UniFed: A Benchmark for Federated Learning Frameworks

Xiaoyuan Liu
UC Berkeley
xiaoyuanliu@berkeley.edu

Tianneng Shi
Shanghai Jiao Tong University
stneng@sjtu.edu.cn

Chulin Xie
UIUC
chulinx2@illinois.edu

Qinbin Li
National University of Singapore
qinbin@comp.nus.edu.sg

Kangping Hu
Zhejiang University
hukp@zju.edu.cn

Haoyu Kim
Peking University
jhy2001@pku.edu.cn

Xiaojun Xu
UIUC
xiaojun3@illinois.edu

Bo Li
UIUC
lbo@illinois.edu

Dawn Song
UC Berkeley
dawnsong@cs.berkeley.edu

Abstract

Federated Learning (FL) has become a practical and popular paradigm in machine learning. However, currently, there is no systematic solution that covers diverse use cases. Practitioners often face the challenge of how to select a matching FL framework for their use case. In this work, we present UniFed, the first unified benchmark for standardized evaluation of the existing open-source FL frameworks. With 15 evaluation scenarios, we present both qualitative and quantitative evaluation results of nine existing popular open-sourced FL frameworks, from the perspectives of functionality, usability, and system performance. We also provide suggestions on framework selection based on the benchmark conclusions and point out future improvement directions.

1 Introduction

Federated Learning (FL) [42, 26] has become a practical and popular paradigm for training machine learning (ML) models. There are many existing open-source FL frameworks. However, unlike Pytorch [43] and TensorFlow [8] for ML, currently, there is not a dominant systematic solution that is maturely developed for most use cases.

To compare existing open-source solutions, we created UniFed, an FL benchmark for standardized evaluations of FL frameworks. Specifically, UniFed helps answer two questions:

- How to qualitatively and quantitatively characterize an FL framework?
- How to choose the best FL framework for a specific real-world application?

We find that existing FL frameworks have significant qualitative differences which we present in Table 2 and Table 3. In addition, our training experiments on nine existing FL frameworks with different FL algorithm implementations suggest that the selection of model type is the main factor that affects model performance compared with the selection of algorithm and framework. Our measurement of system performance shows that, interestingly, when considering training efficiency, communication efficiency, and memory usage, there is no framework that consistently outperforms others.

Submitted to the 36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks. Do not distribute.

The contribution of this paper is summarized below.

1. We define the criteria to characterize an FL framework, including functionality support, system performance, and usability. We also develop a toolkit where users can easily deploy and test FL frameworks in various settings in one command, which facilitates a workflow of a standardized quantitative evaluation.
2. We collect and categorize a list of nine representative FL frameworks. With the criteria and the toolkit, we benchmark and compare them both qualitatively and quantitatively.
3. Summarizing the result in our evaluation, we present a complete guideline that helps FL practitioners choose the FL framework for a specific real-world application.

2 Related work

2.1 Existing datasets and frameworks

We give background on existing FL datasets in Appendix A. In our benchmark, we cover both the simulated datasets and real federated datasets from diverse application domains for evaluation. Specifically, considering the real-world usage of FL frameworks, we adopt the datasets from the LEAF [10] for experiments on cross-device horizontal FL, which are more practical than the simulated datasets. For cross-silo horizontal FL and vertical FL, we adopt the generated datasets from FATE [38] for evaluation, which cover representative FL applications among institutions from finance to healthcare.

There are many system construction efforts on building frameworks to support various FL scenarios. In this work, we focus on open-source FL frameworks that are available for evaluation. Here we identify three general categories along with representative examples.

All-in-one frameworks. Great efforts have been made to construct a single framework that covers most FL-related techniques in both horizontal and vertical FL settings. Such FL frameworks (FATE [38], FedML [22], PaddleFL [6], Fedleaner [3]) focus on the coverage of the functionalities and are often constructed with great engineering efforts. For example, FATE from WeBank is an industrial-grade FL framework that aims to provide FL services for enterprises and institutions.

Horizontal-only frameworks. Instead of aiming to support diverse applications with both horizontal and vertical FL, some frameworks (TFF [7], Flower [9], FLUTE [16]) aim to provide easy-to-use APIs for users to adopt and develop horizontal FL algorithms. For example, based on TensorFlow [8], TFF [7] from Google provides federated learning API and federated core API for users to apply and design FL algorithms, respectively.

Specialized frameworks. While the above frameworks support the general development of FL, some frameworks (CrypTen [27], FedTree [34]) are specially designed for specific purposes. CrypTen [27] focuses on providing secure multi-party computation [46] primitives, while FedTree is designed for the federated training of decision trees.

2.2 Existing benchmarks

We give background on existing FL benchmarks in Appendix B. In summary, they mainly focus on creating federated datasets in different tasks, either from natural client data or from artificially partitioned centralized datasets, to evaluate FL *algorithms*. However, they do not provide the systematic evaluation of FL *frameworks* that are built with industry efforts and used as real FL systems in practice. To fill in this gap, we benchmark nine open-source FL frameworks with 15 common FL datasets to cover different FL settings, data modalities, task, as well as workload sizes.

