposals [22, 27, 34]. For the full learning curves of the protocols, see Appendix C.

From Table 1, we see that overall, FedAvg, in general, takes longer to converge to equal performance than other protocols. Furthermore, we see that FedNAG is able to converge quicker for simpler datasets

[1]https://github.com/P1mguin/conFEDential

[2]https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data

[3]https://www.dshs.texas.gov/THCIC/Hospitals/Download.shtm

Table 1. Model performance of target models on datasets for different optimisers on train, test, and validation data and the number of global rounds until converged.

Simulation	Protocol	Global rounds	Train	Test
CIFAR10 ResNet18	FedAvg	328	68.5%	55.7%
	FedAdam	134	71.4%	57.3%
	FedNAG	921	67.3%	54.8%
CIFAR100 ResNet34	FedAvg	482	39.3%	21.7%
	FedAdam	83	37.8%	20.0%
	FedNAG	317	44.0%	23.0%
Purchase100 FCN	FedAvg	34	99.4%	78.1%
	FedAdam	98	100.0%	76.7%
	FedNAG	30	100.0%	77.4%
Purchase100 Logistic regression	FedAvg	274	93.4%	57.1%
	FedAdam	43	95.1%	61.1%
	FedNAG	206	94.8%	57.2%
	FedNL	24	89.0%	26.3%
Texas100 FCN	FedAvg	37	71.7%	51.8%
	FedAdam	78	68.5%	51.3%
	FedNAG	15	70.1%	51.3%
Texas100 Logistic regression	FedAvg	117	81.4%	56.5%
	FedAdam	130	79.4%	55.8%
	FedNAG	146	78.2%	56.0%
	FedNL	23	85.4%	53.0%

and that FedAdam converges quicker for all datasets. This is because FedNAG can overcome the increasing amount of local minima that exist when the problem gets more complex and, therefore, is not stopped. FedAdam does converge to these local minima but does it quicker than FedAvg and FedNAG as a result of the adaptive learning rate. This adaptive learning rate is not a great help in overcoming local minima, and therefore, the model converges more quickly.

Reconsidering Table 1, we see that FedNAG has a lower difference in prediction accuracy between the training and testing data for CIFAR10 and CIFAR100. This results from the momentum that helps to overcome more frequent local minima. When the problem gets simpler, the frequency of local minima decreases, so FedNAG starts to converge more quickly than FedAvg, with similar prediction accuracy. FedNL converges much quicker than the other protocols. As explained, FedNL takes big and accurate steps that take more computational power. Furthermore, since these big steps are taken on the training data, we see a significant difference between the training and testing data, indicating a large degree of overfitting.

5.3 Threat Model

In the experiments, we test for a global and local passive white-box attacker, i.e. the server and a client. For both, we assume the strongest threat: For the global attacker, we assume they can create an auxiliary dataset that overlaps precisely with the training/testing dataset of all clients combined but is only missing the victim sample, in line with Game 1. They have access to all aggregate variables and a balanced dataset they use to train the Ancillary Attack model. The victim is randomly selected from all data.

Table 2. Description of which scenario uses which component. For instance, scenario D uses all components.

Scenario name	Loss	Label	Activation	Gradient	Ancillary
A	True	True	False	False	False
B	True	True	True	True	False
C	True	True	False	False	True
D	True	True	True	True	True

We assume the local attacker has been selected for each training round. So, they have access to all aggregates transmitted from the server to the clients. The local attacker uses these aggregates in combination with their local dataset to train the Ancillary Attack model. The dataset of the local attacker comprises half members and half non-members and is of a size equivalent to the total data of the federation divided equally among all clients. The victim is randomly selected from all data that does not belong to the local attacker.

5.4 Attack Simulation

We run experiments by varying the components used in the Ancillary Attack model. The attacker can use the ancillary component if the attacker has access to the ancillary aggregate and if one exists. For instance, the local attacker does not have access to the aggregate of the Hessian as that information is only sent *to* the server and not returned. In this example, we could not use an ancillary component.

By default, we use the loss value and target label. The gradient and activation components and the ancillary components are cross-examined to obtain a total of four configurations if a protocol has ancillary variables to which the attacker has access and two if they do not. For clarity, we name each scenario in Table 2. We will refer to each scenario as named in Table 2.

We train the attack model for the global attacker on all the data except the victim sample. Of that data, 85% is used for training and 15% for validation. The local attacker similarly uses 85% of its data as training and 15% for validation. The data to which the local attacker does not have access is used as testing data. For the global attacker, this is the target victim. For the local attacker, the victim and the data of the other clients. For both attackers, the data is put into the model of the aggregation round that has the highest accuracy, i.e. the final model. If multiple rounds resulted in the same accuracy, we use the earlier round.

We train the attack model using Adam [16] with a learning rate of 0.001, betas of 0.9 and 0.999, an epsilon of 0.0001, and no weight decay. We use L2 loss for the attack model. We stop training if the average loss of the last 5 rounds has not decreased by 0.1% for 10 consecutive rounds.

The attack models often converge to a performance equivalent to that of guessing. Therefore, we repeat the simulation until we get a result that is not guessing, with a maximum of 10 attempts. If an experiment did not produce a usable model, we report the evaluation metrics in *italics*. To see how often each experiment was run until a result was found, refer to Appendix D.

5.5 Evaluation Metrics

5.5.1 Attack Accuracy. The number of correctly predicted test samples when the membership decision threshold is 0.5, divided by the total number of samples. The decision threshold is the membership probability that is at least required for us to consider the model prediction membership classification. We use accuracy because of its usage by other works, increasing comparability.

5.5.2 ROC (Logarithmic Scale). As stated by Carlini et al. [5], the decision threshold can be put anywhere to create arbitrary success metrics. Therefore, we disclose the ROC along linear and logarithmic axes to make a fair judgement if metric components increase membership prediction rather than non-membership prediction. For simplicity, we sometimes report the AUC instead of the full curve. We will clarify when the AUC is computed for the logarithmic axes.

