PyFed: extending PySyft with N-IID Federated Learning Benchmark

Houda Bouraqqadi[†], Ayoub Berrag[†], Mohamed Mhaouach[†], Afaf Bouhoute[†,*], Khalid Fardousse[†], Ismail Berrada[†]
[†] Sidi Mohamed Ben Abdellah University, Faculty of Science, Fez, Morocco
[‡] Mohammed VI Polytechnic University, Benguerir, Morocco

Abstract

Federated Learning (FL) is an emerging learning paradigm that enables collaborative model training, across multiple devices using decentralized data, allowing each device to keep the privacy of its local data. Heterogeneity of data distributions is an inherent characteristic of FL. Generally, data samples across user devices are Not-Independent and Identically Distributed (N-IID), making learning in federated settings a challenging task. In this paper, we aim to contribute to FL benchmarking by introducing PyFed, an open source and scalable simulation framework of federated settings, supporting N-IID data. PyFed is fully compatible with PySyft, the secure and private framework for deep learning. It includes a set of benchmark datasets and implements different types of N-IID data distributions. PyFed also provides a set of implementations that can be used as reference for FL development.

Keywords: Federated Learning, N-IID, PySyft

1. Introduction

The recent years have witnessed an increase in the number of intelligent personal devices (e.g., smartphones, wearable devices), resulting in a rapid growth of data [1]. The exponential rate of data generated by devices has boosted the application of machine learning models for developing more intelligent and predictive applications. Traditionally, these applications use central data centers to which user data are transferred for processing and model training. This centralized architecture has many downsides: transferring large data from user devices to the data center limits real-time learning and raises privacy issues. To address these challenges, Federated Learning (FL) has emerged [2–5]. It is a machine learning paradigm that allows collaborative model training across multiple devices using decentralized data. Formally speaking, the FL optimization problem can be defined as the minimization of the following empirical risk, where w is the model parameter, N is the number of client devices holding their local data samples, P_k is the set of data sample indices on client k, $n_k = |P_k|$ is the number of data samples, $n = \sum_{k=1}^{N} n_k$ is the total number of data samples, and f_i is the local loss function.

$$\min_{w\in\mathbb{R}^d} f(w) = \min_{w\in\mathbb{R}^d} (\sum_{k=1}^{N} \frac{n_k}{n} F_k(w)) \quad \text{with} \quad F_k(w) = \frac{1}{n_k} \sum_{i\in P_k} f_i(w)$$

One of the core challenges of FL is to manage the heterogeneity of data distributions among devices [6, 7]. Typically, data samples across user devices are Not-Independent and Identically Distributed (N-IID), which can affect the convergence of FL. To support the development of FL, multiple frameworks and libraries have been developed among which we mention PySyft [8]. In fact, most existing benchmarks ([9–13]) lack of standardized FL algorithm implementations and support of diverse FL configurations, making comparing performance in various cases of N-IID data a difficult task [14].

[*]afaf.bouhoute@usmba.ac.ma

In this paper, we aim to address the lack of benchmarking frameworks especially those dealing with N-IID data, by extending the popular library PySyft, with PyFed an FL benchmarking suite. PyFed assembles five representative datasets and implements various N-IID data types and aggregation methods. A comparison of PyFed and other existing FL benchmarks is presented in Table 1. The main contributions of the paper are:

- An extension of the widely used PySyft library to support FL benchmarking, focusing especially on the N-IID challenge.
- An extensive evaluation of some solutions for dealing with N-IID data, using five famous datasets.

Table 1. Comparison of federated learning Benchmarks

Name	Various ML models	Various datasets	N-IID types simulation	Handling N-IID datasets
LEAF [9]	✓	✓	✗	✗
Street Image [11]	✓	✗	✗	✗
Edge AIBench [12]	✓	✓	✗	✗
OARF [10]	✓	✓	✗	✗
Liu et al [13]	✗	✗	✓	✗
PyFed (our work)	✓	✓	✓	✓

The remainder of this paper is organized as follows. Section 2 presents the methodology of the proposed benchmarking framework PyFed. Section 3 presents the performed experiments and discusses the results. Section 4 concludes the paper and gives some perspectives.