3 Benchmark design

To get first-hand experience and quantitatively evaluate target FL frameworks, we design UniFed benchmark toolkit and integrate all target frameworks with minimal intrusions in coding. Figure 1 shows an example evaluation workflow using our toolkit. UniFed toolkit contains four components:

(1) An environment launcher that provides a Command Line Interface (CLI) to read the experiment specification from a configuration file and launch a distributed testing environment. (2) A scenario loader python package that facilitates easy access to evaluation scenarios with automatic caching. (3) A set of framework-dependent code patches that inject extra code to the target FL frameworks for evaluation-related data loading and performance logging. (4) A global log analyzer that collects log files from distributed evaluation nodes and reports the benchmarking result. With our benchmark toolkit, one can start an FL training experiment using any FL framework with a one-line command.

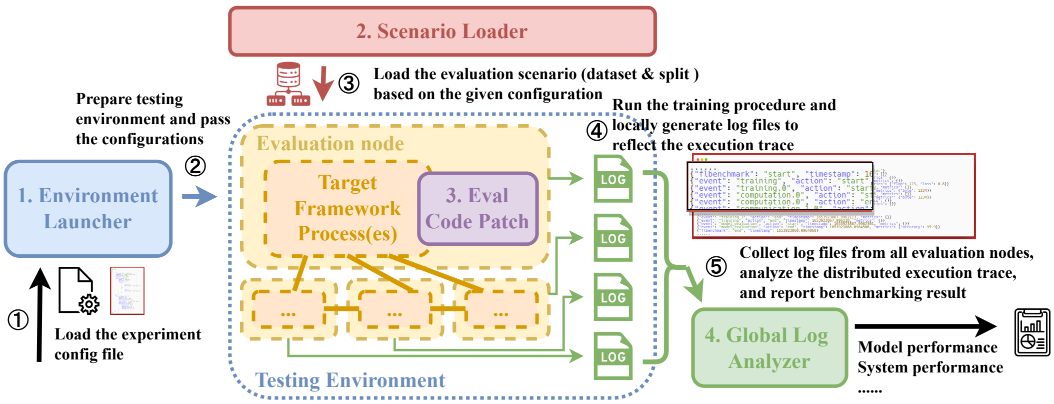

Figure 1: Design of the benchmark workflow.

To incorporate a new FL framework into our workflow, we first update our environment launcher to support the deployment of the target framework based on the instructions from its documentation. We then write code patches to the target framework for integration: (a) Support the data loading of our evaluation scenarios. (b) Configure the framework behavior based on the configuration file received from the launcher. (c) Generate timestamped event logs to record the training procedure so that different frameworks generate log files with a unified format for fair comparisons. More details are given in Section 5.1. To evaluate the target framework, following the steps in Figure 1, one needs to write a configuration file specifying the experiment details (e.g. which framework to use, what model type to train, what learning rate to apply) and start the training with the CLI from the environment launcher. One can also run the CLI provided by the global log analyzer anytime to see if the training has finished and, after the training finishes, get a comprehensive report of both model and system performance.

Overall, users can easily adopt our UniFed toolkit to deploy and test different FL frameworks in comprehensive scenarios with a one-line command. Moreover, developers can integrate their frameworks into our toolkit for comparison following the same workflow.

4 Evaluation principles of UniFed

4.1 Evaluation scenarios and evaluation targets

For clarity, we use the term model performance to refer to the model's ability to perform a task and the term system performance to refer to its training efficiency (mainly discussed in Section 4.3). Theoretically, despite the difference in implementation (e.g. different ML backends and communication orchestration), as long as the frameworks achieve the same mathematical procedure with the same FL training algorithm, the resulting model should have similar model performance. In this work, we measure the model performance across different frameworks to verify this statement. We also explore the algorithm efficiency difference across different training algorithms and models.

We first define the scope of an evaluation scenario in our benchmark. Inspired by experiments in existing FL studies, in this work, we define *one evaluation scenario* to be *a set of clients who hold fixed dataset splits (training/validation/test) with a fixed partition across different clients and an optional aggregator/arbiter who potentially also holds a dataset with only the test set*. In vertical FL, each data instance has a unique identifier (id) and vertical split data instances have aligned identifiers

