UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE

CONFERENCE OF EUROPEAN STATISTICIANS

Expert meeting on Statistical Data Confidentiality

26–28 September 2023, Wiesbaden

INSIGHTS INTO PRIVACY-PRESERVING FEDERATED MACHINE LEARNING FROM THE PERSPECTIVE OF A NATIONAL STATISTICAL OFFICE

Benjamin Santos, Julian Templeton, Rafik Chemli, Saeid Molladavoudi (Statistics Canada)
Francesco Pugliese, Erika Cerasti, Massimo De Cubellis (ISTAT)
Matjaz Jug (Statistics Netherlands)
Statistics Canada, Statistics Netherlands, ISTAT

benjamin.santos@statcan.gc.ca, julian.templeton@statcan.gc.ca, rafik.chemli@statcan.gc.ca, saeid.molladavoudi@statcan.gc.ca, frpuglie@istat.it, erika.cerasti@istat.it, decubell@istat.it, m.jug@cbs.nl

Abstract

To explore the utility of Federated Learning from the perspective of a National Statistical Office (NSO), this paper highlights the effectiveness of the technique when utilizing it in different ways. Under the United Nations PETLab, researchers from three NSOs expand the work previously done under the UNECE's HLG-MOS group to understand the utility of using Federated Learning to generate statistics from data which cannot be collected (UNECE, 2023). Using a benchmark Human Activity Recognition dataset, the dataset is partitioned into four subsets where a central NSO updates their centralized Machine Learning (ML) model from the updated local models trained by the other NSOs, without anyone being able to view or access the data held by another NSO. Four different aggregation strategies, FedAvg, FedYogi, FedAdagrad, and FedAdam, are tested within the distributed environment to explore how their vanilla implementations compare when evaluating the updated central model. Here, FedAvg performs the best overall with FedYogi performing well, but failing to properly classify some of the classes. Differential Privacy is then applied with varying privacy budgets to explore the impacts of better protecting the models for each aggregation strategy. In general, more privacy results in worse performance, but each aggregation strategy maintains a similar pattern regarding its effectiveness, with FedAvg and FedYogi still performing best in general. The added noise does occasionally improve results with FedAdagrad and FedAdam due to the poorer initial performance. This indicates that Federated Learning is a viable technique for NSOs to consider when data acquisition is impossible but that the choice of aggregation strategy is critical and requires tuning based on the use case. In general, FedAvg and FedYogi are good baseline approaches to be used with Differential Privacy, while selecting appropriate privacy budgets. NSOs can also utilize Homomorphic Encryption to help protect the client models when performing delegated computing at a computational cost relative to the model's complexity.

1 Introduction

National Statistical Offices (NSOs) and other major organizations have a wealth of high-quality data which are used to provide crucial statistics to the general public, guide decisions being made by leaders, and improve business operations. Collection and dissemination of data involves rigorous processes and proper considerations to the privacy of any personal information. This goes beyond an ethical consideration, and for NSOs the safeguarding of this data can be mandated by law, such as by Canada's Statistics Act. To that end, NSOs have been using various methodologies to protect the data gathered and any statistics released. However, data collection can sometimes be impossible depending on the sensitivity and legal protections surrounding the target data. In such cases where the data cannot be collected but certain statistics on the data need to be generated for the good of the public, Federated Learning (FL) is a technique which can be used.

Privacy Enhancing Technologies (PETs) are actively researched technologies which provide new methods to enhancing the privacy of data. PETs can support a variety of use cases such as defense against membership inference and linkage attacks with Differential Privacy (DP) (UNECE, 2023), which adds noise to the data or statistics generated, or by allowing mathematical operations to be performed on encrypted data with the use of Homomorphic Encryption (HE) (UNECE, 2023; UN, 2023). While there are a variety of PETs, FL is one which allows Machine Learning (ML) models to be trained in a distributed setting without the data needing to leave client devices. Using this approach, it would be possible for an NSO or other organization to configure surveys, crowdsourcing campaigns, or collaborations to generate statistics from data which would not be collected. To this end various NSOs have been collaborating within the United Nations PETLab to explore the utility of FL. Initial work performed within the UNECE HLG-MOS IPP group has highlighted that FL can be an effective tool when data cannot be collected (UNECE, 2023). However, there are many aspects surrounding the approach which need to be considered and explored.

