ENHANCING THE PRIVACY OF FEDERATED LEARNING THROUGH DATA SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) is a distributed machine learning architecture where edge devices collaboratively learn the shared model, while the training data is securely held at the edge devices. FL promises a way forward to preserving data privacy by sending model updates in the form of gradients or the weights themselves. However, these updates still contain the essence of the original training data and can be reconstructed using gradient-based attacks. To overcome this, we propose a novel Privacy Preserving Federated Learning algorithm (*PPFed*) wherein we generate a condensed dataset from the original training data at each edge device. The client's then train their local models on the condensed dataset which is then broadcasted to the server, followed by regular federated averaging. Our method provides privacy by being robust against gradient-based attacks, which holds across different benchmark datasets and CNN based architectures.

1 INTRODUCTION

Federated Learning (FL) introduced by McMahan et al. (2017) is a distributed machine learning paradigm where a shared Global Model (GM) is collaboratively learned by centrally aggregating Local Models (LM) which are trained on edge devices using their private data. Sharing the local training data with a central server can lead to data privacy concerns. To avoid these concerns, this distributed learning framework was developed where the trained local models are shared instead of training data. Further, the local models can be of smaller size than the local data, and sending the client models to server also results in efficient bandwidth utilization. Lastly, edge devices are becoming more computationally capable to enable local training, with GPU accelerated devices like Nvidia Jetson being used in self-driving cars Zhang et al. (2021) and urban camera networks .

There are several practical and useful applications of FL such as next keyword prediction in smart phones without revealing the keywords and phrases typed by users to a central service, Hard et al. (2018). FL also benefits machine learning in healthcare where local models trained on patient data that are retained within individual hospitals such as their MRI scans or blood samples, can be aggregated into a more effective global model spanning hospitals for improving diagnostics Rieke et al. (2020), Li et al. (2020a). Another promising application of FL is in an autonomous vehicle network Zhang et al. (2021). Here, large data samples collected by the on-board sensors of self-driving vehicles can be used to train local models, which are often smaller in size than the local training data. These compact local models can then be sent to cloud severs for central aggregation, thus requiring lesser network bandwidth.

In the FL framework, clients independently train their models and share the gradients or the model updates as they cannot share their data due to privacy concerns. One way to solve this problem is FedSGDMcMahan et al. (2017), where the clients update the computed gradients or model updates after every optimization step to the server, but this causes high communication cost. Federated Averaging (FedAvg) McMahan et al. (2017) overcomes this drawback by allowing multiple local updates before sending the models to server. FedAvg has faster convergence when the data is iid distributed across the clients but it has slower convergence when the data is non-iid distributed. Most of the recent works on FL focus on improving the accuracy of the model under data heterogeneity Karimireddy et al. (2020); Acar et al. (2021); Li et al. (2020b). However, they all require communicating the gradients or the model parameters with the server.

A common assumption in FL is that by sharing only the local weights or the gradients privacy can be preserved. However, recent works Huang et al. (2021); Zhu et al. (2019); Zhao et al. (2020) show that if the architecture and the gradients of the model are known, then the inputs to the model can be reconstructed. Thus, as the server has the knowledge of both the models and the gradients, it can potentially use these gradients and reconstruct the client's local data.

These findings can pose a threat to the local data privacy that is assumed in FL. To overcome this limitation, we propose a novel method *Privacy Preserving Federated Learning (PPFed)* where we perform federated learning on a small synthetically generated dataset (condensed data (CD)), rather than the original training dataset. These compact and representative samples are generated locally at the client using a gradient-matching technique for condensation, and subsequently used to train the local model. We demonstrate empirically that gradient-based attacks on our proposed method obtains poor reconstruction performance (based on MSE and perceptually) when compared to the FedAvg, thus enhancing the privacy of the original local training data. However, the computational cost/training time of the proposed method is higher than simple FedAvg due to the overheads of generating the condensed training samples. These claims are validated through experimental results obtained by training several common Deep Neural Networks (DNN) models using Nvidia Jetson AGX GPU-accelerated edge clients. We also evaluate our performance across several datasets and report accuracy, loss, MSE along with the visualization of reconstructed data.

Our key contributions are as follows.