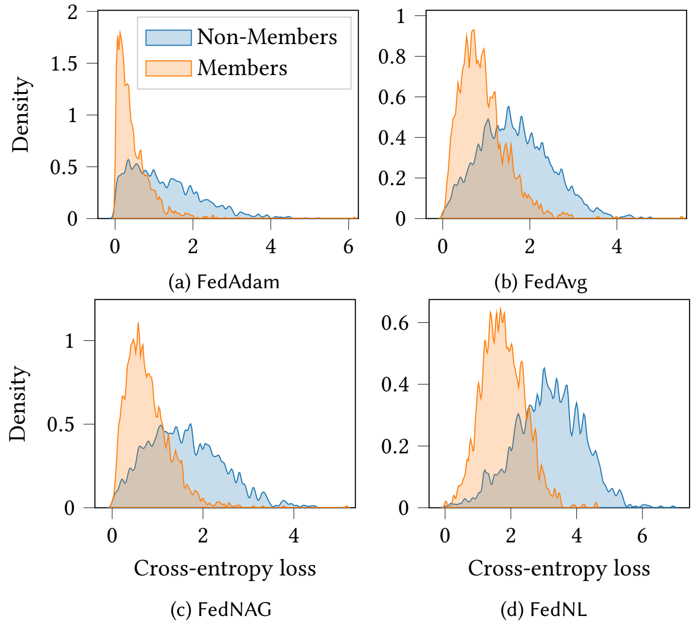


Fig. 2. Difference in loss distribution between members and non-members for logistic regression on Purchase100.

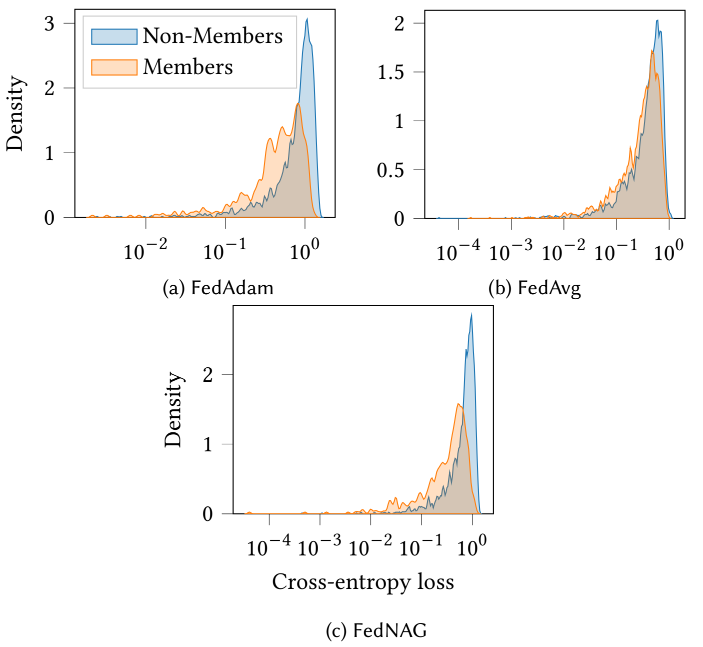


Fig. 3. Difference in loss distribution between members and non-members for ResNet18 on CIFAR-10.

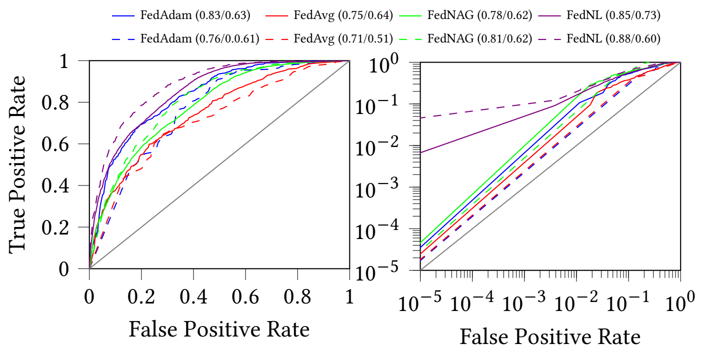


Fig. 4. ROC curves for a global attacker, attacking logistic regression for Purchase100 optimised with various learning methods. The attacker uses only the loss and the label component. In dashed lines, the validation ROC and the continuous lines, the training ROC. The first value in the legend is the AUC under the linear axes, and the second value is computed along logarithmic axes.

6 EVALUATION & DISCUSSION

6.1 Impact of Optimisation Method

We start the evaluation by investigating the effect the optimisation method has on the performance of the MIA without the ancillary component enabled. In Table 3, we show that, in general, a global attacker can achieve a higher attack accuracy when a federation uses an optimisation protocol alternative to FedAvg. To understand this, we start by evaluating the experiment set that sees the highest relative changes in MIA accuracy. We compare the difference in loss distribution over all classes for logistic regression on Purchase100 in Figure 2. From this figure, we see that alternatives to FedAvg achieve a lower loss for both members and non-members - with FedNL being the exception for this dataset. However, we can also see that alternatives to FedAvg have a greater statistical distance between the member and non-member loss distribution. In Table 4, we show the generalisation error next to the MIA accuracy. In this table, we take the generalisation error as the difference between the mean cross-entropy loss. From this table, we can draw that not per se low model performance, but the generalisation error increases membership inference susceptibility. Though alternatives have smaller differences in prediction accuracy between member

Table 3. Accuracy of the global attack over FedAdam, FedAvg, FedNAG and FedNL using only the loss and label. The accuracy is the weighted combination of the training, validation and testing accuracy, weighted by the amount of data in each split. FedNL was only used on logistic regression models, so the other model architectures do not apply to FedNL.

	CIFAR-10 ResNet18	**CIFAR-100 ResNet34**	**Purchase100 Logistic regression**	**Purchase100 FCN**	**Texas100 Logistic regression**	**Texas100 FCN**
FedAdam	60%	65%	69%	65%	68%	60%
FedAvg	52%	62%	69%	60%	62%	58%
FedNAG	61%	71%	70%	63%	61%	59%
FedNL	n/a	n/a	75%	n/a	66%	n/a

Table 4. The increase in Membership Inference Attack (MIA) accuracy and increase in the difference in expected cross-entropy loss (generalisation error) for non-members and members, relative to the FedAvg protocol for logistic regression on Purchase100.

	Generalisation error	**Relative increase**	**MIA accuracy**	**Relative increase**
FedAvg	0.75	n/a	69%	n/a
FedAdam	0.80	7%	69%	0%
FedNAG	0.80	7%	70%	1%
FedNL	1.38	80%	75%	9%

Table 5. The increase in Membership Inference Attack (MIA) accuracy and increase in the difference in expected cross-entropy loss (generalisation error) for non-members and members, relative to the FedAvg protocol for ResNet18 trained on CIFAR-10.