2. PyFed Methodology

2.1. N-IID data

In real world scenarios, data can be N-IID in several ways [6, 13, 14]. Let us assume that, for a learning task T, the feature set is x, the labels are y, and P_i is the data distribution of the client i. Thus, the taxonomy of the N-IID data regimes that can arise for any client partitioned dataset, can be classified into various classes:

- Covariate shift: the marginal distributions $P_i(x)$ may vary across clients, even if $P_i(y|x) = P_j(y|x)$, for all clients i and j.
- Prior probability shift: the marginal distributions $P_i(y)$ may vary across clients, even if $P_i(x|y) = P_j(x|y)$, for all clients i and j.
- Concept shift:
  (1) Same label, different features: the conditional distribution $P_i(x|y)$ may vary across clients even if even if $P_i(x) = P_j(x)$, for all clients i and j.
  (2) Same features, different label: the conditional distribution $P(y|x)$ may vary across clients even if even if $P_i(y) = P_j(y)$, for all clients i and j.
- Unbalancedness: the amounts of local data held by clients may differ considerably.
- Inter and intra-partition correlation: data across multiple client devices can be correlated, or different partitions of the data of one client are dependent.

2.2. PyFed Architecture

As illustrated in Figure 1, PyFed adopts a modular architecture, interfacing well-known datasets, supporting IID and N-IID data distributions and establishing a methodology for model evaluation and result reproduction. PyFed also extends PySyft by adding additional modules to support N-IID data. The main modules of PyFed are presented below. More details about the package contents are available on the PyFed github [15].

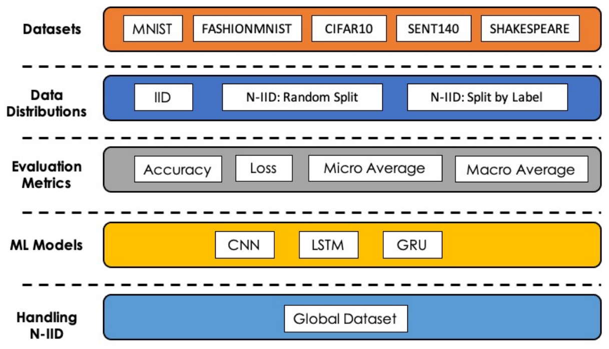


Figure 1. PyFed Overview.

Datasets. This module includes the datasets and their respective pre-processing scripts. It also contains additional scripts (data loader) for loading and splitting the data. Currently, PyFed includes five datasets, for both image and text classification: *CIFAR10* [16], *MNIST* [17], *Fashion-MNIST* [18], *Sentiment140* [19], *Shakespeare* [20].
Data splitting. N-IID data is simulated into 4 types, according to the configurations mentioned above, categorized in two categories, namely random split and label split. Note that, a multitude of options/parameters allow customising the split of these two categories.

- **Random split (unbalancedness).** In this class split, we distribute the samples of the dataset randomly, so the workers (data holders) may (not) have all classes but with different distributions.
- **Split by label (covariate and concept shifts).** This N-IID split is based on labels, so we can specify the different classes that each worker will have. We define 3 types depending on how the workers with the same classes will share the samples.
  (1) **Type 0**: The workers that have the same labels, share also the same samples.
  (2) **Type 1**: The workers share the samples of the same class randomly.
  (3) **Type 2**: The workers do not share any samples of the same class, i.e even if two workers have the same class, they will never share the same samples.

Models. The module implements the ML classification models. Given the specificity of the datasets, the following neural networks were implemented: Convolutional Neural Networks (CNN) for image datasets and Recurrent Networks (GRU) for text datasets. The module also includes a sub-package named "metrics", which contains the main used metrics (Accuracy, Loss and Micro/Macro average), as well as a notebook for result visualization.
Aggregation. This module includes the aggregation methods for FL. PyFed implements the most popular FL aggregation algorithms "FedAvg" [4].
Utils. This module contains a collection of general-purpose utility functions, including those for experiment set up. Experiments can be configured by assigning additional arguments, either manually in the command line or by using a YAML configuration file.
Run. This module contains scripts for starting the workers and the training process.

