Every Vote Counts: Ranking-Based Training of Federated Learning to Resist Poisoning Attacks

Hamid Mozaffari, Virat Shejwalkar, and Amir Houmansadr,
University of Massachusetts Amherst

https://www.usenix.org/conference/usenixsecurity23/presentation/mozaffari

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

USENIX'23 Artifact Appendix: "Every Vote Counts: Ranking-Based Training of Federated Learning to Resist Poisoning Attacks"

Hamid Mozaffari, Virat Shejwalkar & Amir Houmansadr
University of Massachusetts Amherst
{hamid, vshejwalkar, amir}@cs.umass.edu

A Artifact Appendix

A.1 Abstract

This report provides details of our implementation of a federated learning algorithm called Federated Rank Learning (FRL) which is designed to achieve high test accuracy and mitigate the risk of model poisoning attacks in a non-iid client distribution. The report includes the implementation of FRL on CIFAR10 and MNIST datasets with different percentages of malicious clients. The evaluation workflow of the code includes running experiments to validate the major claims of the paper and measuring the test accuracy of the models. The report also includes the installation instructions and requirements for running the code.

A.2 Description & Requirements

A.2.1 How to access

Our artifact, the implementation of our work on Federated Rank Learning (FRL), can be accessed at https://github.com/SPIN-UMass/FRL. The code is written in PyTorch and is publicly available for anyone to use. The repository includes a comprehensive readme file that provides instructions on how to run different experiments, making it easy for others to replicate our results and build upon our work. To ensure that our artifact is easily accessible and can be referenced by others in the future, we have chosen to host it on the popular repository hosting platform GitHub.

A.2.2 Hardware dependencies

To evaluate our artifact, a system with a GPU is required for faster learning. Our experiments were conducted on an NVIDIA GeForce GTX 1080 Ti with 11GB RAM.

A.2.3 Software dependencies

The experiments in this study were conducted using the PyTorch 1.13.1 and Numpy 1.23.5 libraries. PyTorch is a widely-used deep learning framework that provides a seamless integration of computation graphs and tensors, making it an ideal choice for implementing and training neural networks.

Table 1: In our experiments, we use the following, state-of-the-art model architectures.

Architecture	Layer Name	Number of parameters
LeNet (MNIST)	Conv(32)	288
	Conv(64)	18432
	FC(128)	1605632
	FC(10) or FC(62)	1280
Conv8 (CIFAR10)	Conv(64), Conv(64)	38592
	Conv(128), Conv(128)	221184
	Conv(256), Conv(256)	884736
	Conv(512), Conv(512)	3538944
	FC(256), FC(256), FC(10)	592384

Numpy, on the other hand, is a library for the Python programming language that provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays. By using these two libraries in our experiments, we were able to efficiently implement and evaluate the performance of our models.

A.2.4 Benchmarks

We provide two benchmark datasets widely used in prior works on federated learning robustness:

MNIST is a 10-class class-balanced classification task with 70,000 gray-scale images, each of size 28×28. We experiment with LeNet architecture given in Table 1. For local training in each FRL/FL round, each client uses 2 epochs. For training ranks (experiments with FRL), we use SGD with learning rate of 0.4, momentum 0.9, weight decay 1e-4, and batch size 8.

CIFAR10 is a 10-class classification task with 60,000 RGB images (50,000 for training and 10,000 for testing), each of size 32×32. We experiment with a VGG-like architecture given in Table 1. For local training in each FRL/FL round, each client uses 5 epochs. For training ranks (experiments with FRL), we optimize SGD with learning rate of 0.4, momentum of 0.9, weight decay of 1e-4, and batch size of 8.

Table 1 shows the state-of-the-art model architectures and corresponding datasets that we use in our experiments. We also show the number of parameters in each of their layers.

A.3 Set-up

A.3.1 Installation

To install the code for this study, please follow these steps:

1. To download the repository, use the following command: **git clone https://github.com/SPIN-UMass/FRL.git**. The final stable URL is: https://github.com/SPIN-UMass/FRL/tree/4cf2550972e0e6299f61f682579f10b8e32c39d7.
2. Create a new conda environment. you can do so using the following command: **conda create --name FRL_test python=3.10.9**
3. Activate the environment: **conda activate FRL_test**
4. Then, to install the dependencies, run: **pip install -r requirements.txt**

This will download the repository and install all of the necessary dependencies, including PyTorch and Numpy, as specified in the software appendices section. Once the installation is complete, you should be able to run the code and reproduce the results from the study.