	Generalisation error	**Relative increase**	**MIA accuracy**	**Relative increase**
FedAvg	0.48	n/a	52%	n/a
FedAdam	2.91	506%	60%	15%
FedNAG	2.76	475%	61%	17%

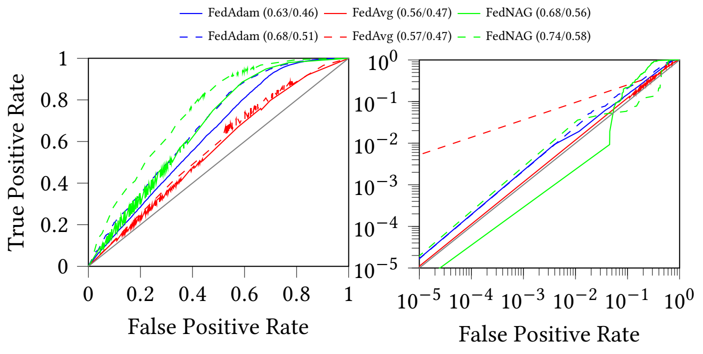


Fig. 5. ROC curves for a global attacker, attacking ResNet18 trained for CIFAR-10 optimised with various learning methods. The attacker uses only the loss and the label component. In dashed lines, the validation ROC and the continuous lines, the training ROC. The first value in the legend is the AUC under the linear axes, and the second value is computed along logarithmic axes.

and non-member data, as shown in Table 1. The generalisation error for them is greater. This is because alternatives to FedAvg can use their ancillary variables to get deeper into the minimum in which they converge. As a result, they predict the same result but do it more confidently. This difference is not shown in the prediction accuracy because both models still predict the same value. It is shown in the difference in the cross-entropy loss values as that value accounts for the confidence of the prediction. Therefore, alternatives to FedAvg seem to carry an increased susceptibility to MIAs.

To show that this relation holds for other datasets, we also describe ResNet18 on CIFAR10, which results in the smallest relative difference in MIA accuracy between optimisation protocols. In Figure 3, we again visualise the difference between non-members' and members' loss values. To make the subtle difference clearer, this time, we show it along a logarithmic axis. From this, we understand the results of Table 5, which also show that an increase in generalisation error coincides with an increase in MIA accuracy. We see that our earlier patterns uphold: the bigger the distance in distribution between members and non-members, the smaller the resilience to standard MIAs.

Finally, in Figure 4 and Figure 5, we show what this difference does to the ROC and the AUC for both logistic regression on Purchase100 and ResNet18 trained on CIFAR-10. In this figure, we see that the performance again increases for alternatives to FedAvg. Also, we see that this performance is not only an average case increase but also an increase in the AUC on the logarithmic axes. That means that alternatives to FedAvg are not only more vulnerable on an average case but also more vulnerable outlying data samples are now at increased risk.

So, we have shown that the FedAdam, FedNAG and FedNL protocols result in a machine learning model that is more susceptible to MIAs than optimisation with the FedAvg would have been. Despite these protocols resulting in better overall prediction accuracy, there is a greater difference in prediction accuracy for member and non-member data. Therefore, they make it easier to tell apart whether a prediction corresponds to a member or non-member input, increasing susceptibility to MIAs.

6.2 Impact of Components

In Table 6, we show the MIA accuracy of our attack model trained with various components by a global attacker. Additionally, in Table 7, we show what impact the new components have on the AUC. As shown and explained by Nasr et al. [23] and Carlini et al. [5], the

Table 6. The accuracy of the global attack over FedAdam, FedAvg, FedNAG, and FedNL with attack models using different components. Scenario A uses the loss and label components; scenario B uses the loss, label, activation and gradient components; scenario C uses the loss, label and ancillary component; and scenario D uses the loss, label, activation, gradient and ancillary component. The accuracy is the weighted combination of the training, validation, and testing accuracy, weighted by the amount of data of each split. Scenarios C and D do not apply to the protocols without ancillary variables. We only applied FedNL on logistic regression models. The results in italics highlight experiments that did not converge to more than guessing after ten attempts.

	CIFAR-10				**CIFAR-100**				**Purchase100**				**Purchase100**				**Texas100**				**Texas100**			
	ResNet18				**ResNet34**				**Logistic regression**				**FCN**				**Logistic Regression**				**FCN**			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
FedAdam	60%	69%	n/a	n/a	65%	69%	n/a	n/a	69%	77%	n/a	n/a	65%	70%	n/a	n/a	68%	74%	n/a	n/a	60%	67%	n/a	n/a
FedAvg	52%	62%	n/a	n/a	62%	64%	n/a	n/a	69%	71%	n/a	n/a	60%	68%	n/a	n/a	62%	69%	n/a	n/a	58%	66%	n/a	n/a
FedNAG	61%	69%	59%	66%	71%	75%	72%	72%	70%	70%	70%	71%	63%	66%	64%	*51%*	61%	66%	61%	68%	59%	61%	56%	63%
FedNL	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	75%	88%	89%	85%	n/a	n/a	n/a	n/a	66%	79%	81%	*53%*	n/a	n/a	n/a	n/a

Table 7. The ROC AUC of the global attack over FedAdam, FedAvg, FedNAG, and FedNL with attack models using different components. Scenario A uses the loss and label components; scenario B uses the loss, label, activation and gradient components; scenario C uses the loss, label and ancillary component; and scenario D uses the loss, label, activation, gradient and ancillary component. The first value is the ROC AUC computed along linear axes; the second value is computed along logarithmic axes. Both values are the weighted combination of the ROC AUC, weighted by the amount of data of each split. Scenarios C and D do not apply to the protocols without ancillary variables. We only applied FedNL on logistic regression models. The results in italics highlight experiments that did not converge to more than guessing after ten attempts.

	CIFAR-10				**CIFAR-100**				**Purchase100**				**Purchase100**				**Texas100**				**Texas100**			
	ResNet18				**ResNet34**				**Logistic regression**				**FCN**				**Logistic regression**				**FCN**			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
FedAdam	0.65 0.49	0.71 0.63	n/a	n/a	0.69 0.55	0.69 0.54	n/a	n/a	0.78 0.63	0.85 0.61	n/a	n/a	0.66 0.54	0.65 0.61	n/a	n/a	0.69 0.62	0.75 0.63	n/a	n/a	0.65 0.49	0.80 0.59	n/a	n/a
FedAvg	0.56 0.47	0.61 0.55	n/a	n/a	0.66 0.52	0.67 0.55	n/a	n/a	0.74 0.57	0.75 0.61	n/a	n/a	0.62 0.48	0.68 0.56	n/a	n/a	0.68 0.59	0.76 0.62	n/a	n/a	0.59 0.49	0.76 0.51	n/a	n/a
FedNAG	0.70 0.56	0.76 0.70	0.69 0.56	0.79 0.81	0.67 0.60	0.71 0.62	0.71 0.56	0.70 0.59	0.78 0.59	0.77 0.61	0.79 0.65	0.78 0.62	0.63 0.51	0.65 0.54	0.64 0.59	0.65 0.53	0.66 0.57	0.73 0.58	0.67 0.57	0.65 0.58	0.61 0.53	0.68 0.55	0.68 0.83	*0.52* *0.49*
FedNL	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.85 0.71	0.94 0.71	0.96 0.75	0.96 0.76	n/a	n/a	n/a	n/a	0.74 0.52	0.83 0.65	0.73 0.59	*0.51* *0.50*	n/a	n/a	n/a	n/a

inclusion of the activation and gradient component increases the attack performance of the attack model. As shown in Tables 6 and 7, our experiments show the same pattern.