Setting	Scenario name	Modality	Task type	Performance metrics	Client number	Sample number
cross-device horizontal	celeba [39]	Image	Binary Classification (Smiling vs. Not smiling)	Accuracy	894	20,028
	femnist [30, 14, 10]	Image	Multiclass Classification (62 classes)	Accuracy	178	40,203
	reddit [10]	Text	Next-word Prediction	Accuracy	813	27,738
cross-silo horizontal	breast_horizontal [1]	Medical	Binary Classification	AUC	2	569
	default_credit_horizontal [47, 17]	Tabular	Binary Classification	AUC	2	22,000
	give_credit_horizontal [4]	Tabular	Binary Classification	AUC	2	150,000
	student_horizontal [15, 17]	Tabular	Regression (Grade Estimation)	MSE	2	395
	vehicle_scale_horizontal [44, 17]	Image	Multiclass Classification (4 classes)	Accuracy	2	846
					Vertical split details	
vertical	breast_vertical [1]	Medical	Binary Classification	AUC	A: 10 features 1 label B: 20 features	
	default_credit_vertical [47, 17]	Tabular	Binary Classification	AUC	A: 13 features 1 label B: 10 features	
	dvisits_vertical [11]	Tabular	Regression (Number of consultations Estimation)	MSE	A: 3 features 1 label B: 9 features	
	give_credit_vertical [4]	Tabular	Binary Classification	AUC	A: 5 features 1 label B: 5 features	
	motor_vertical [2]	Sensor data	Regression (Temperature Estimation)	MSE	A: 4 features 1 label B: 7 features	
	student_vertical [15, 17]	Tabular	Regression (Grade Estimation)	MSE	A: 6 features 1 label B: 7 features	
	vehicle_scale_vertical [44, 17]	Image	Multiclass Classification (4 classes)	Accuracy	A: 9 features 1 label B: 9 features	

Table 1: Evaluation scenarios in UniFed. UniFed borrow 15 datasets from existing works to cover different FL settings, modalities, task types, and workload sizes.

in different participants. The model performance is measured with all available test instances among participants in an aligned, unweighted, but non-deduplicated way. We list all evaluation scenarios considered in this paper in Table 1.

We then define the scope of an evaluation target that determines the granularity of our evaluation. FL frameworks often support multiple FL algorithms (e.g. FedAvg [42], SecureBoost [13]) that uses different local training methods and different mathematical procedures for aggregation. Using a specific FL algorithm, one can train different ML models (e.g. linear regression, logistic regression, MLP, LeNet [31]) for a given task. Moreover, with different training parameters (e.g. epoch number, batch size, learning rate, choice optimizer), the same ML model can achieve different final model performances for the same task. Considering all differences, in this work, we define the basic unit for evaluation in the benchmark to be a combination of *(FL framework, FL algorithm, ML model)* and always measure the performance with a proper set of hyperparameters, which are chosen separately for different evaluation scenarios in advance with grid searches for the best model performance.

4.2 Functionality support

Different FL frameworks support different sets of functionalities. Specifically, we consider the model support in both horizontal and vertical settings, the deployment support, and privacy-protection features. The model support reflects whether the evaluation target is able to train a specific type of model in a specific setting. The deployment support measures the scalability of the evaluation target and challenges its communication infrastructure. And for privacy-protection features, we examine whether the evaluation target has proper mechanisms to resist certain types of privacy threats. We give a functionality comparison for all nine frameworks in Table 2. Note that, UniFed focus on features that are commonly supported by the frameworks off the shelf. There are also latest research projects developing more functionalities for better training optimization, better robustness [50], more comprehensive differential privacy, and improved fairness. Most frameworks in our evaluation can potentially be extended for those additional functionalities, which is an interesting future direction.

From the table, we make the following observations.

Framework	All-in-one frameworks				Horizontal-only frameworks			Specialized frameworks	
	FATE	FedML	PaddleFL	Fedlearner	TFF	Flower	FLUTE	CrypTen	FedTree
Model support - Horizontal									
Regression	Y	Y	Y	Y	Y	Y	Y	N/A	N
Neural network	Y	Y	Y	Y	Y	Y	Y	N/A	N
Tree-based model	Y	N	N	N	N	N	N	N/A	Y
Model support - Vertical									
Regression	Y	Y	Y*	N	N	N	N	Y	N
Neural network	Y	N	Y*	Y	N	N	N	Y	N
Tree-based model	Y	N	N	Y	N	N	N	N	Y
Deployment support									
Single-host simulation	Y	Y	Y	Y	Y	Y	Y	Y	Y
Multi-host deployment (<16 hosts)	Y	Y*	Y	Y	N	Y	Y	Y	Y
Cross-device deployment (>100 host)	N	Y*	Y	Y	N	Y	Y	N/A	Y
Privacy protection against the semi-honest server									
Does not require a 3rd party aggregator (vertical)	Y	N	Y	Y	N	N	N	Y	Y
Aggregator does not learn model param (arbitar scenario)	Y	N	Y	N	N	N	N	N/A	Y
Aggregator does not learn individual model gradient (secagg)	Y	Y	Y	N	Y	N	N	N/A	Y
Privacy protection against semi-honest peer clients									
Clients does not learn anything about the model param (vertical)	Y	N	Y	N	N/A	N/A	N/A	Y	Y
Clients does not learn gradients from other clients (vertical)	Y	Y	Y	N	N/A	N/A	N/A	Y	Y
Privacy protection in the final model									
Support training with central DP (dpsgd/gradient edits)	N	N	Y	N	Y	Y	Y	N	Y

Table 2: Functionality support in different FL frameworks. Asterisks indicate a claimed support for certain functionalities that are missing or cannot run in the open-source implementation.