3. Experiments

3.1. Experimental Setup

We implement 3 different neural networks depending on the datasets and the ML task: CNNs for image classification and LSTM/GRU for text classification. Details of the model architectures for each dataset can be found [15]. For each dataset, the training data was distributed to K=100 clients, while the testing data was kept for performance evaluation. The fraction of data and labels held by each client was randomly determined and is set in the configuration files. The training data is split among the clients in IID and N-IID settings. N-IID data partitions are created by splitting the datasets randomly and by label. Different types of partitioning using labels are considered, depending on whether the clients holding the same classes share or not the same samples. In each round of FL, the fraction of participating clients is set to $C = 0.1$ to select a maximum of $K \times C = 10$ clients.

Table 2. Benchmark Results for IID distributions (baseline)

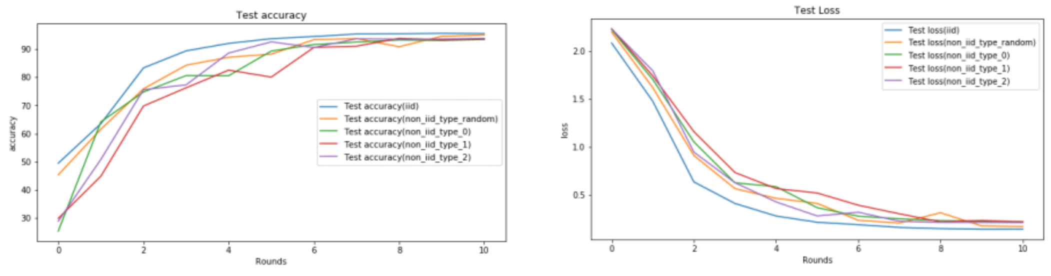
Datasets	*CIFAR10*	*Fashion-MNIST*	*MNIST*		*Sent140*	*Shakespeare*
Model	CNN	CNN	CNN	CNN batch	LSTM	GRU
Accuracy	67	86.81	95.63	96.33	65.45	50.36
Loss	0.8043	0.368	0.1384	0.1154	0.8345	1.2452

Table 3. Benchmark results for N-IID distributions

Dataset	**Model**	**N-IID (Label split)**						**N-IID (Random split)**	
		Type 0		**Type 1**		**Type 2**			
		Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
CIFAR10	CNN	66.78	0.8132	65.89	0.8453	65.45	0.8464	66.89	0.8121
Fashion-MNIST	CNN	85.36	0.4029	85.8	0.3956	85.42	0.4009	86.57	0.3727
MNIST	CNN	93.45	0.2171	93.88	0.2164	93.84	0.2086	95.04	0.1671
	CNN(with BN)	94.25	0.1902	94.74	0.1771	94.76	0.1884	96.09	0.13
Sent140	LSTM	64.4	0.9244	64.23	0.9445	65.78	0.8123	65.1	0.8663
Shakespeare	GRU	48.26	1.3452	48.76	1.2052	45.23	1.7452	49.46	1.2952

3.2. **Results and Discussion**

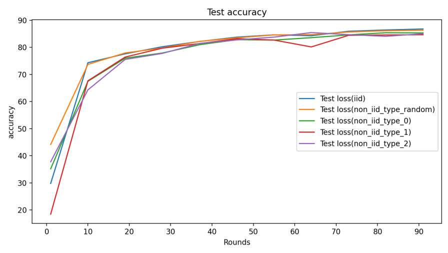
The main results with the IID setting are presented in Table 2. The results show high classification accuracy scores achieved on *MNIST* and *Fashion-MNIST* datasets and average accuracy scores on *CIFAR10*, *Sent140* and *Shakespeare*. These results will serve as a baseline to which the N-IID results will be compared. The results for N-IID settings are presented in Table 3. In accordance with the existing works, the results show limited performance with the N-IID setting compared to the results obtained with the IID setting. Similar performance is achieved by all three types of N-IID data splits with a slight improvement in the random split. Some of the obtained results are also illustrated using the accuracy and loss curves for the *MNIST* and *Fashion-MNIST* datasets (Figure 2 and Figure 3).