6.2.1 Impact of the Ancillary Component. From our results, we see that the ancillary component shows a minor improvement in average case evaluation metrics and a greater improvement in metrics fixating on predicting membership rather than non-membership. Moreover, we see that the improvement decreases as the data and model get more complex. Similar to the generalisation error of Figures 2 and 3, the updates to the ancillary variable get more nuanced as the complexity of the model and data increases. A subtle difference is harder to model, and therefore, the improvement decreases on an average case. A small part of the samples are still not very subtle, which is what is reflected in the ROC AUC (log).

Across optimisation methods, we find that FedNL more easily overfits than FedNAG. This is because FedNL itself is more prone to overfitting than FedNAG, as we show in Table 1. As a result, this is not the result of a difference in complexity. Rather, this is because different protocols lead to different generalisation errors, which leads to a difference in MIA susceptibility, as we showed in the previous section.

In Figure 6, we show what effect the ancillary component has on the AUC along linear and logarithmic axes. From this, it is clear that the ancillary component does not greatly improve the average case attack, but it does improve it much for a minor part of the dataset, which is more threatening.

This means that we have shown that the ancillary component does not, on average, help with increasing the MIA accuracy. Rather, the ancillary component improves to predict membership on the outliers in the dataset and, therefore, is a useful tool in predicting membership. Furthermore, the more complex input data and model architecture get, the smaller the contribution of the ancillary component on membership prediction. There is no notable difference in the ancillary component across optimisation methods, only that for some protocols, it cannot be used.

6.3 Impact of Attacker Role

So far, our evaluations have only regarded a global attacker. In Table 8, we show the MIA accuracy for the attack model trained with various components by a local attacker and in Table 9, we show what AUC was achieved using these models.

In Tables 8 and 9, we see that for all datasets, the attacker can achieve results that are close to a global attacker. Considering that the local attacker sometimes only had access to 1% of all data, we find it reasonable to consider each participant in the federation a threat.

Overall, we see that the patterns we mentioned concerning the impact of the optimisation method and the impact of components are equal for a global and local attacker. For the scenarios in which we used not only the loss and label components, we sometimes see that a local attacker outperforms a global attacker. In theory, our global attacker should be able to at least achieve the performance

Table 8. The accuracy of the local attack over FedAdam, FedAvg, FedNAG, and FedNL with attack models using different components. Scenario A uses the loss and label components; scenario B uses the loss, label, activation and gradient components; scenario C uses the loss, label and ancillary component; and scenario D uses the loss, label, activation, gradient and ancillary component. The accuracy is the weighted combination of the training, validation, and testing accuracy, weighted by the amount of data of each split. Scenarios C and D do not apply to the protocols without ancillary variables. We only applied FedNL on logistic regression models. The local attacker cannot apply the ancillary attack on FedNL as they do not have access to the ancillary variable.

	CIFAR-10				**CIFAR-100**				**Purchase100**				**Purchase100**				**Texas100**				**Texas100**			
	ResNet18				**ResNet34**				**Logistic regression**				**FCN**				**Logistic Regression**				**FCN**			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
FedAdam	57%	61%	n/a	n/a	67%	65%	n/a	n/a	69%	78%	n/a	n/a	63%	65%	n/a	n/a	68%	74%	n/a	n/a	60%	61%	n/a	n/a
FedAvg	54%	57%	n/a	n/a	58%	59%	n/a	n/a	65%	67%	n/a	n/a	59%	59%	n/a	n/a	56%	69%	n/a	n/a	58%	62%	n/a	n/a
FedNAG	56%	*54%*	*52%*	62%	72%	65%	64%	*56%*	71%	69%	70%	59%	59%	61%	60%	60%	58%	57%	59%	58%	56%	59%	57%	58%
FedNL	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	71%	88%	n/a	n/a	n/a	n/a	n/a	n/a	61%	61%	n/a	n/a	n/a	n/a	n/a	n/a

Table 9. The ROC AUC of the local attack over FedAdam, FedAvg, FedNAG, and FedNL with attack models using different components. Scenario A uses the loss and label components; scenario B uses the loss, label, activation and gradient components; scenario C uses the loss, label and ancillary component; and scenario D uses the loss, label, activation, gradient and ancillary component. The first value is the ROC AUC computed along linear axes; the second value is computed along logarithmic axes. Both values are the weighted combination of the ROC AUC, weighted by the amount of data of each split. Scenarios C and D do not apply to the protocols without ancillary variables. We only applied FedNL on logistic regression models. The local attacker cannot apply the ancillary attack on FedNL as they do not have access to the ancillary variable.

	CIFAR-10				**CIFAR-100**				**Purchase100**				**Purchase100**				**Texas100**				**Texas100**			
	ResNet18				**ResNet34**				**Logistic regression**				**FCN**				**Logistic regression**				**FCN**			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
FedAdam	0.61	0.62	n/a	n/a	0.67	0.65	n/a	n/a	0.71	0.75	n/a	n/a	0.64	0.65	n/a	n/a	0.71	0.76	n/a	n/a	0.62	0.67	n/a	n/a
	0.54	0.56			0.53	0.55			0.59	0.62			0.54	0.56			0.63	0.65			0.53	0.54		
FedAvg	0.56	0.59	n/a	n/a	0.59	0.61	n/a	n/a	0.69	0.71	n/a	n/a	0.61	0.62	n/a	n/a	0.61	0.73	n/a	n/a	0.57	0.68	n/a	n/a
	0.51	0.55			0.52	0.54			0.56	0.61			0.51	0.53			0.53	0.57			0.50	0.52		
FedNAG	0.61	*0.50*	*0.51*	0.63	0.70	0.66	0.71	*0.54*	0.74	0.72	0.75	0.62	0.62	0.64	0.63	0.64	0.64	0.64	0.64	0.65	0.59	0.63	0.60	0.63
	0.52	*0.50*	*0.48*	0.54	0.62	0.59	0.65	*0.49*	0.61	0.61	0.69	0.63	0.53	0.53	0.58	0.57	0.60	0.59	0.68	0.67	0.50	0.53	0.60	0.52
FedNL	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.77	0.73	n/a	n/a	n/a	n/a	n/a	n/a	0.63	0.60	n/a	n/a	n/a	n/a	n/a	n/a
									0.72	0.55							0.54	0.55						

of our local attacker by using the same fraction of their data as a local attacker has. This difference likely comes from the model instability that also causes the model to frequently converge to a result equivalent to guessing.