- We propose a novel FL method PPFed, where we perform federated learning using the condensed data rather than the original training data.
- We empirically show that with Convolutional Neural Network (CNN) architectures, PPFed is robust to gradient-based attacks on the server when compared to FedAvg and other algorithms that send the local models trained on the original data.
- We also compare PPFed with an alternative DOSFL+ method where all clients generate condensed data and send them to the server, which centrally trains a global model over the aggregated condensed data in a FL single round. We analyze the accuracy vs. privacy trade-offs of these approaches.

2 Related Work

Dataset condensation is the method to create the synthetic sample set of size much lesser than the original training data. This is explored in the works of Zhao et al. (2021) and Zhou et al. (2020).

FL in research is very rapidly progressing, we only present the works that are closely related to our work. For more detailed introduction to FL one can refer Kairouz et al. (2021). Since FL is introduced by McMahan et al. (2017) most of the FL literature was based on improving the performance of the global model under data heterogeneity.

This includes regularization based approaches such as Karimireddy et al. (2020), Li et al. (2020b), Acar et al. (2021) and the data generation approaches such as Zhang et al. (2022). Distillation based strategies are also used for model aggregation in Lin et al. (2020). In the works of Mishchenko et al. (2019) Dutta et al. (2020), the communication cost is reduced by compressing the model parameters/gradients.

Condensed data in the context of FL was first proposed by Zhou et al. (2020) as a means to minimize the communication cost and train the global model in one communication round. The authors proposed to send the condensed data to the server in one shot as the emphasis is more on improving communication efficiency.

In Geiping et al. (2020) authors consider for the first time the issue of privacy in FL when subjected to gradient-based attacks such as Zhu et al. (2019). In the work, Huang et al. (2021) authors propose defenses against the gradient inversion attacks, some of these requiring the use of batch normalization and dependant on training parameters such as using a larger batch size.

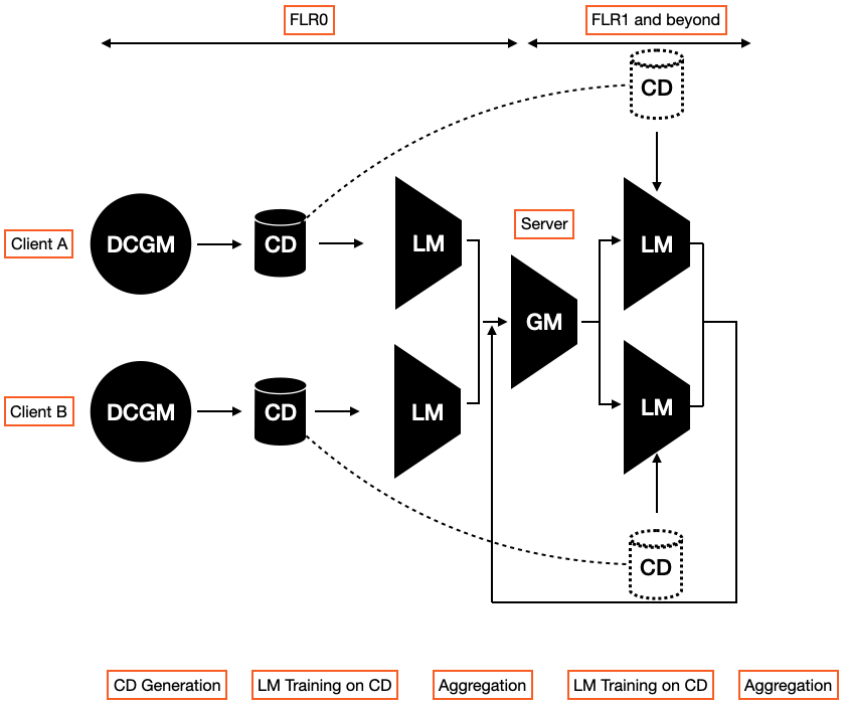


Figure 1: Architecture of the proposed PPFed method. We first generate the *Condensed Data (CD)* in each client using the Dataset Condensation with Gradient Matching (DCGM) algorithm. This happens just once before Fl round 1 (FLR1). Then, we train a *Local Model (LM)* in each client over the CD. The LMs are shared with the server, which aggregates the model parameters to create a *Global Model (GM)* and broadcasts it back to the clients. In future rounds, the clients start with the GM and train over their CD to generate a new LM, and so on.