1. **Model support.** For horizontal settings, most frameworks support both regression and neural networks, while only a few (FATE, FedTree) support tree-based models. For vertical settings, only all-in-one frameworks support the corresponding algorithms and the coverage is incomplete. Tree-based vertical training is only supported by three frameworks (FATE, Fedlearner, and FedTree).

2. **Deployment support.** While all frameworks support the single-host deployment as a basic functionality, surprisingly most frameworks provide the multi-host deployment option for realistic FL simulation. The only exception is TensorFlow Federated which has multi-host deployment as its incoming feature in development. However, for cross-device support where we challenge the scalability of the evaluation target, although most frameworks claim they support the cross-device training, we experience various glitches in practice that prevent a successful deployment. More details are discussed in Section 5.

3. **Privacy enhancement.** We investigate the privacy features that are actually implemented in our evaluation targets and categorize them based on their different threat models. Aligned with [26], we identify three types of protections against attackers with different access. (1) Specifically, to keep private information from an honest-but-curious central server in a vertical setting, some frameworks (FATE, PaddleFL, Fedlearner, CrypTen, FedTree) support different protocols without arbiters which provide the ultimate protection. For example, FATE uses HE-based solutions [20, 48, 49] for regression and neural network while CrypTen uses sMPC-based solutions [28]. For tree-based models, most frameworks use SecureBoost [13] in the vertical setting and, in the horizontal setting, a histogram secure aggregation (HistSecAgg) mentioned in [5]. In addition, some frameworks (FATE, PaddleFL, FedTree) take advantage of arbiters for better computation efficiency but do not reveal any model parameter. In settings where the aggregator needs the final model as the output, there is also the option of secure aggregation that prevents the aggregator from learning individual model gradients. (2) On the other side, to prevent clients from getting extra information in vertical settings, most frameworks that support vertical settings implement corresponding protection. The only exception is Fedlearner which only implements split learning and introduces certain amounts of gradient leakage [33]. We also notice that most implemented protection mechanisms are assuming a semi-honest model. (3) Finally, to protect user privacy and defend potential privacy attacks (e.g., membership inference, model-inversion) on the final production model, some frameworks (PaddleFL, TFF, Flower, FLUTE, FedTree) support applying differential privacy [18] in training.

4.3 System performance

Although there is a huge overlap in the functionality support for different FL frameworks, the implementations are often quite different, leading to different performance characteristics. To finish the complete FL training task, the frameworks often need to preprocess the data, locally computes certain functions, and potentially communicate with an aggregator to collaboratively learn the model. In most cases, the frameworks improve the model iteratively and repeat the above steps after a configured number of epochs or until certain criteria are matched.

Framework	All-in-one frameworks				Horizontal-only frameworks			Specialized frameworks	
	FATE	FedML	PaddleFL	Fedlearner	TFF	Flower	FLUTE	CrypTen	FedTree
Documentation									
Detailed tutorial	Y	Y	Y	N	Y	Y	Y	Y	Y
Code example	Y	Y	Y	Y	Y	Y	Y	Y	Y
API documentation	Y	N	N	N	Y	Y	N	Y	Y
Engineering									
Native test & benchmark	Y	Y	N	N	Y	Y	Y	Y	Y
GPU support	Y	Y	Y	Y	Y	Y	Y	Y	Y
Built-in ML building block									
CNN	Y	Y	Y	Y	Y	Y	Y	Y	N
RNN	Y	Y	Y	Y	Y	Y	Y	N	N
Rich Optimizers	Customized	Torch	PaddlePaddle	TensorFlow	TensorFlow	Y	Torch	Only SGD	N/A

Table 3: Usability feature comparison in different FL frameworks.

In this work, we target to measure the system performance in three aspects: training efficiency, communication cost, and resource consumption. Specifically, we are interested in a direct comparison between different frameworks training the same ML model. Because of the difference in their ML backend, communication orchestration, and implementation quality for model aggregation and synchronization, we target to find the best FL frameworks for each of our evaluation scenarios . We use logging in UniFed toolkit for performance tracking which we discuss in Section 3 to record and analyze the complete training procedure of a potentially distributed evaluation target. We discuss the evaluation result in Section 5 and a direct system performance comparison is given in Table 7.

4.4 Usability

In addition to functionality and efficiency, whether the framework is easy to learn and convenient to use also affects its popularity. In this work, we first define a set of qualitative attributes and apply them to different FL frameworks to measure their usability. Specifically, we focus on three aspects: documentation, engineering, and built-in ML components. As frameworks often use their own term to refer to different pieces in their documentation, we standardize our requirement of tutorial, code example, and API documentation in Appendix C. In terms of engineering efforts, we mainly check if the target framework has its own tests and performance benchmark, and also check whether the GPU support can be explicitly configured. Last, we examine whether the framework has integrated basic ML building blocks like specific network structures and optimizers for convenient usage.