Furthermore, we think the local attacker can achieve accuracy comparable to that of a global attacker by how we designed the experiment. For that, two factors are in play: The first is that the data is split iid amongst the clients. As a result, the local attacker is given samples from the complete member and non-member data. Consequently, the local attacker has at least learned *something* about all the data it faces. In contrast, if the attacker only has access to the very tails of both distributions, it could hardly predict anything about the tail at the opposite end of the distribution, as it has learned *nothing* about it. This aligns with what Liu and Dong [21] describe. When more clients are introduced, non-iid federations are better protected than iid federations are against MIAs.

Secondly, the local attacker is given a balanced dataset. In other words, they have as many members as non-members. In a typical setting, the client must use more than half of its data to participate in training - if it behaves honestly - this would lower the amount of non-member data drastically. In such a scenario, the client would have a very unbalanced dataset with which to optimise their attack model. This would further harm their attack model performance.

So, we have shown that the local attacker achieves similar results as the global attacker. In some cases, the local attacker outperforms the global attacker due to the instability of the model architecture. We think the local attacker achieves these good results because the data is split iid and because they have access to a balanced dataset.

7 FUTURE WORK

From the evaluation and discussion, various experiments call for further investigation. Firstly, the idea of the Ancillary Attack can be explored more thoroughly. One option is to let the attacker maintain ancillary variables outside of the federation's protocol. For instance, an attacker could maintain a local momentum of the received global parameters. This momentum is never shared, but, as we have shown, ancillary variables can increase the prediction accuracy of a Membership Inference Attack (MIA). Then, even beyond protocols, the attacker can keep track of various ancillary variables, which could increase their attack likelihood. Future work should determine if and which variables work best for an attacker and, more importantly, how to defend against such 'self-tailoring Ancillary Attacks'.

Other works on the Ancillary Attack may look into the effect different layers have on the quality of the MIA. Similar to Nasr et al. [23], it may be that earlier layers provide less information than later layers of the model architecture do. This may greatly increase the time required for convergence of the model.

Beyond the Ancillary Attack, future work could advance the knowledge of the MIA susceptibility of different optimisation protocols in Federated Learning (FL) by repeating similar experiments on greater datasets and different models. For instance, more federations may be simulated with equal datasets, but different models, such

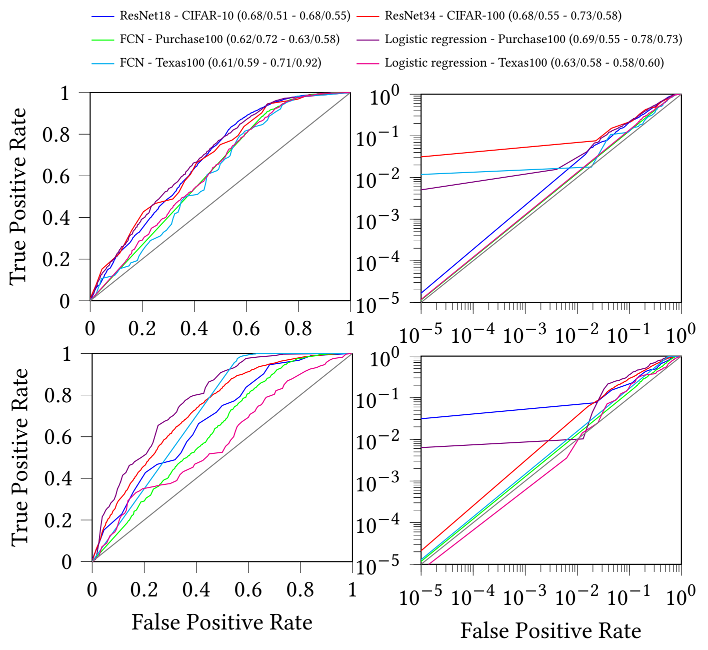


Fig. 6. Validation ROC curves for a global attacker, attacking models that are optimised using FedNAG. The upper figures display the ROC curves for an attack model using only the loss and label component; the lower figures also use the ancillary component. The first pair in the legend are without the ancillary component, and the remaining pair is with the ancillary component. Within the pair, the first value is the AUC under the linear axes; the second is the AUC along logarithmic axes.

as DenseNet [13] or AlexNet [18]. By doing so, these simulations can find the effect the model size has on the difference in MIA susceptibility between protocols. To explain, the choice in protocol may matter more for larger than for smaller models. Also, it is interesting to see whether providing the attack model with multiple input rounds may cause one protocol to become more susceptible than other protocols because of the cross-iteration optimisation information in the ancillary variables and model parameters.

It is interesting to investigate further the difference in inference power between local and global attackers. We have shown that a local attacker with a small amount of data can achieve similar results to a global attacker if their attack dataset is balanced and the data is split iid. It is interesting to see whether these two conditions are required for a local attack to succeed. Besides, it can be investigated what percentage of the data is required for a local attacker to become as powerful as the theoretical limit of the global attacker.

Finally, to find out the real implications the choice in optimisation method has on privacy, future work should further the security audit and find out its effect on, for instance, attribute inference or full data reconstruction. In these experiments, the thesis of the ancillary attack may contribute to furthering awareness of what to defend against and how to use federated learning.

8 CONCLUSION

In this paper, we considered a new variable that plays a role in the susceptibility of a Federated Learning (FL) system to a Membership Inference Attack (MIA), the optimisation protocol. We considered FedAvg [22], FedAdam [27], FedNAG [34] and FedNL [29], to show that some protocols lead to a greater generalisation error which leads to a greater susceptibility to Membership Inference Attacks (MIAs). In our results, we found that alternatives to FedAvg cause a greater generalisation error and, hence, should be considered when defending against MIAs.

To investigate the risk that is caused by alternative optimisation strategies further, we created the Ancillary Attack. This attack leverages the ancillary variables that some FL protocols use to increase their prediction accuracy or decrease their required training time. We showed that the update to the ancillary variable can be used alongside typical MIA models to increase the membership prediction for a white box attacker. This is because the ancillary variable is used for two purposes: Either it is used to maintain information learned in one iteration across multiple iterations, or it is used to generate much more information in one iteration. We showed that because of this increase in information, the ancillary component works. From this, we conclude that the difference in pattern updates on the ancillary variables between members and non-members can be used to distinguish membership in the training data.