3 Proposed Method

We first give a brief overview of the dataset condensation algorithm which is used to distill the original training data to generate a compressed version of it in the form of fewer samples. Subsequently, we discuss its use in our proposed PPFed approach.

3.1 Dataset Condensation

Dataset condensation learns to condense large datasets into a small set of informative synthetic samples for training DNNs from scratch. The main objective while crafting the condensed data is to obtain models trained on the condensed data that can yield generalization performance comparable to the models trained on the true data.

If $\tilde{w}$ and w denote the trained model parameters on true and condensed data respectively. The condensed data generated should satisfy the following:

$$\mathbb{E}_{(x,y)\in D} l(\tilde{w}, (x, y)) = \mathbb{E}_{(x,y)\in D} l(w, (x, y)) \tag{1}$$

where $l(w, (x, y))$ denotes the task specific loss function. In this work we consider the classification task and the loss as cross entropy loss function, where D denotes the data generating distribution. The above goal is realized by various dataset condensation methods such as Dataset Condensation with Gradient Matching (DCGM) by Zhao et al. (2021) and dataset distillation by Wang et al. (2018). We employ DCGM in our design due to its better performance.

3.2 PRIVACY PRESERVING FEDERATED LEARNING (PPFED)

In federated learning we optimize the following cost function:

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{2}$$

where w denotes the model parameters and n denotes the number of clients participating in the FL. $f_i(w)$ is the optimization objective of the client i and is given by Eq. 3:

$$f_i(w) = \mathop{\mathbb{E}}_{(x,y)\in\tilde{D}_i} l(w, (x, y)) \tag{3}$$

where $\tilde{D}_i$ is the condensed data that is generated from the true data D_i, and (x, y) are condensed training samples where x is the input and y the corresponding class label. Here, l is the loss function, which can be cross entropy loss function. Models trained on the true data and condensed data should give the same generalization performance as per Eq. 1. Hence we replace D with $\tilde{D}$ and use Eq. 3 to train client models locally.

Towards this goal, we split our PPFed algorithm into two stages (Figure 1): Stage 1 to generate Condensed Dataset (CD) using DCGM algorithm Zhao et al. (2021) and Stage 2 to perform multiple rounds of FedAvg. The two stages of PPFed can be de-coupled from each other, i.e., the generation of CD is independent of the FedAvg stage. The generated condensed data is reused across multiple rounds of FL or even for training a different model in future as storing the condensed data is memory efficient.

Each client i generates the CD ($\tilde{D}_i$) using the original training dataset D_i only once. Clients then train the individual local models (LM) on the CD, over multiple FL rounds. The gradients or the parameters of the trained LM from each client is sent to the server, which then aggregates the local models into a global model (GM) using FedAvg. While other methods for global aggregation can be used, we use the FedAvg for its simplicity and it offers a reasonable baseline for comparison. The GM is sent to all the clients to start the next round of FL. Then, the clients start with GM as their initial model and train a new LM using the CD generated earlier. The updated LMs are sent by the client to the server, and this process repeats over multiple rounds till convergence.

The accuracy of the PPFed method depends on the quality of the condensed dataset. If we need better accuracy we can generate a better or larger set of condensed dataset. This, however, can take a longer computational time. This induces a training time vs. accuracy trade-off. We analyze this aspect in detail in the experiments section. Our approach is also shown in detail in the Algorithm 1.

4 EXPERIMENTAL SETUP

4.1 EXTRACTING GRADIENTS IN FL SETUP

Gradient-based attack was first proposed by Zhu et al. (2019), and later extended to FL by Geiping et al. (2020). We present a brief overview of such an FL attack, which we use to evaluate the privacy of the FL approaches we consider. For more details refer Geiping et al. (2020).