Based on the criteria listed above, we show our evaluation result in Table 3. Most frameworks provide details on their installation and usage. FedML, Fedlearner, and FLUTE do not provide API documentation for users to set up different FL scenarios easily. In terms of engineering efforts, all frameworks provide internal testing and benchmarking code except PaddleFL and Fedlearner. Moreover, all frameworks support the usage of GPU to accelerate training. In terms of ML building blocks, most frameworks have integrated CNN and RNN except CrypTen and FedTree, which are designed for specific purposes. FATE and Flower are compatible with different backend libraries such as TensorFlow and PyTorch, while the other frameworks support optimizers in its own backend.

5 Benchmark evaluation

5.1 Implementation

We implement and open source [1] UniFed toolkit discussed in Section 3. Specifically, our environment launcher uses SSH to connect to the evaluation node and prepares the testing environment. We wrap the data loading for datasets from [38], further automate the file caching from [10], and fix the dataset splits for the evaluation scenarios as discussed in Section 4.1. Our own logging format based on the JSON file structure records the timestamp for critical events. We explain the details of the logging format in Appendix D. For each framework, we create a separate code patch following the principle of minimal intrusion and resource consumption. We explain the details about separate patches to each individual framework in Appendix E. All experiments use evaluation nodes with 20 vCPU in Intel Xeon Gold 6230.

[1] https://github.com/AI-secure/FLBenchmark-toolkit

Setting	Model	FATE	FedML	PaddleFL	Fedlearner	TFF	Flower	FLUTE
femnist cross-device (Accuracy)	logistic_regression	/	0.083	0.053	/	0.058	0.036	0.072
	mlp_128	/	0.652	0.591	/	0.644	0.663	0.641
	mlp_128_128_128	/	0.701	0.671	/	0.722	0.707	0.697
	lenet	/	0.822	0.792	/	0.822	0.819	0.820
give credit cross-silo (AUC)	logistic_regression	0.693	0.788	0.788	0.790	0.790	0.795	0.790
	mlp_128	0.830	0.832	0.828	0.834	0.832	0.831	0.833
	mlp_128_128_128	0.831	0.834	0.827	0.835	0.832	0.834	0.834

Table 4: FedAvg with different models on different tasks in the horizontal FL setting. FATE and Fedlearner do not support cross-device setting and are excluded from the comparison. We can observe that different FL frameworks show similar performance in general when using the same model.

5.2 Benchmark results

With UniFed toolkit, we run experiments and present representative benchmark quantitative results related to the research questions in Section 1 and Section 4. We also analyze possible reasons for the experiment outcomes by comparing implementations and designs in the frameworks.

RQ1: Does the choice of FL framework affects the model performance trained using the same FL algorithm? As mentioned in Section 4.1, we expect a unified model performance across different FL frameworks because of the same mathematical procedure of training. With our benchmark result, we verify this finding with the most commonly supported FedAvg and the result is shown in Table 4. Results for FATE and Fedlearner are partially missing due to their limited support in the cross-device setting. Specifically, a file naming issue in FATE prevents it from scaling hundreds of clients, and although Fedlearner supports cross-device training, it does not support sampling a subset of clients and has different synchronizing mechanisms which prevent a fair comparison.

In Table 4, the model performance of the same model is generally consistent across different FL framework implementations (performance difference within 1.1% in most cases) and the trend that deeper models perform better can be observed in all frameworks for the selected scenarios. In addition, we observe that (1) The logistic regression model does not work well in femnist scenario, which leads to consistently poor performance in all frameworks. (2) In the cross-silo setting, the logistic regression model in FATE has a relatively low performance which might be relevant to its default early-stop behavior triggered by convergence. (3) The PaddleFL model performance is unstable and consistently lower, which is potentially caused by its different ML backend PaddlePaddle [41].

RQ2: Are different FL algorithm implementations comparable when training the same type of model? In addition to FedAvg, FL frameworks also implemented other FL algorithms to cover specific use cases. As the selection of algorithms is less consistent for tree-based and vertical cases, in this research question, we focus on a comparison between different frameworks with different FL algorithm implementations training the same model. The result is presented in Table 5. We note that FedML only supports regression in the vertical setting. For PaddleFL, we failed to run the sMPC example following the official instructions in its latest version 1.2.0 and its split learning support is also removed. Fedlearner only provides one-layer networks for split learning off-the-shelf. CrypTen does not support tree-based models and FedTree does not support non-tree-based models.

In Table 5, again we observe relatively consistent model performance in each row, which suggests the model selection is still the main factor that influences the model performance even with different FL algorithm implementations. In addition, to explain the larger diversity compared with Table 4, we note that (1) For the logistic regression, FATE achieves slightly better model performance probably due to its default regularization option, while FedML suffers a performance loss that might be caused by its default LeakyReLU activation. (2) FL algorithm in FATE failed to efficiently support the training of a 3-layer multi-level perceptron (MLP). We report the result after an insufficient training of one epoch which takes more than 8 hours. CrypTen achieves better performance with sufficient training using an efficient sMPC-based approach. (3) Tree-based models have more consistent model performance despite their different FL algorithm implementations in different programming languages.