Within this work, researchers within Statistics Canada, the Italian National Institute of Statistics (ISTAT), and Statistics Netherlands have collaborated to explore different aspects of FL when tested on a Human Activity Recognition (HAR) dataset. Proper usage of FL can lead to improved statistics on sensitive topics and can help build trust with the populace since data privacy is a key concern. First, this paper will present background information on FL, DP, and HE. Second, an analysis will be performed when comparing four different federated aggregation strategies, with and without using differentially private ML training with varying privacy budgets, which are utilized to combine updates from clients before updating the central ML model. This will exhibit the tradeoff of better protecting a model against attacks versus the general performance of a model, alongside comparing the different aggregation methods. Finally, we test the ability to perform FL with encrypted ML models using HE. This will provide better insights into the use of FL and we will focus discussions from the perspective of its potential use by NSOs.

2 Background Information

Prior to highlighting the experiments performed and discussing the results, we first present a detailed look into FL and the aggregation strategies explored. This will provide a better understanding of what is tested and why these different strategies need to be carefully considered. Brief details on DP and HE are then presented to understand the approaches as they are applied in conjunction with FL.

2.1 Federated Learning Background

FL enables the training of ML models on decentralized data. Instead of transferring data from all the contributing clients (devices or institutions) to a central location to train a model, FL brings the model to where the data is (Konečný et al., 2016). Models are trained by local clients and the weights or gradients are aggregated at a central server, also referred to as the central authority, while preserving data confidentiality. The method of

aggregation can vary and different methods will be discussed later in this paper. Hence only model updates, not raw data, are exchanged between clients and the central server. This is particularly useful in the scenario where an organization, such as an NSO, may want or need to generate statistics from data held by others who are reluctant to share the data or who are legally bound to keep the data on premises. FL works in two main settings, as outlined in Kairouz et al. (2021): cross-silo, which involves large institutions, and cross-device, which involves a large set of edge devices.

The quality of the models generated with FL techniques depends on several factors, such as the quality of distributed data, the number of participating clients, the communication efficiency between clients and the server, and the aggregation strategy used to combine the local model updates into a global model. If the data distribution across clients is representative of the overall population and the aggregation process is well-designed, FL can achieve competitive performance compared to centralized approaches (Nilsson et al., 2018).

In FL, the main objective is to create an ML model with strong overall performance for a given task, without allowing the central authority holding the main model to see or reproduce the original data. One of the issues when working within a distributed setting is regarding the evaluation of the quality of the input data. There may be poisoning attacks by using poor quality or incorrect data (Tolpegin et al., 2020), or client drift (local client models move away from global optimal models) (Varno et al., 2022).

To obtain a robust model when considering some potential issues with the process, there are various aggregation strategies which can be applied within a FL setting. We can classify the aggregation strategies into two main types: client-variance reduction and adaptive global model update. The former focuses on reducing the variance between client models during the global model update process. In this approach, techniques such as weighted averaging are used. The goal is to achieve a better representation of the overall model, considering the particularities of the clients and reducing the impact of divergent client data. The latter is an approach that focuses on adjusting the global model update process based on the performance of the clients. In this case, methods are used to adjust the weights, or contributions of clients, in the aggregation process based on performance evaluation metrics, such as accuracy or loss. This means that better performing or more reliable clients can contribute more significantly to the overall model update than lower performing clients.

Both approaches aim to improve the efficiency and effectiveness of the global model update process in FL. Client-variance reduction focuses on reducing the differences between client models, while adaptive global model update focuses on adjusting the weights of clients based on their performance. These can be used together or separately depending on the specific needs and challenges of the use case.