Gradients-based attacks need information about the model architecture and the gradients of the parameters, and the input data samples are reconstructed by matching the gradients, as summarized below:

$$x^* = \arg\min_x l_g(\nabla f_l(w), \nabla f(w, x)) \tag{4}$$

where $\nabla f_l(w)$ denotes the leaked gradients and $\nabla f(w, x)$ is the gradient of the model which is matched with the leaked gradient by optimizing the x.

$$w_i(t+1) = w(t) - \alpha * \sum_{k=0}^{n_g} \nabla f_i(w_i^k(t)) \tag{5}$$

Algorithm 1 PPFed

Stage 1:

for FL Round $t = \{0\}$ **do**
  Server broadcasts the weights $w(t)$
  Each client $i \in n$ executes DCGM Zhao et al. (2021) to generate condensed dataset $\tilde{D}_i$ using the original training data present locally (D_i) in an i.i.d. manner.
  $w_i(t)$ = CLIENTUPDATE($w(t)$) $\forall i \in n$
  Server averages LMs $w_i(t)$ generate by the clients to generate GM $w(t+1) = \frac{1}{n}\sum_{i=0}^{n} w_i(t)$
end for

Stage 2:

for FL Round $t = \{1, \ldots, T-1\}$ **do**
  Server broadcasts the weights $w(t)$
  $w_i(t)$ = CLIENTUPDATE($w(t)$) $\forall i \in n$
  Server averages LMs $w_i(t)$ generate by the clients to generate GM $w(t+1) = \frac{1}{n}\sum_{i=0}^{n} w_i(t)$
end for

function CLIENTUPDATE($w(t)$)
  Set $w_i(t) = w(t)$
  for Epoch $e = \{0, 1, \ldots, E\}$ **do**
    for Batch $b = \{0, 1, \ldots, B\}$ **do**
      $w_i(t) = w_i(t) - \alpha \times \nabla f_i(w_i(t))$ where $(x, y) \in \tilde{D}_i$ Eq. 2
    end for
  end for
end function

Eq.5, shows the client model update $w_i(t+1)$ after n_g, the number of gradient updates at the client i, and $w_i^0(t) = w(t)$. $w(t)$ is the global model shared by server, here we are assuming that the client is using stochastic gradient descent update rule to perform the optimization.

We consider the honest but curious server scenario where server can reconstruct the client data using the parameters shared as per FL protocol. The server has knowledge of the model architecture being trained and gradients for the model updates are shared by the clients. Using the model weights that are shared, the server can subtract the initial model weights (which it broadcasted to all clients) from the updated client weights that it receives after a round of local training to get the accumulated gradient. Let g_i denote the accumulated gradient of the client i, $w(t)$ represent the global model at time t, and $w_i(t+1)$ denote the updated model of client i. Then the server can compute the accumulated gradient up to a scale as follows:

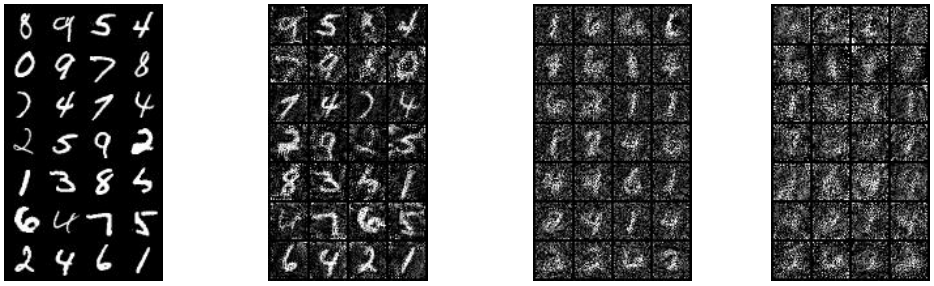
$$g_i \propto w(t) - w_i(t+1) \tag{6}$$

So the FL server can easily compute the accumulated gradients by subtracting the weights up to the scale of the learning rate. This difference can then be used to reconstruct the images using any of the gradient inversion methods Zhu et al. (2019); Geiping et al. (2020).

We observe that attacks based on DLG algorithm Zhu et al. (2019) provide better reconstruction quality and hence use on it. We consider that clients perform the update on a batch of size 4 and do a single gradient descent update. Even in this scenario, we show PPFed can preserve the privacy of the model when such attacks are employed by the server.

4.2 HARDWARE AND SOFTWARE PLATFORM

We conduct our experiments by running clients on Nvidia Jetson Xavier AGX, which has a 512-core Volta GPU, an 8-core ARM CPU and 32GB of LPDDR4x memory shared between CPU and GPU. The server runs on an Nvidia RTX 3090 GPU workstation. The client and server are connected over a 1 Gbps Ethernet link. We implement our algorithm using *Flower* API and PyTorch DL framework. Each client runs as a process on a separate AGX device and the server process runs on a single 3090 workstation.