Setting	Model	FATE	FedML	PaddleFL	Fedlearner	CrypTen	FedTree
default credit vertical (AUC)	Regression (logistic_regression)	0.717 HE-based	0.650 HE-based	/	/	0.708 sMPC-based	/
	Neural network (mlp_128_128_128)	0.737 (1 epoch) HE-based	/	/	/	0.789 sMPC-based	/
	Tree-based model (gbdt_64_64_6)	0.820 SecureBoost	/	/	0.819 SecureBoost	/	0.817 SecureBoost
give credit horizontal (AUC)	Tree-based model (gbdt_64_64_6)	0.861 HistSecAgg	/	/	/	/	0.861 HistSecAgg

Table 5: Comparison among different FL algorithm implementations that train the same model. We observe that the model performance is still mainly determined by the model selection.

Setting	Name	1st		2nd		3rd	
		alg&model	perf	alg&model	perf	alg&model	perf
cross-device horizontal	celeba (Accuracy)	FedAvg leaf_cnn	90.19%	FedAvg resnet_18	88.99%		
	femnist (Accuracy)	FedAvg lenet	82.23%	FedAvg mlp_128_128_128	72.24%	FedAvg mlp_128	66.33%
	reddit (Accuracy)	FedAvg lstm	13.36%				
cross-silo horizontal	breast_horizontal (AUC)	FedAvg mlp_128_128_128	98.86%	FedAvg logistic_regression	98.70%	FedAvg mlp_128	98.54%
	default_credit_horizontal (AUC)	HistSecAgg gbdt_64_64_6	78.46%	FedAvg mlp_128_128_128	77.70%	FedAvg mlp_128	77.21%
	give_credit_horizontal (AUC)	HistSecAgg gbdt_64_64_6	86.10%	FedAvg mlp_128_128_128	83.45%	FedAvg mlp_128	83.38%
	student_horizontal (MSE)	FedAvg mlp_128_128_128	21.04	FedAvg mlp_128	21.99	HistSecAgg gbdt_64_64_6	22.79
	vehicle_scale_horizontal (Accuracy)	FedAvg mlp_128_128_128	100.0%	FedAvg mlp_128	100.0%	HistSecAgg gbdt_64_64_6	99.64%
vertical	breast_vertical (AUC)	SecureBoost gbdt_64_64_6	100.0%	sMPC-based mlp_128_128_128	100.0%	sMPC-based mlp_128	99.97%
	default_credit_vertical (AUC)	SecureBoost gbdt_64_64_6	81.99%	sMPC-based mlp_128_128_128	78.89%	sMPC-based mlp_128	77.87%
	dvisits_vertical (MSE)	SecureBoost gbdt_64_64_6	0.32	sMPC-based mlp_128_128_128	0.57	sMPC-based mlp_128	0.60
	give_credit_vertical (AUC)	SecureBoost gbdt_64_64_6	86.79%	sMPC-based mlp_128_128_128	83.38%	sMPC-based mlp_128	82.79%
	motor_vertical (MSE)	sMPC-based mlp_128_128_128	3.66E-4	SecureBoost gbdt_64_64_6	3.64E-3	sMPC-based mlp_128	9.98E-03
	student_vertical (MSE)	SecureBoost gbdt_64_64_6	3.26	sMPC-based mlp_128_128_128	11.03	sMPC-based mlp_128	12.43
	vehicle_scale_vertical (Accuracy)	SecureBoost gbdt_64_64_6	99.17%	sMPC-based mlp_128_128_128	96.34%	sMPC-based mlp_128	94.21%

Table 6: Best algorithm and model combinations for each evaluation scenario. Tree-based models generally have advantages in vertical settings and deeper models are often preferred.

RQ3: How to select a model and FL algorithm combination to achieve a good model performance for the given application scenario? From RQ1 and RQ2, we verified that the model type is the major factor that influences the model performance as long as the implementation in the FL framework is correct. In RQ3, we want to find the best of such combination among all available FL algorithms and model combinations we tested in UniFed evaluation scenarios. Specifically, for cross-silo horizontal and vertical settings, we compare available models of regression, shallow neural network (1-layer MLP), deep neural network (3-layer MLP), and tree-based model (GBDT). For the cross-device settings, we find promising models that are available off-the-shelf (CNN model in LEAF [10], ResNet [23], LeNet [32], LSTM [24]) for a reference. We present a ranked comparison result in Table 6.

We notice that in some scenarios, the model performance is sensitive to the change in model type, while for other scenarios, the difference is less significant and sometimes multiple models get good performance. Specifically, (1) Tree-based models often perform better in vertical settings, in some cases even by a large margin. (2) Deeper neural networks often achieve better performance than shallow ones in most cases. We recommend the practitioners find scenarios in the benchmark that are similar to their use case for a reference for the model selection. For scenarios where the model performance is less sensitive to the model selection, the practitioners should consider comparing the system performance, which is discussed in the next RQ.