Traditional ML optimization methods like distributed stochastic gradient descent (SGD) are often unsuitable for FL due to high communication costs. To address this, many federated optimization methods employ local client updates, where clients update their models multiple times before communicating with the server. This approach significantly reduces the communication costs required for training a model. One such method is FedAvg, presented in McMahan et al. (2017), where clients perform multiple epochs of SGD on their local datasets, then send their local models to the server which averages them to create a new global model. Although the FedAvg aggregation strategy has achieved considerable success, recent studies have highlighted convergence issues in certain scenarios.

2.2 Federated Learning Aggregation Strategies

Following the overview on FL, we will now discuss the different aggregation strategies which we will be testing within this paper and discuss some potential issues with them. In this work we focus on the adaptive global model update aggregation strategies and in particular on the following: FedAvg, FedAdam, FedAdagrad, and FedYogi. Note that this paper tests these strategies with their vanilla implementations from the selected Python library, without appropriately tuning the hyperparameters as needed in production (such as the learning rates). The four aggregation strategies are described below.

FedAvg (Federated Averaging) is a FL algorithm that aims to train a global model from the local model updates of multiple clients by calculating the average of the model parameters. This average is calculated across all clients and the aggregated model is used for further training iterations (McMahan et al., 2017).

FedAdagrad (Federated Adaptive Gradient) is a variant algorithm that exploits the adaptive gradient descent method called Adagrad. It adapts the learning rate for each model parameter based on its historical gradients, allowing the model to converge faster and achieve better performance. It is a technique based on averaging the gradients, rather than the parameters straightaway (Reddi et al., 2020).
FedAdam (Federated Adam) is another FL algorithm that combines the advantages of the Adam optimizer with the FL setting. It employs adaptive learning rates and momentum to efficiently update the global model by using the local updates from clients. The gradients computed locally by the devices are aggregated in the central server. This is done by combining the gradients using an aggregation algorithm, such as weighted averaging, to obtain an approximate global gradient (Reddi et al., 2020).
FedYogi (Federated Yogi) is a FL algorithm inspired by the Yogi optimizer. It incorporates elements of both adaptive learning rates and momentum to handle non-convex optimization problems in FL scenarios (Reddi et al., 2020).
Standard federated optimization methods, such as FedAvg, may present convergence issues in some settings. In particular, it exhibits unfavorable behavior in the case of heavy-tailed distributed noise in stochastic gradients. Heaviness of the tails is linked to generalization errors since it hinders the convergence to the optimal solution (Gurbuzbalaban et al., 2021). To control variance in stochastic gradients, adaptive methods have been developed. Adaptive methods are characterized by an adaptive learning rate, which automatically decreases as the algorithm progresses.
The first adaptive algorithm developed was Adagrad. In Adagrad, the learning rate depends on the squared past gradients, which works well with sparse gradients. When gradients are dense and non-convex, however, the rapid decay of the learning rate in Adagrad degrades performance. To overcome such problems, Adam has been developed, where the gradients are scaled down by the square roots of "exponential moving averages" of squared past gradients. In this case, the learning rate is adaptively tuned according to the recent gradients, with past gradients forgotten quickly. Since this method does not work well with sparse gradients, a new algorithm, Yogi, has been developed, where the updates are additive instead of multiplicative (Zaheer et al., 2018).
In a FL setting, the heterogeneity of the data distribution across clients can make a local model drift away from the global solution. Hence, the aggregation at the server side of divergent models degrades the convergence and accuracy of the global model (Nguyen et al., 2022; Reddi et al., 2020). Applying adaptive optimizers on the server side to the average of the clients' model updates can accelerate learning and improve performance.
The federated versions of these adaptive methods are FedAdagrad, FedAdam, and FedYogi. In these algorithms, the server model is updated based on the cumulative history of updates rather than on the average of clients' updates at the current time. Full details on the formulas used by these algorithms are in their respective papers.