Finally, we showed that a local attacker can successfully apply (ancillary) attacks on different optimisation protocols if the data is spread iid and the local attacker is given a balanced dataset.

REFERENCES

[1] Charu C. Aggarwal. 2018. *Neural Networks and Deep Learning: A Textbook.* Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-94463-0
[2] Borja Balle, Giovanni Cherubin, and Jamie Hayes. 2022. Reconstructing Training Data with Informed Adversaries. In *2022 IEEE Symposium on Security and Privacy (SP).* 1138–1156. https://doi.org/10.1109/SP46214.2022.9833677
[3] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. 2022. Flower: A Friendly Federated Learning Research Framework. arXiv:2007.14390 (March 2022). https://doi.org/10.48550/arXiv.2007.14390 arXiv:2007.14390 [cs, stat].
[4] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. https://www.wandb.com/ Software available from wandb.com.
[5] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership Inference Attacks From First Principles. arXiv:2112.03570 (April 2022). https://doi.org/10.48550/arXiv.2112.03570 arXiv:2112.03570 [cs].
[6] Erin Craig, Chenyang Zhong, and Robert Tibshirani. 2021. Survival stacking: casting survival analysis as a classification problem. arXiv:2107.13480 (July 2021). http://arxiv.org/abs/2107.13480 arXiv:2107.13480 [stat].
[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 (June 2021). https://doi.org/10.48550/arXiv.2010.11929 arXiv:2010.11929 [cs].
[8] Ahmed Roushdy Elkordy, Jiang Zhang, Yahya H. Ezzeldin, Konstantinos Psounis, and Salman Avestimehr. 2022. How Much Privacy Does Federated Learning with Secure Aggregation Guarantee? arXiv:2208.02304 (Aug. 2022). https://doi.org/10.48550/arXiv.2208.02304 arXiv:2208.02304 [cs, math].
[9] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-Aware Minimization for Efficiently Improving Generalization. arXiv:2010.01412 (April 2021). https://doi.org/10.48550/arXiv.2010.01412 arXiv:2010.01412 [cs, stat].
[10] Daniel Mauricio Jimenez Gutierrez, Aris Anagnostopoulos, Ioannis Chatzigiannakis, and Andrea Vitaletti. 2024. FedArtML: A Tool to Facilitate the Generation of Non-IID Datasets in a Controlled Way to Support Federated Learning Research.

IEEE Access 12 (2024), 81004–81016. https://doi.org/10.1109/ACCESS.2024.3410026

[11] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2018. LOGAN: Membership Inference Attacks Against Generative Models. arXiv:1705.07663 (Aug. 2018). https://doi.org/10.48550/arXiv.1705.07663 arXiv:1705.07663 [cs].

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

[14] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. arXiv:1912.04977 (March 2021). http://arxiv.org/abs/1912.04977 arXiv:1912.04977 [cs, stat].

[15] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2021. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. arXiv:1910.06378 (April 2021). http://arxiv.org/abs/1910.06378 arXiv:1910.06378 [cs, math, stat].

[16] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 (Jan. 2017). https://doi.org/10.48550/arXiv.1412.6980 arXiv:1412.6980 [cs].

[17] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images.* Technical Report.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (May 2017), 84–90. https://doi.org/10.1145/3065386

[19] Klas Leino and Matt Fredrikson. 2020. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. 1605–1622. https://www.usenix.org/conference/usenixsecurity20/presentation/leino

[20] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A Community Library for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 175–184. arXiv:2109.02846 [cs.CL] https://aclanthology.org/2021.emnlp-demo.21

[21] Siqi Liu and Fang Dong. 2023. MIA-FedDL: A Membership Inference Attack against Federated Distillation Learning. In *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 1148–1153. https://doi.org/10.1109/CSCWD57460.2023.10152831

[22] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2023. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629 (Jan. 2023). http://arxiv.org/abs/1602.05629 arXiv:1602.05629 [cs].

[23] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. 739–753. https://doi.org/10.1109/SP.2019.00065

[24] Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate O (1/k** 2). *Doklady Akademii Nauk SSSR* 269, 3 (1983), 543.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 (Dec. 2019). https://doi.org/10.48550/arXiv.1912.01703 arXiv:1912.01703 [cs, stat].

[26] Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics* 4, 5 (1964), 1–17.

[27] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. 2021. Adaptive Federated Optimization. arXiv:2003.00295 (Sept. 2021). https://doi.org/10.48550/arXiv.2003.00295 arXiv:2003.00295 [cs, math, stat].

[28] Herbert Robbins and Sutton Monro. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22, 3 (Sept. 1951), 400–407. https://doi.org/10.1214/aoms/1177729586

[29] Mher Safaryan, Rustem Islamov, Xun Qian, and Peter Richtárik. 2022. FedNL: Making Newton-Type Methods Applicable to Federated Learning. arXiv:2106.02969 (May 2022). https://doi.org/10.48550/arXiv.2106.02969 arXiv:2106.02969 [cs, math].

[30] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella-Béguelin. 2023. SoK: Let the Privacy Games Begin! A Unified Treatment of Data Inference Privacy in Machine Learning. arXiv:2212.10986 (April 2023). https://doi.org/10.48550/arXiv.2212.10986 arXiv:2212.10986 [cs].

[31] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. 3–18. https://doi.org/10.1109/SP.2017.41

[32] Anshuman Suri, Pallika Kanani, Virendra J. Marathe, and Daniel W. Peterson. 2023. Subject Membership Inference Attacks in Federated Learning. arXiv:2206.03317 (June 2023). https://doi.org/10.48550/arXiv.2206.03317 arXiv:2206.03317 [cs].

[33] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2021. Demystifying Membership Inference Attacks in Machine Learning as a Service. *IEEE Transactions on Services Computing* 14, 6 (Nov. 2021), 2073–2089. https://doi.org/10.1109/TSC.2019.2897554

[34] Zhengjie Yang, Wei Bao, Dong Yuan, Nguyen H. Tran, and Albert Y. Zomaya. 2022. Federated Learning with Nesterov Accelerated Gradient. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (Dec. 2022), 4863–4873. https://doi.org/10.1109/TPDS.2022.3206480 arXiv:2009.08716 [cs, stat].