Setting	Model	FATE	FedML	PaddleFL	Fedlearner	TFF	Flower	FLUTE	CrypTen	FedTree
Training time in total										
femnist horizontal	Neural network (lenet)	/	2,000 epochs 2,373 s FedAvg	2,000 epochs 1,476 s FedAvg	/	2,000 epochs 7,268 s FedAvg	2,000 epochs 669 s FedAvg	2,000 epochs 614 s FedAvg	/	/
default credit vertical	Tree-based model (gbdt_64_64_6)	64 trees 2,800 s SecureBoost	/	/	64 trees 9,810 s SecureBoost	/	/	/	/	64 trees 489 s SecureBoost
	Neural network (mlp_128_128_128)	1 epoch 30,952 s HE-based	/	/	/	/	/	/	10 epochs 1,354 s sMPC-based	/
Communication cost										
femnist horizontal	Neural network (lenet)	/	80,000 Rounds 19.71 GiB	N/A N/A	/	80,000 Rounds 19.71 GiB	80,714 Rounds 20.41 GiB	80,060 Rounds 19.96 GiB	/	/
default credit vertical	Tree-based model (gbdt_64_64_6)	2,636 Rounds N/A	/	/	50,535 Rounds 1.36 GiB	/	/	/	/	11,969 Rounds 0.39 GiB
	Neural network (mlp_128_128_128)	1,886 Rounds N/A	/	/	/	/	/	/	350,289 Rounds 69.87 GiB	/
Peak memory usage in total										
femnist horizontal	Neural network (lenet)	/	0.55 GiB	9.21 GiB	/	1.95 GiB	52.12 GiB	4.91 GiB	/	/
default credit vertical	Tree-based model (gbdt_64_64_6)	9.93 GiB	/	/	0.71 GiB	/	/	/	/	5.46 GiB
	Neural network (mlp_128_128_128)	17.50 GiB	/	/	/	/	/	/	0.44 GiB	/

Table 7: System performance comparison in training time, communication cost, and peak memory usage. "/" suggests the lack of functionality and "N/A" suggests missing logging due to module separation (see Appendix E). No framework consistently outperforms others in all three factors.

RQ4: Which FL framework has the best system performance and what causes the differences? Here we consider the training time, the communication cost of participants, and the peak memory consumption as metrics to evaluate system performance. In this way, the benchmark provides reference on the FL framework selection for application scenarios with different hardware and resource constraints. Table 7 shows the evaluation results.

We have the following observations. (1) Regarding FedAvg on femnist, Flower and FLUTE have a much lower training time than the other frameworks. FedML and TFF are slow since they launch the clients with less or no parallelism among clients' training. All frameworks have the same or close communication cost following the FedAvg algorithm. For peak memory usage, Flower has a high memory requirement as it keeps the states of all clients at all times regardless of client sampling. (2) Regarding vertical FL with the tree-based model, FedTree is significantly faster than FATE and Fedlearner. Training with the same number of trees, FATE has the lowest communication frequency and Fedlearner has the lowest peak memory usage. (3) Regarding vertical FL with neural networks, while FATE and CrypTen adopt different privacy techniques to protect the transferred messages, CrypTen is much faster than FATE with lower memory usage. However, the communication frequency of CrypTen is high. Overall, there is no framework that consistently outperforms others in all three factors (i.e., training efficiency, communication efficiency, and memory usage).

6 Discussion and future work

Based on our benchmark results in Section 5, here we answer the question in the introduction by providing a complete FL framework selection guideline and also discuss future works.

For FL practitioners to select an FL framework for a specific use case, the first step is to analyze the qualitative requirement of the use case and narrow down the scope using Table 2 and Table 3. They should also find the benchmark scenario that is most similar to their use case and refer to Table 6 for a list of preferred model types. Considering the infrastructure hardware constraint for the use case, practitioners should cross-check Table 7 and Table 2 to find frameworks that best match their deployment environment. If no existing FL framework satisfies all constraints for the use case, practitioners should consider the option of customizing one of the frameworks and can refer to Table 3 to evaluate the feasibility and difficulty for further development.

There are the following future directions to further improve UniFed: (1) We expect more datasets can be incorporated into UniFed as the FL studies grow, especially for vertical FL. (2) We will periodically check the latest and representative FL frameworks (e.g., FedScale [29] which is one recent open-source framework that we do not consider due to time constraints) and include them into UniFed. (3) We may discuss and evaluate the fairness and incentives of FL frameworks when there are enough frameworks enabling these factors. (4) We plan to launch an open competition to call for efficient, effective, and secure solutions using the existing FL frameworks.

References

[1] Breast cancer wisconsin (diagnostic) data set. https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data. Accessed: 2022-06-08.

[2] Electric motor temperature. https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature. Accessed: 2022-06-08.

[3] Fedlearner. https://github.com/bytedance/fedlearner. Accessed: 2022-06-06.

[4] Give me some credit. https://www.kaggle.com/c/GiveMeSomeCredit/data. Accessed: 2022-06-08.