(a) True Images (b) Reconstructed Images (FedAvg) (c) CD Images (DOSFL+) (d) Reconstructed CD Images (PPFed)

Figure 2: Images reconstructed using the gradients leaked from FedAvg and PPFed for MNIST dataset. DOSFL+ shares CD images with server and hence reveals it.

4.3 Models and Datasets

We train the following DNN models using the FL setup: 1) **ConvNet**: The specific DNN we use includes 3 blocks, each with 128 filters, followed by InstanceNorm, ReLU and AvgPooling modules. A linear classifier follows the last block. 2) **LeNet**: The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers. 3) **Multi-Layer Perceptron (MLP)**: This is a simple neural network containing just an input, a hidden and an output layer. For more details refer to Zhao et al. (2021). We use the CIFAR10, MNIST and FashionMNIST datasets along with ConvNet, Lenet and MLP, respectively.

4.4 Baseline Algorithms

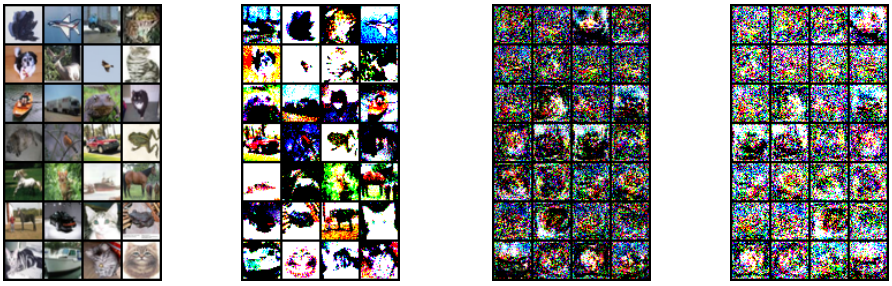
We compare our proposed PPFed method against the FedAvg baeline approach and a variation of the DOSFL by Zhou et al. (2020). DOSFL is a one shot FL learning scheme where the condensed data is shared with server to train the global model in a single round. We extend DOSFL by using the latest DCGM dataset condensation method instead of the one reported in their paper, and refer to this as DOSFL+.

4.5 Testing Methodology

We report the test accuracy of algorithms for the model and dataset combinations in Table 2. We use two FL experiment setups: with 2 clients and with 5 clients. In our setting we make the full device participation in every FL round. The training dataset is partitioned i.i.d fashion across these clients and we consider the balanced setting where every client gets same quantity of data.

4.6 Hyper-parameters

We use a learning rate of 0.001 with Stochastic Gradient Descent (SGD) algorithm with a momentum of 0.9. For generating the condensed dataset, we follow the hyper-parameters as specified in the DCGM paper by Zhao et al. (2021). We generate 50 images per class by varying the number of iterations in the DCGM algorithm as $\{10,30,50,70,100\}$. The more the number of iterations of DCGM, the better the quality of the condensed data is supposed to be, but also longer the time taken to generate the CD. This helps us analyze and compare the time taken (including CD generation and training time) vs the accuracy achieved. When we compare against FedAvg and DOSFL+ we use the CD generated from 100 iterations of DCGM. Clients use a batch size of 256 while training their local models. The local models are trained for single epoch for FedAvg due to its larger dataset size and for 20 epochs for PPFed due to the smaller number of samples.



(a) True Images (b) Reconstructed Images (FedAvg) (c) CD Images (DOSFL+) (d) Reconstructed CD Images (PPFed)

Figure 3: Images reconstructed using the gradients leaked from FedAvg and PPFed using CIFAR10

(a) True Images (b) Reconstructed Images (FedAvg) (c) CD Images (DOSFL+) (d) Reconstructed CD Images (PPFed)

Figure 4: Images reconstructed using the gradients leaked from FedAvg and using FashionMNIST