[35] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. 268–282. https://doi.org/10.1109/CSF.2018.00027

[36] Samuel Yeom, Irene Giacomelli, Alan Menaged, Matt Fredrikson, and Somesh Jha. 2020. Overfitting, robustness, and malicious algorithms: A study of potential causes of privacy risk in machine learning. *Journal of Computer Security* 28, 1 (Jan. 2020), 35–70. https://doi.org/10.3233/JCS-191362

A PREDICTION DIFFERENCE BETWEEN MEMBERS AND NON-MEMBERS

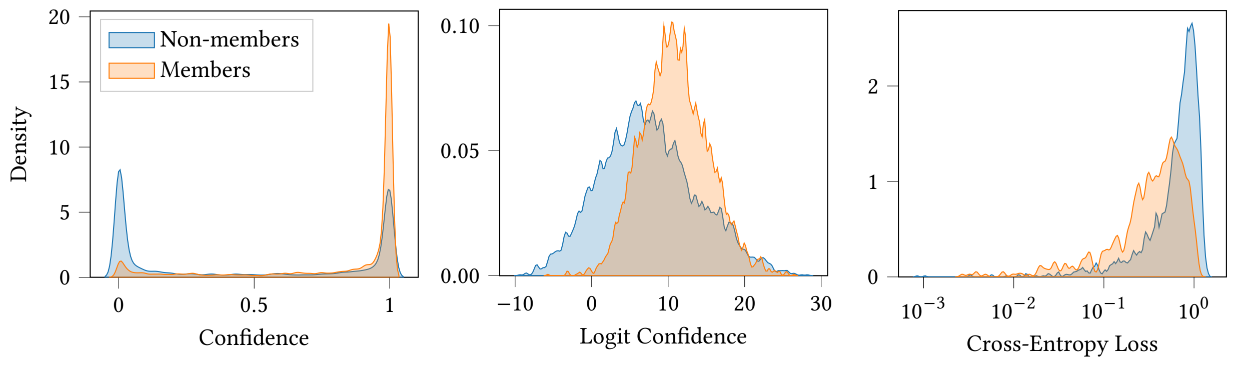
In Figure 7, we plot the difference in confidence, logit confidence and cross-entropy loss for members and non-members for a ResNet18 model trained on CIFAR10 with the FedNAG protocol. With this figure, the reason why membership inference attacks work is easily shown. Overall, this machine learning model is more confident in the prediction for data it has seen before - training data - and its prediction is closer to the target value.

B HYPERPARAMETERS OF THE TARGET MODELS

To simulate an attack on models that are the best option of a federation, we applied grid search across a range of hyperparameters.

Table 10. The spaces in which we grid search for the best hyperparameters. FedNL does not have any hyperparameter, and so no hyperparameter space is applicable.

Protocol	**Hyperparameters**	
FedAdam	η	$\{10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}\}$
	η_g	$\{10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}\}$
FedAvg	η	$\{10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}, 0.2, 10^{-0.5}\}$
FedNAG	η	$\{10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}, 0.2, 10^{-0.5}\}$
	γ	$\{0.85, 0.90, 0.95\}$
FedNL	n/a	



(a) Confidence distribution over all classes (b) Logit confidence distribution over all classes (c) Loss distribution over all classes

Fig. 7. Difference in confidence and loss for members and non-members for FCN trained with FedAVG on Purchase100 dataset.

Table 11. The hyperparameters used for the models trained using FedNL.

Protocol	**Hyperparameters**	
Purchase *Logistic regression*	B	∞
Texas *Logistic regression*	B	∞

Table 12. The hyperparameters used for the models trained using FedAvg.

Protocol	**Hyperparameters**	
CIFAR10	B	64
ResNet18	η	0.2
CIFAR100	B	64
ResNet34	η	$10^{-0.5}$
Purchase	B	64
FCN	η	$10^{-0.5}$
Purchase	B	64
Logistic regression	η	$10^{-2.0}$
Texas	B	64
FCN	η	0.2
Texas	B	64
Logistic regression	η	$10^{-0.5}$

Table 13. The hyperparameters used for the models trained using FedAdam.

Protocol	**Hyperparameters**	
CIFAR10	B	64
ResNet18	η	10^{-2}
	η_g	10^{-1}
	β_1	0.9
	β_2	0.99
	ϵ	10^{-3}
CIFAR100	B	64
ResNet34	η	$10^{-2.5}$
	η_g	10^{-1}
	β_1	0.9
	β_2	0.99
	ϵ	10^{-3}
Purchase	B	64
FCN	η	$10^{-2.5}$
	η_g	$10^{-1.5}$
	β_1	0.9
	β_2	0.99
	ϵ	10^{-3}
Purchase	B	64
Logistic regression	η	10^{-2}
	η_g	$10^{-1.5}$
	β_1	0.9
	β_2	0.99
	ϵ	10^{-3}
Texas	B	64
FCN	η	10^{-2}
	η_g	10^{-2}
	β_1	0.9
	β_2	0.99
	ϵ	10^{-3}
Texas	B	64
Logistic regression	η	$10^{-2.5}$
	η_g	10^{-1}
	β_1	0.9
	β_2	0.99
	ϵ	10^{-3}

In Table 10, we describe across what space we searched for what variable for what protocol. FedNL takes no variables apart from batch size. However, by design, the batch size is intended to take the whole dataset at once. Hence, we do not apply any searching for FedNL. We made no difference in searching between datasets and models. From the results, we selected the configuration that resulted in the highest prediction accuracy on the test data. This search resulted in the hyperparameter configurations described in Tables 11, 12, 13, and 14.

Table 14. The hyperparameters used for the models trained using FedNAG.