2.3 Differential Privacy Background

At a high level, DP is a PET which injects noise into data or statistics such that the results are the same whether any single datapoint within a database is or is not present within the database (Dwork, 2006). There is significant research being done regarding DP and the approach is relevant to the work being done by NSOs. However, within the scope of this paper we will focus solely on its application when training a deep neural network. Here, we explore combining the injection of noise alongside the training of the ML model by a client. Since DP better protects the privacy of the outputs from the ML models, this will provide more confidence to a data holder that the central authority will be unable to reproduce the input data used for training from the outputs of their trained ML model.
DP uses a privacy budget ϵ to determine how much noise to inject into the data. The lower the value of ϵ, the more private the ML model will become. The more privacy that is used, the worse the model will perform overall; thus, it is critical to find a good balance between privacy and utility. Within this work, we test various ϵ values when differentially private training is performed on a client's-side to understand the tradeoff in a distributed environment. While attacks against the model are not explored, lower values of ϵ will provide better protections against data linkage and membership inference attacks (Shokri et al., 2017).

2.4 Homomorphic Encryption Background

HE is a cryptographic technique that allows mathematical operations to be performed on encrypted data. The resulting computations are also encrypted and once they are decrypted the results should be the same as if the operations have been performed on the original unencrypted data (UNECE, 2023; ICO, 2022). HE is a public-key cryptographic scheme with the possible addition of an evaluation key to perform the computations on encrypted data (ICO, 2022).
An important use case for HE is on secure delegated computing, as outlined in UN (2023); Dugdale et al. (2022). In this scenario, one or more parties delegate some computation on sensitive data to a third party, such as a cloud computing provider. Because of the sensitivity of the data and privacy concerns, the delegator does not want the cloud provider to see the data. However, the delegator also may lack computational resources, hence it needs the cloud provider's help with a particular task. Using HE, if the delegator uses private keys to encrypt the data and sends the public key (and evaluation key) to the provider, then the computation can be performed securely on the provider's premise. Once the computation is done, the delegator can retrieve the encrypted result and decrypt it using its private key. An interesting example of training a ML model using HE within the context of an NSO can be found in Zanussi et al. (2021).
Delegated computing with HE is interesting for an NSO where the delegated computations can be done on encrypted data, prohibiting cloud providers from seeing the data. HE can also help improve trust with the public. For FL, specifically, HE can be used to prevent the central authority from peeking at individual models received by the clients (UN, 2023; UNECE, 2023). In particular, federated clients can hold private keys and send encrypted models to the server that performs the aggregation on encrypted models using public and evaluation keys. This prevents the server or aggregator from inspecting the local models sent by the client. Unfortunately, HE does not come without costs, where computation, memory, and communication costs are heavier in this setting.

3 Exploring Federated Learning Aggregation Strategies and Differentially Private Training

With all background information detailed, this section will present the results from experiments conducted which compare and evaluate four different aggregation strategies, with and without DP being applied in the client model training process. This will highlight how well each strategy performs without optimization and how DP affects the results.

3.1 Simulation Definition

To perform the FL simulations, we utilize the Flower library (Beutel et al., 2022). This library provides the ability to quickly create FL experiments with different FL frameworks and can be utilized in practical applications. Flower also provides implementations for each of the aggregation strategies tested: FedAvg, FedAdagrad, FedAdam, and FedYogi. PyTorch has been used to design the Multi-Layer Perceptron (MLP) model which is trained within the experiments due to its simple and pythonic approach. Since PyTorch models are used, the Opacus Python library is utilized to apply DP during the training process when desired (Yousefpour et al., 2022). Opacus provides a simple way for a model, optimizer, and dataloader from PyTorch to utilize specified target ϵ and δ values to perform differentially private training. The three privacy budgets targeted during the simulations with DP are $\epsilon = \{10, 1, 0.3\}$, with a δ value of 1e-5. All simulations performed within this section are run on a Jupyter Notebook on Google Colab.
The dataset that we use for the experiments is a HAR dataset made of 7,352 datapoints and 561 variables. A validation split of 20% is used for evaluation. The simplistic MLP model that has been used is designed to have the inputs go to a first Hidden Layer of 512 neurons, a ReLU activation function, a second Hidden