A.3.2 Basic Test

To run a simple experiment on the CIFAR10 dataset using the Federated Rank Learning (FRL) algorithm, run the following command: **python main.py --data_loc /CIFAR10/data/ --config experiments/001_config_CIFAR10_Conv8_FRL_1000users_noniid 1.0_nomalicious.txt**. This will initiate a federated learning experiment on the CIFAR10 dataset using 1000 clients in a non-iid fashion with a Dirichlet distribution parameter $\beta = 1.0$. The experiment will run for 2000 global FL rounds, with 25 clients selected for local updates in each round. Upon completion, the results of the experiment will be recorded and can be analyzed to evaluate the performance of the FRL algorithm on the CIFAR10 dataset.

A.4 Evaluation workflow

A.4.1 Major Claims

These are the major claims made in our paper:

(C1): FRL can achieve similar performance as FedAvg, Trimmed-Mean and Multi-Krum on CIFAR10 and MNIST distributed over a large number of clients in a non-iid fashion when there is no malicious client, as illustrated in Table 1 in the paper.

(C2): FRL can achieve high test accuracy when 10% of the clients are malicious on CIFAR10 and MNIST distributed over a large number of clients in a non-iid fashion, as illustrated in Table 1 in the paper. FedAvg, Trimmed-Mean and Multi-Krum results in lower test accuracy.

(C3): FRL can achieve high test accuracy when 20% of the clients are malicious on CIFAR10 and MNIST distributed over a large number of clients in a non-iid fashion, as illustrated in Table 1 in the paper. FedAvg, Trimmed-Mean and Multi-Krum results in lower test accuracy.

A.4.2 Experiments

The experiments directory includes the experiments performed in the paper. This section explains the purpose of each experiment and the expected outcome.

(E1): *[FL with 0% malicious client] [30 human-minutes + 16 compute-hour]: For claim C1.*

Execution:

For CIFAR10:

- run FRL: **python main.py --data_loc /CIFAR10/data/ --config experiments/001_config_CIFAR10_Conv8_FRL_1000users_noniid1.0_nomalicious.txt**.
- We also provided more experiments for FedAVG, Trimmed-Mean and Multi-Krum in the experiments directory.

For MNIST:

- run FRL: **python main.py --data_loc /MNIST/data/ --config experiments/004_config_MNIST_LeNet_FRL_1000users_noniid1.0_nomalicious.txt**
- We also provided more experiments for FedAVG, Trimmed-Mean and Multi-Krum in the experiments directory.

Results: The results of the experiment will be stored in the Logs directory. For MNIST, the expected test accuracy should be around 98%, and for CIFAR10, the expected test accuracy should be around 85%.

(E2): *[FRL with 10% malicious client] [30 human-minutes + 16 compute-hour]: For claim C2.*

Execution: For CIFAR10, run python **main.py --data_loc /CIFAR10/data/ --config experiments/002_config_CIFAR10_Conv8_FRL_1000users_noniid1.0_10pmal.txt**. For MNIST, run **python main.py --data_loc /MNIST/data/ --config experiments/005_config_MNIST_LeNet_FRL_1000users_noniid1.0_10pmal.txt**.

Results: The results of the experiment will be stored in the Logs directory. For MNIST, the test accuracy should be around 98%, and for CIFAR10, the test accuracy should be around 79%.

(E1): *[FRL with 20% malicious client] [30 human-minutes + 16 compute-hour]: For claim C3.*

Execution: For CIFAR10, run **python main.py --data_loc /CIFAR10/data/ --config experiments/003_config_CIFAR10_Conv8_FRL_1000users_noniid1.0_20pmal.txt**. For MNIST, run **python main.py --data_loc /MNIST/data/ --config experiments/006_config_MNIST_LeNet_FRL_1000users_noniid1.0_20pmal.txt**.

Results: The results of the experiment will be stored in the Logs directory. For MNIST, the test accuracy should be around 98%, and for CIFAR10, the test accuracy should be around 69%.

It is important to note that the results may vary slightly due to the random initialization of the models and the random sampling of clients in each round of federated learning. Therefore, it is recommended to run the experiments multiple times and report the average of the results to obtain a more robust evaluation of the performance of the FRL algorithm.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.