5 Results and Discussion

5.1 Accuracy vs Privacy Trade off

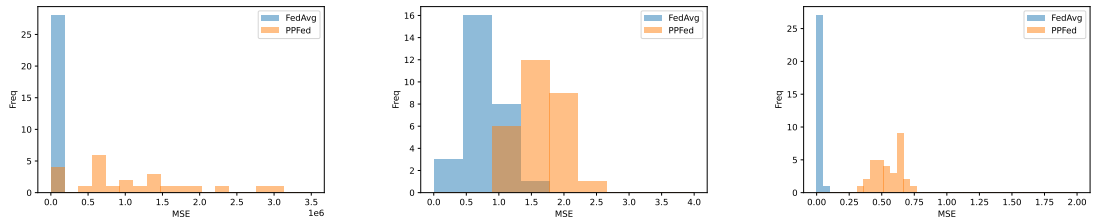
In Figure 2 we show the quality of images reconstructed using the Gradient in version attacks. Figure 2a shows the true inputs on which the gradients are leaked while Figure 2b shows the reconstructed true images. Similarly Figures 2c and 2d display the condensed images and the reconstructed condensed images. We can see that the image reconstruction quality is poor for Figure 2d compared to the true images in Figure 2a when using the gradients of the LeNet model trained on the condensed data, as PPFed does. The digits are unrecognizable. In contrast, the images in Figure 2b reconstructed using the gradients of the model trained using the true images, such as FedAvg, are fairly close to the true images and the digits are recognizable. In methods such as DOSFL+, the condensed data (Figure 2c) is transmitted to the server, and thereby revealed. We see that the condensed data in itself does not provide privacy and adequately mask the original data. This subjective evaluation demonstrates that PPFed has better privacy when compared to FedAvg and DOSFL+ in mitigating image reconstructability.

Similar benefits are seen when training ConvNet on CIFAR10 data in Figure 3. However, this distinction is diminished when training MLP on the Fashion MNIST datasets, reported in Figure 4. Gradient inversion attacks are able to easily reconstruct the data on the fully connected layers of MLP Geiping et al. (2020). So on MLP architectures, we conclude that privacy cannot be preserved even using our PPFed approach, while we achieve good privacy preservation in CNN architectures.

In Figure 5 we compute the histogram of the Mean Squared Error between the reconstructed image and the closest image belonging to that class. In the Table 1 we present the mean and standard deviations of the distribution of MSE. In all the three cases we can see that mean value of MSE is higher for PPFed compared to FedAvg.

Table 1: Mean and Standard deviation (Std) of the Mean Squared Error (MSE) between the reconstructed image and the *closest* true image.

Model + Dataset	Mean (PPFed)	Mean (FedAvg)	Std (PPFed)	Std (FedAvg)
ConvNet+ CIFAR10	10^6	873.8	8×10^5	1021.8
LeNet+MNIST	1.65	0.77	0.32	0.29
MLP + FMNIST	0.54	0.006	0.014	0.099

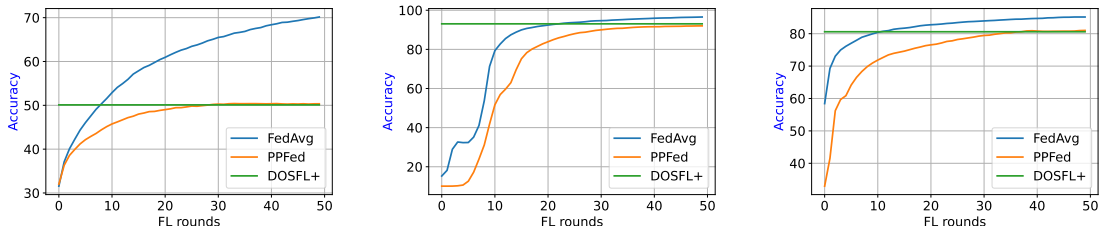


(a) ConvNet + CIFAR10 (b) LeNet + MNIST (c) MLP+FMNIST

Figure 5: Histogram of MSE of the reconstructed images from the gradients of FedAvg and PPFed

The privacy benefit of PPFed does comes with the accuracy downsides, as can be seen from the Figures 6, which show the convergence plots of PPFed, FedAvg and DOSFL+ algorithms. It is clear that FedAvg performs best in terms of accuracy, while DOSFL+ and PPFed are close to each other at the end of convergence. DOSFL+ has constant performance since it only performs one FL round of communication to attain convergence while FedAvg and PPFed improve with the rounds.