Layer of 512 neurons, a ReLU activation function, and finally a Softmax output layer with six neurons, which correspond to the six class labels of the HAR dataset. For this dataset the output labels are 'Walking', 'Walking_upstairs', 'Walking_downstairs', 'Sitting', 'Standing', and 'Laying'. Note that the class distribution contains some imbalances such as the class 'Walking_downstairs' having less samples compared to the rest, however the distribution is not significantly skewed.
For all tests, Table 1 below displays the non-default set of hyperparameters used. The number of epochs represents the number of times the entire local dataset of a client/device is iterated through during the local training process. This way, we can control the depth of local training and allow the client/device to make multiple passes over its local data, potentially improving the accuracy of the local model. The number of rounds refers to the total number of times the global model is trained and updated using the aggregated local model updates from all participating clients/devices. In FL with Flower, the training process consists of multiple rounds, where each round involves clients/devices performing local training and contributing their model updates to the central authority for aggregation and the global model update. Therefore, the number of epochs affects the quality of local models, while the number of rounds influences the collaborative training process and the refinement of the global model.

TABLE 1. Simulation Hyperparameter Settings

Hyperparameters	**Settings**
Number of epochs	5
Number of rounds	10
Batch size	32
Learning rate	0.001

To evaluate the performance of the models, we track the accuracies over each round, the f1 macro scores, and the f1-scores per class. Within this paper we will showcase the model accuracies and the f1 macro scores for the tests while making reference to any relevant distinctions found from the f1-scores per class. These will provide an overview of the general performance of the model over the rounds and avoid only considering the general performance of the model given by the accuracy (considering the performance over the classes).

3.2 Simulation Results

This section will display and discuss the results from testing FedAvg, FedAdagrad, FedAdam, and FedYogi with and without differentially private training, using $\epsilon = 10$, $\epsilon = 1$, and $\epsilon = 0.3$. The discussions will highlight the differences between the aggregation strategies with and without DP being used. Figures 1 and 2 showcase the accuracies and f1 macro scores for all aggregation strategies and privacy budgets tested.

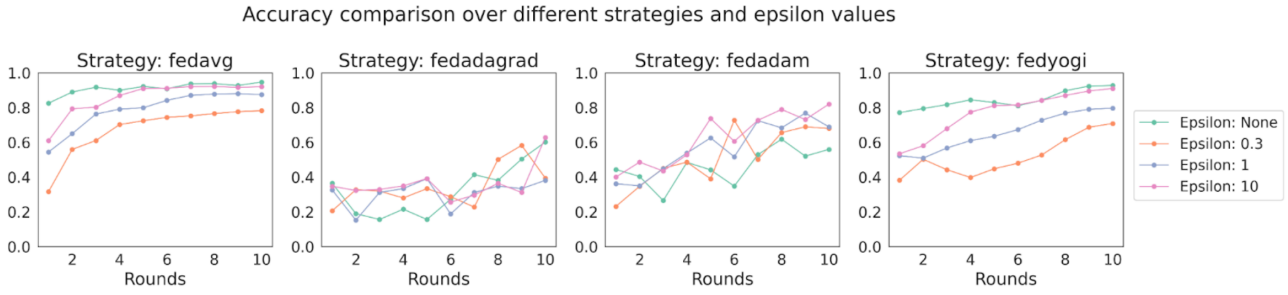

FIGURE 1. Accuracy scores during federated learning training.

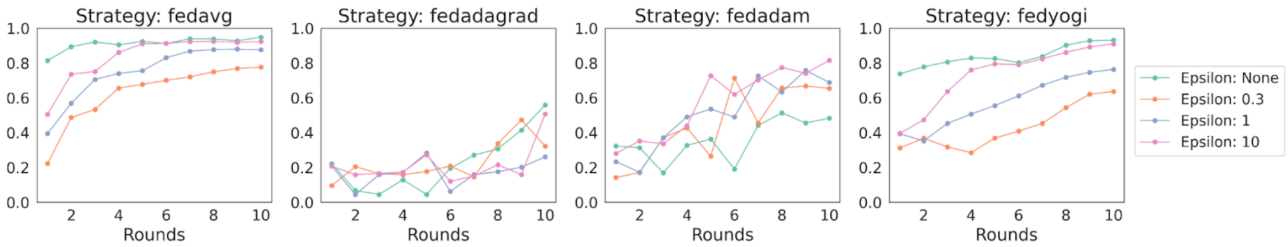

FIGURE 2. F1 macro scores during federated learning training.