[5] Homo secureboost. https://github.com/FederatedAI/FATE/blob/master/doc/federatedml_component/ensemble.md#homo-secureboost. Accessed: 2022-06-09.

[6] Paddlefl. https://github.com/PaddlePaddle/PaddleFL. Accessed: 2022-06-06.

[7] Tensorflow federated. https://github.com/tensorflow/federated. Accessed: 2022-06-06.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[9] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. de Gusmão, and N. D. Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[10] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[11] A. C. Cameron, P. K. Trivedi, F. Milne, and J. Piggott. A microeconometric model of the demand for health care and health insurance in australia. *The Review of economic studies*, 55(1):85–106, 1988.

[12] D. Chai, L. Wang, K. Chen, and Q. Yang. Fedeval: A benchmark system with a comprehensive evaluation model for federated learning. *arXiv preprint arXiv:2011.09655*, 2020.

[13] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[14] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[15] P. Cortez and A. M. G. Silva. Using data mining to predict secondary school student performance. 2008.

[16] D. Dimitriadis, M. H. Garcia, D. M. Diaz, A. Manoel, and R. Sim. Flute: A scalable, extensible framework for high-performance federated learning simulations. *arXiv preprint arXiv:2203.13789*, 2022.

[17] D. Dua and C. Graff. UCI machine learning repository, 2017.

[18] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[19] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[20] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[21] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, P. S. Yu, Y. Rong, et al. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *Workshop on Distributed and Private Machine Learning: The International Conference on Learning Representations (DPML-ICLR*, 2021.

[22] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] S. Hu, Y. Li, X. Liu, Q. Li, Z. Wu, and B. He. The oarf benchmark suite: Characterization and implications for federated learning systems. *arXiv preprint arXiv:2006.07856*, 2020.

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

[27] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *arXiv 2109.00984*, 2021.

[28] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[29] F. Lai, Y. Dai, S. S. Singapuram, J. Liu, X. Zhu, H. V. Madhyastha, and M. Chowdhury. Fedscale: Benchmarking model and system performance of federated learning. In *International Conference on Machine Learning (ICML)*, 2022.

[30] Y. LeCun. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist`. Accessed: 2022-06-08.

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[33] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang. Label leakage and protection in two-party split learning. In *International Conference on Learning Representations*, 2022.

[34] Q. Li, Y. Cai, Y. Han, C. M. Yung, T. Fu, and B. He. Fedtree: A fast, effective, and secure tree-based federated learning system. `https://github.com/Xtra-Computing/FedTree/blob/main/FedTree_draft_paper.pdf`, 2022.

[35] Q. Li, Y. Diao, Q. Chen, and B. He. Federated learning on non-iid data silos: An experimental study. In *IEEE International Conference on Data Engineering*, 2022.

[36] Y. Liang, Y. Guo, Y. Gong, C. Luo, J. Zhan, and Y. Huang. Flbench: A benchmark suite for federated learning. In *BenchCouncil International Federated Intelligent Computing and Block Chain Conferences*, pages 166–176. Springer, 2020.

[37] B. Y. Lin, C. He, Z. Zeng, H. Wang, Y. Huang, M. Soltanolkotabi, X. Ren, and S. Avestimehr. Fednlp: A research platform for federated learning in natural language processing. *NAACL Findings*, 2022.

[38] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang. Fate: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, 22(226):1–6, 2021.

[39] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[40] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34, 2021.

[41] Y. Ma, D. Yu, T. Wu, and H. Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115, 2019.

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

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[44] J. Siebert. *Vehicle Recognition Using Rule Based Methods*. Turing Institute, 1987.

[45] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi. Privacy preserving vertical federated learning for tree-based models. *arXiv preprint arXiv:2008.06170*, 2020.

[46] A. C.-C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[47] I.-C. Yeh and C.-h. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2):2473–2480, 2009.

[48] Q. Zhang, C. Wang, H. Wu, C. Xin, and T. V. Phuong. Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. In *IJCAI*, pages 3933–3939, 2018.

[49] Y. Zhang and H. Zhu. Additively homomorphical encryption based deep neural network for asymmetrically collaborative machine learning. *arXiv preprint arXiv:2007.06849*, 2020.

[50] B. Zhu, L. Wang, Q. Pang, S. Wang, J. Jiao, D. Song, and M. I. Jordan. Byzantine-robust federated learning with optimal statistical rates and privacy guarantees. *arXiv preprint arXiv:2205.11765*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss our limitations and the future work in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We provide a benchmark for existing FL frameworks so it is not closely relevant.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include those in the supplemental material and provide a GitHub link for our implementation.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include them in the supplemental material and GitHub implementation.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We observe different stability properties for FL frameworks with different ML backends, which makes it inconsistent to report error bars across all frameworks. However, we verified that, the conclusions we present in the paper are consistent in different runs with different random seeds, which can be reproduced with the codebase we provide. (See Section 5)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] It is mentioned in the cited source.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We only use existing datasets in our work.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We only use existing datasets in our work.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]