The Table 2 compares the accuracy of the PPFed, FedAvg and DOSFL+ algorithms. It can be seen that PPFed performs identical to the DOSFL+ in all the cases at the same time it also offers robustness against the gradient inversion attacks. DOSFL+ cannot offer privacy as the condensed images are directly sent to server, In the case of MNIST we can see that the condensed images carry the natural image prior. Both the DOSFL+ and PPFed under-performs compared to FedAvg for CIFAR-10 as the quality of the DCGM is not good enough to attain the desired accuracy Zhao et al. (2021). For the case of LeNet+MNIST just by compromising the accuracy of 4.5 % we can attain a better privacy.



(a) CIFAR10 (b) Lenet (c) FashionMNIST

Figure 6: Comparison of methods across different datasets with 5 clients

5.2 Accuracy vs Time Trade off

In Figure 7 we analyze the Accuracy vs Time trade-off for PPFed. Hollow markers indicate the accuracy of a freshly initialised model trained for 300 epochs, on CD generated using the value of K indicated by the shape of the marker; and the time taken for generation of CD, at the end of Stage 1 of PPFed. Furthemore, each solid marker represents the accuracy of an model trained for 20 local epochs on CD generated using K value indicated by the marker during an FL Round, and the training + aggregation time taken for that FL Round in Stage 2 of PPFed. The black line

Table 2: Top 1% Accuracy comparison for 5 clients across datasets and models

Model+Dataset	FedAvg (Accuracy %)	DOSFL+ (Accuracy %)	PPFed (Accuracy %)
ConvNet+CIFAR10	70.2	50.1	50.3
LeNet+MNIST	96.5	93.0	92.0
MLP+FMNIST	85.2	80.6	81.0

joining solid markers in these plots indicates the Pareto frontier which is the set of Pareto-efficient configurations with respect to accuracy and training time.

In order to find the right configuration (K is the number of DCGM iterations) for generating CD, such that 80% accuracy is reached in the least time, one need not explore all the possible configurations to get to that value. Instead, one could follow the configurations on the Pareto frontier (black line) and move along it until 80% accuracy is reached - the marker near that accuracy gives us the needed configuration, which in this case is K = 30, and which results in the least end-to-end time of 720s for reaching that accuracy. In a similar manner, if we were given a time budget of say 1200s, then a maximum accuracy of about 90% can be attained by using the $K = 50$ configurations.

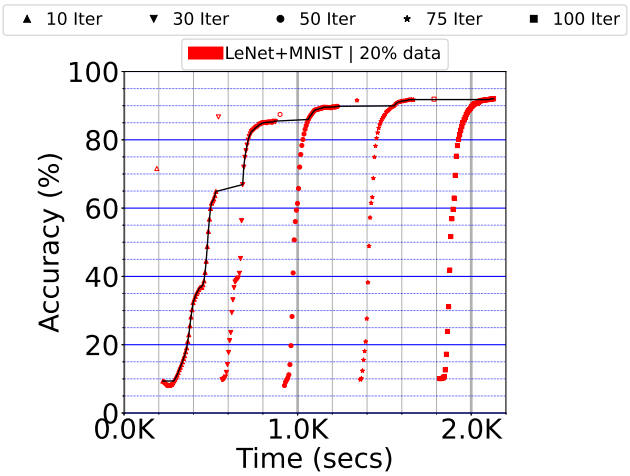


Figure 7: LeNet+MNIST | i.i.d. | 5 clients | Accuracy vs Time

6 Conclusion and Future Work

In this work we propose a novel FL scheme PPFed were the models are trained on the condensed data. We empirically show that PPFed is robust to gradient based attacks than FedAvg for the CNN-based model architectures, and achieves better privacy. We consider only iid data here and in future, plan to evaluate the robustness of gradient based attacks on non-iid data. Our proposed approach is generic by design. One can include any dataset condensation method, and any state-of-the-art aggregation methods such as FedDyn Acar et al. (2021) and Scaffold Karimireddy et al. (2020) to further enhance the accuracy of the model. We leave these aspects for the future investigation. We also observe that MLP architectures are vulnerable to gradient based attacks, preserving the privacy in such cases will be an interesting future direction.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

Aritra Dutta, El Houcine Bergou, Ahmed M Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3817–3824, 2020.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241, 2021.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020b.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. 2017.

Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. End-to-end federated learning for autonomous driving vehicles. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10174–10183, 2022.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *ICLR*, 1(2):3, 2021.

Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.