First, we will explore the results for each aggregation strategy without DP used (where ϵ is 'None'). From the accuracy curves, it is clear that FedAvg and FedYogi outperform FedAdagrad and FedAdam throughout the entire training process. Both end up having higher overall accuracies and have an immediately strong accuracy result, rather than starting with poor performance. Overall, FedAvg ends with a higher accuracy and achieves better results quicker than the rest. FedAdagrad performs the worst overall but ends up with a similar accuracy compared to FedAdam. FedAdagrad also is increasing consistently at the end, which indicates that more training time can improve the performance, but that it is slower to reach strong results with this configuration. FedAdam also has more fluctuation than the other options as the accuracy frequently decreases and increases.

When considering the f1 macro scores, the general performance is the same as what is observed with the accuracies, but the overall scores are worse. Specifically, the FedAdagrad and FedAdam f1-scores per class appear to drastically fluctuate for each class label, with only a subset being properly learned, whereas FedAvg and FedYogi do much better in general. FedAvg does consistently well for all six classes, but FedYogi is slow to learn some of the classes, resulting in lower overall scores. In general, without using differentially private training, FedAvg is performing the best with FedYogi doing well. Using FedAdagrad and FedAdam without properly tuning their hyperparameters gives poor results, thus experiments can consider FedAvg or FedYogi as a strong baseline aggregation strategy for the initial evaluation of a FL implementation. FedAvg is a simplistic approach making it a good first option to quickly test.

While FedAvg and FedYogi perform best in general for these tests, it is important to understand how they perform when trained while using DP. This better protects the resulting model and helps alleviate certain risks when the privacy budget ϵ is sufficiently small. Within these tests a higher ϵ value of 10 is utilized to understand the impact of injecting little privacy into the model. An ϵ value of 1 is used to then observe the impact when a moderate amount of privacy is injected. Finally, an ϵ of 0.3 is tested to use significantly more privacy during the training, resulting in a model which is better defended from attempts to reverse engineer the inputs from the outputs. The resulting f1 macro scores and accuracies exhibit that the impacts of DP change depending on the robustness of the aggregation strategy which is used.

FedAvg and FedYogi, which have strong overall performance, exhibit results which are expected. As the privacy budget is increased, the overall performance is reduced. For both strategies, the results for $\epsilon = 10$ almost converge to the normal outputs since little privacy is added. The performance diminishes as more privacy is injected. This showcases the logical tradeoff between adding more privacy and reducing the utility of the trained model. Observing the f1-scores per class exhibits that a small number of classes struggle to be quickly learned when ϵ is smaller, which causes the reduced overall results. While most classes slowly become learned, some strategies struggle to learn the 'Walking_downstairs' class with reduced ϵ values. Given more rounds, the overall results may increase slightly as the poor performing class predictions improve.

FedAdagrad and FedAdam showcase different behaviors in the results when trained with DP. Rather than the performance consistently decreasing, there are instances where DP helps improve the performance. For FedAdam specifically, this is observed in each test whereas the base implementation of FedAdagrad is only slightly worse than when a low privacy budget is used. This can be attributed to the poor baseline performance of these untuned methods for this experiment. Since the initial performance is low, the noise being added by DP may end up helping, especially for FedAdam. In practice however, a better general result is preferred since this

performance increase is not be guaranteed. While FedAdam has some higher results than FedYogi when using DP, the variance within the outputs may cause unpredictability compared to FedAvg and FedYogi. Thus, while other strategies may improve performance over time while using DP, FedAvg performs the strongest overall in this simulation. FedYogi also can perform well with noise in the training, but takes longer to learn from the noisy values. In different use cases, the issues with FedAvg may become more relevant than what is observed from these tests, making FedYogi still a good option to consider.
Overall FedAvg and FedYogi perform the best out of the four strategies when only using the initial implementations within this scenario. While this can change depending on the use case and how tuning is done, the results indicate that using either FedAvg or FedYogi can provide strong results without additional tuning. In scenarios where the simplicity of the implementation is important, such as when HE is applied, FedAvg is a strong option to quickly derive simulation results. For production implementations, more thorough testing should be done with the strategies, but FedAvg and FedYogi will be useful when determining the effectiveness of a FL approach.