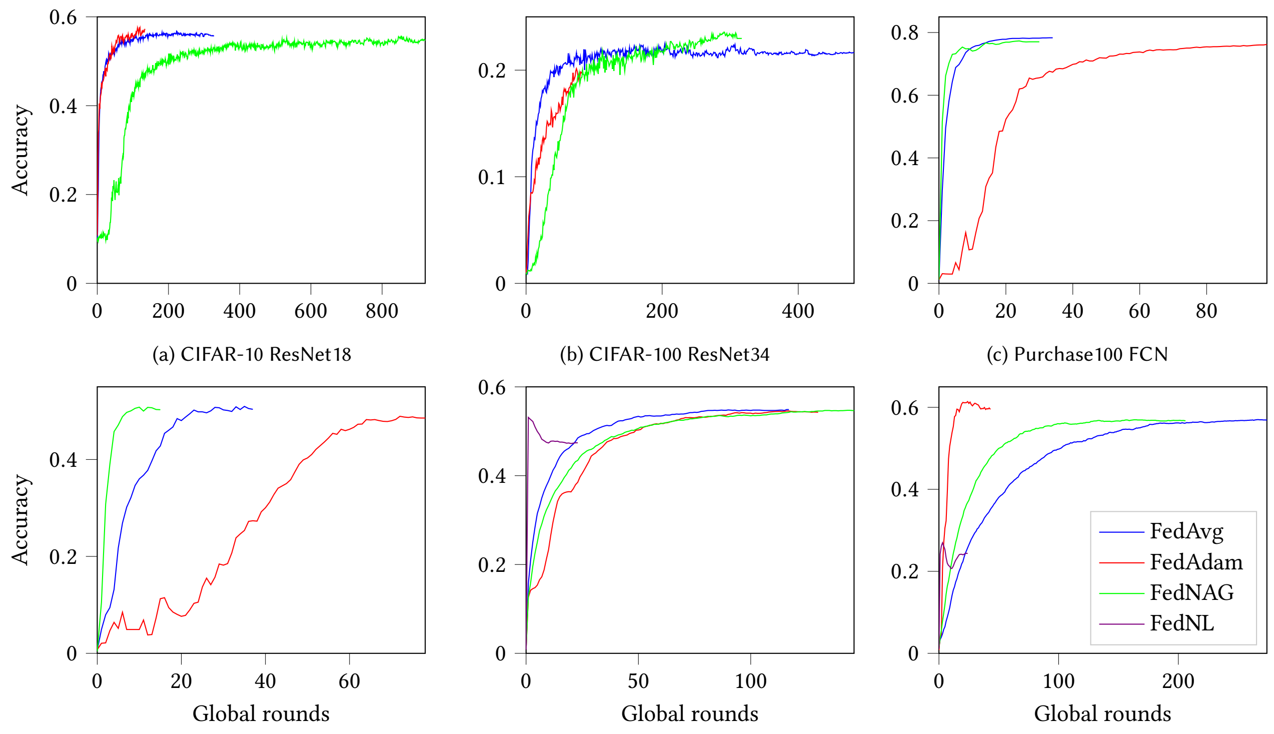
Protocol	**Hyperparameters**	
CIFAR10	B	64
ResNet18	η	$10^{-1.0}$
	γ	0.85
CIFAR100	B	64
ResNet34	η	$10^{-0.5}$
	γ	0.9
Purchase	B	64
FCN	η	$10^{-1.0}$
	γ	0.9
Purchase	B	64
Logistic regression	η	$10^{-2.5}$
	γ	0.85
Texas	B	64
FCN	η	$10^{-1.0}$
	γ	0.9
Texas	B	64
Logistic regression	η	$10^{-2.0}$
	γ	0.95

C LEARNING CURVES OF TARGET MODELS

In Figure 8, we plotted the learning curves for the models we attack. For the simpler Texas100 and Purchase100 dataset, it can be seen that alternatives to FedAvg can find a better or equal solution in less iterations. For CIFAR-10 we see the alternatives take longer to converge. We hypothesize this is because CIFAR-10 contains more complex relations and therefore more local minima in which FedAvg gets stuck but the alternatives do not. This pattern does not hold for CIFAR-100 which is likely due to the overall poor performance of the CIFAR-100 models.

D ATTACK ATTEMPTS

In Tables 15 and 16, we show how many training attempts we made before the attack model prediction was meaningful. The model is perceived as meaningful if the ROC AUC on the training data rounded to 1 decimal was more than 0.5. From this data, we see that the more complex the model gets, the harder it gets to converge to a usable model. We hypothesise this is simply because of an increase in the amount of parameters. The study's goal was not to investigate model stability, so we do not have any further findings on what may be used to let the model converge to something meaningful more quickly.



(a) CIFAR-10 ResNet18 (b) CIFAR-100 ResNet34 (c) Purchase100 FCN

(d) Texas100 FCN (e) Texas100 Logistic Regression (f) Purchase100 Logistic Regression

Fig. 8. Different learning curves of the victim models that are under attack.

Table 15. The number of training attempts before the global attack model resulted in better average prediction than guessing. The data describes a global attack model over FedAdam, FedAvg, FedNAG and FedNL with attack models using different components. Scenario A uses the loss and label components; scenario B uses the loss, label, activation and gradient components; scenario C uses the loss, label and ancillary component; and scenario D uses the loss, label, activation, gradient and ancillary component. Scenarios C and D do not apply to the protocols without ancillary variables. We only applied FedNL on logistic regression models. In italics are the stopped models, as we had a maximum of ten attempts.

	CIFAR-10				**CIFAR-100**				**Purchase100**				**Purchase100**				**Texas100**				**Texas100**			
	ResNet18				**ResNet34**				**FCN**				**Logistic regression**				**FCN**				**Logistic regression**			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
FedAdam	2	6	n/a	n/a	3	3	n/a	n/a	1	2	n/a	n/a	3	2	n/a	n/a	3	1	n/a	n/a	1	3	n/a	n/a
FedAvg	1	3	n/a	n/a	2	2	n/a	n/a	3	1	n/a	n/a	1	2	n/a	n/a	1	1	n/a	n/a	3	3	n/a	n/a
FedNAG	1	4	7	6	2	3	5	6	2	2	4	9	4	3	6	*10*	1	2	4	3	2	1	4	*10*
FedNL	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	2	2	4	5	n/a	n/a	n/a	n/a	1	3	4	2

Table 16. The number of training attempts before the local attack model resulted in better average prediction than guessing. The data describes a global attack model over FedAdam, FedAvg, FedNAG and FedNL with attack models using different components. Scenario A uses the loss and label components; scenario B uses the loss, label, activation and gradient components; scenario C uses the loss, label and ancillary component; and scenario D uses the loss, label, activation, gradient and ancillary component. Scenarios C and D do not apply to the protocols without ancillary variables. We only applied FedNL on logistic regression models. In italics are the models that were stopped, as we had a maximum of ten attempts.

	CIFAR-10				**CIFAR-100**				**Purchase100**				**Purchase100**				**Texas100**				**Texas100**			
	ResNet18				**ResNet34**				**FCN**				**Logistic regression**				**FCN**				**Logistic regression**			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
FedAdam	2	1	n/a	n/a	1	5	n/a	n/a	1	5	n/a	n/a	1	1	n/a	n/a	1	5	n/a	n/a	3	5	n/a	n/a
FedAvg	1	4	n/a	n/a	3	6	n/a	n/a	2	5	n/a	n/a	1	2	n/a	n/a	3	2	n/a	n/a	2	7	n/a	n/a
FedNAG	2	*10*	*10*	10	2	7	5	*10*	1	4	3	6	1	3	4	4	2	1	4	9	4	3	6	7
FedNL	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	3	6	n/a	n/a	n/a	n/a	n/a	n/a	3	5	n/a	n/a