4 Encrypted Federated Learning

To test HE in a FL setting we consider three different scenarios: (1) encrypting the last linear layer of the model, (2) encrypting the last two, and (3) encrypting the last three layers (note that we leave the activation layers on the clear). Each NSO holds a common set of private and public keys using the Paillier cryptosystem, where more details on the cryptosystem can be found in Paillier (1999). This public key cryptosystem is homomorphically additive, thus, using Paillier HE we can compute FedAVG on the encrypted and cleartext layers (UNECE, 2023). To this end, the server or aggregator only holds the public key and can easily compute the average of the weights without knowing anything about the encrypted layers. At the end of each round of training, each NSO can decrypt the global (averaged) model using the common private key. To handle the communication in FL, we must serialize plaintexts and ciphertexts before sending them and deserialize them at the end points.

TABLE 2. Homomorphic Encryption in Federated Learning Simulation Results

Strategy	Relative training time	Relative RAM	Relative model size	Encryption-serialization time	Decryption-deseialization time
FedAvg	1	1	1	1	1
eFedAvg (1)	1.04	1	8.78	23	631
eFedAvg (2)	597	1.08	4001	1237	2532

In table 2 we present cases 1 and 2 where we use the vanilla strategy FedAvg from McMahan et al. (2017) for comparison. We present the relative results of homomorphically encrypted FedAvg (eFedAvg) against FedAvg. It is important to remark that the evaluation metrics in all cases are essentially the same, as we expect from the homomorphic property of HE schemes. On the other hand, we can see how the models expand in size, which reflects the relative increase of RAM use. We also observe increases to the time needed for the training, encryption-serialization, and decryption-deserialization. Due to the amount of RAM available in the instance used for training, we did not perform a full round of FL training with HE for fully encrypted models in these tests. However, we can report a model expansion of 350,000 times that of an unencrypted model. The encryption-serialization and decryption-deserialization steps take 84,000 times and 155,000 times longer respectively, which is a drastic increase. Lastly, we have also compared case 1 against another HE scheme called CKKS (Cheon et al., 2017) and have found that with CKKS, the ciphertext are 115,000 times larger, however the encryption and decryption steps are faster (11,000 times and 50,000 times faster respectively).
Model expansion, encryption, serialization, decryption, and deserialization are key factors to consider when choosing an HE scheme and what to encrypt (whether it is the partial or the full model). These can affect communication costs within FL and must be appropriately considered. A privacy assessment should be conducted when using HE with FL to determine how much of a model must be encrypted to ensure that sensitive information is not leaked.

5 Discussions and Conclusions

The results from this paper have highlighted that FedAvg and FedYogi are strong performing aggregation strategies when applied in a baseline FL setting with the selected HAR dataset. While the use of DP with these strategies reduces their effectiveness, the end model is better protected against attacks which is important when building trust with the clients participating in the training. For NSOs exploring the possibility of using FL, it is important to carefully test and consider the available aggregation strategies while ensuring that they are tuned appropriately. That said, naively choosing FedAvg or FedYogi can still produce strong results and are good strategies to quickly test a FL environment. FedAvg's simplicity also makes it easier to use alongside HE. Outside of these simulations, increasing the number of epochs per round can also better support strategies such as FedYogi, which can benefit from the additional local training on a client's device.

Traditional ML considerations, such as class imbalance, should still be appropriately considered when designing the FL environment. The evaluation metrics used to evaluate the models after a round of training should be selected to match the expected class imbalance (if possible). When a poor round of training occurs, the model updates can be ignored or tracked separately in case there is a shift from the client data. HE can better help protect the data when delegated computing is needed to ensure that no others can view or infer information from the weights or gradients being passed. This comes at a computational and memory cost relative to the size of the model being encrypted and aggregation strategy used. While FL alone does not alleviate all potential privacy issues, such as attacking a client's local model to infer information, using DP and/or HE can alleviate some of these shortcomings as required by the use case.

To conclude, we have explored the use of four FL aggregation strategies. Each of these strategies have been evaluated with and without differentially private training of varying privacy budgets. Homomorphically encrypting varying layers of model weights has also been tested. The results indicate the FL is a viable approach which can be considered by NSOs or other organizations when data acquisition is impossible. The trained models can be appropriately supported through other PETs, where certain aggregation strategies, such as FedAvg and FedYogi, can be used without being optimized to test the FL implementation. While more explorations into these approaches can be made, it is clear that these options make the ability to work with sensitive or legally protected data possible through open and privacy-focused collaborations. By being clear and transparent with FL, these collaborations may result in more robust statistics being available to the populace and leaders for specific topics.

References

Beutel, D. J., T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane (2022). Flower: A friendly federated learning research framework.

Cheon, J., A. Kim, M. Kim, and Y. Song (2017). Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology-ASIACRYPT 2017*, pp. 409–437. Springer.

Dugdale, C., S. Molladavoudi, B. Santos, and J. Templeton (2022). Privacy enhancing technologies at statistics canada. In *Proceedings of the Annual Meeting, Statistical Society of Canada*.

Dwork, C. (2006). Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener (Eds.), *Automata, Languages and Programming*, Berlin, Heidelberg, pp. 1–12. Springer Berlin Heidelberg.

Gurbuzbalaban, M., U. Simsekli, and L. Zhu (2021). The heavy-tail phenomenon in sgd. In *International Conference on Machine Learning*. PMLR.

ICO (2022). Privacy-enhancing technologies (pets). https://ico.org.uk/media/about-the-ico/consultations/4021464/chapter-5-anonymisation-pets.pdf.

Kairouz, P., H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning 14*(1–2), 1–210.

Konečný, J., H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon (2016). Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*.

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR.

Nguyen, H., L. Phan, H. Warrier, and Y. Gupta (2022). Federated learning for non-iid data via client variance reduction and adaptive server update. *arXiv preprint arXiv:2207.08391*.

Nilsson, A., S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand (2018). A performance evaluation of federated learning algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, DIDL '18, New York, NY, USA, pp. 1–8. Association for Computing Machinery.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology -EUROCRYPT '99*. Springer.

Reddi, S., Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan (2020). Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Shokri, R., M. Stronati, C. Song, and V. Shmatikov (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE.

Tolpegin, V., S. Truex, M. E. Gursoy, and L. Liu (2020). Data poisoning attacks against federated learning systems. In L. Chen, N. Li, K. Liang, and S. Schneider (Eds.), *Computer Security – ESORICS 2020*, Cham, pp. 480–501. Springer International Publishing.

UN (2023). United nations guide on privacy-enhancing technologies for official statistics: case study 15. Technical report, United Nations Committee of Experts on Big Data and Data Science for Official Statistics, New York.

UNECE (2023). Input-privacy preservation report. Technical report, High-Level Group for the Modernisation of Official Statistics, Brussels, Belgium.

Varno, F., M. Saghayi, L. Rafiee Sevyeri, S. Gupta, S. Matwin, and M. Havaei (2022). Adabest: Minimizing client drift in federated learning via adaptive bias estimation. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner (Eds.), *Computer Vision – ECCV 2022*, Cham, pp. 710–726. Springer Nature Switzerland.

Yousefpour, A., I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov (2022). Opacus: User-friendly differential privacy library in pytorch.

Zaheer, M., S. Reddi, D. Sachan, S. Kale, and S. Kumar (2018). Adaptive methods for nonconvex optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 31. Curran Associates, Inc.

Zanussi, Z., B. Santos, and S. Molladavoudi (2021). Supervised text classification with leveled homomorphic encryption. In *Proceedings 63rd ISI World Statistics Congress*, Volume 11, pp. 16.