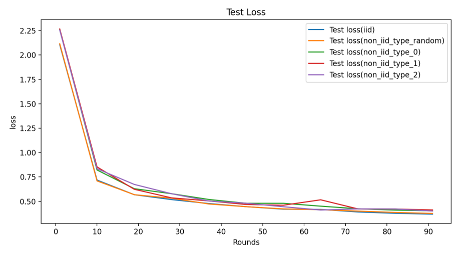
(a) Test accuracy *(b)* Test loss

Figure 2. Test accuracy and loss graphs for *MNIST* dataset

These latter show the improvement of the test accuracy/loss after each round. Comparing the plots of the different data distributions on *MNIST* (Figure 2), we figure out that, compared to IID and random split, the accuracy with the split by label types starts with lower accuracy but converges to the same values after 10 rounds. Slower improvement is noticed when clients share the same class's samples randomly (the red curve in Figure 2). On the other hand, the improvement was faster with the split of type 0, where the same samples are shared among clients having the same classes. The test accuracy/loss curves for the *Fashion-MNIST* are illustrated in Fig.3. Compared to *MNIST*, we notice that accuracy/loss evolution for *Fashion-MNIST* over rounds is more stable for all types of data splits. Convergence was achieved after 90 rounds. Figure 4 compares the micro average train and test losses for the two datasets.

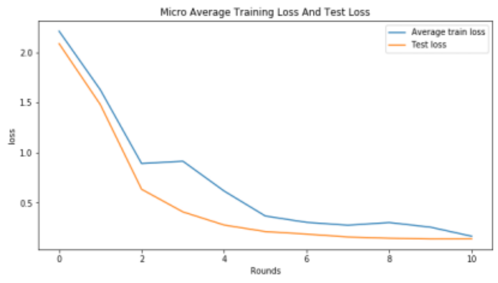


(a) Test accuracy

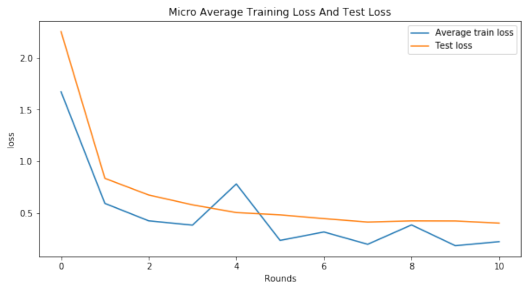


(b) Test loss

Figure 3. Test accuracy and loss graphs for *Fashion-MNIST* dataset



(a) *MNIST* Micro average Loss



(b) *Fashion-MNIST* Micro average Loss

Figure 4. Micro average losses for *MNIST* and *Fashion-MNIST* datasets

As indicated in Section 2.2, one way to cope with N-IID data is to share a small set of IID data among clients (global dataset) [7]. Considering the poor performance achieved with studied datasets, we choose to implement this strategy with the *Shakespeare* dataset. Adding the global dataset increased the client data by 15%. The obtained results are presented in Table 4. Consistently with the results of [7], the classification accuracy was improved by up 4.66%.

Table 4. Accuracy values of GRU on SHAKESPEARE before and after add global dataset

N-IID type	**Accuracy%**	**Accuracy (add global dataset)**	**Improvement rate**
Random split	49,46	51,45	1,99
Split by label type 0	48,26	50,79	2,53
Split by label type 1	48,76	50,12	1,36
Split by label type 2	45,23	49,89	4,66

4. Conclusion

In this paper, we introduce PyFed a simulation framework for federated settings, developed based on the most widely used deep learning library PySyft. PyFed provides a set of datasets, implementations of ML models and various way of N-IID data splitting. Various experiments were carried out with our proposed framework, using IID and N-IDD data (3 types of data splits were simulated), different datasets and ML models. Moreover, one of the existing methods to handle N-IID data was implemented and the results were matched. As future work, we aim to expand our framework by adding other aggregation methods.

References

[1] *Data volume of internet of things (IoT) connections worldwide.* https://www.statista.com/statistics/1017863/worldwide-iot-connecteddevices-data-size/. Accessed: 2021-02-17.

[2] J. Konečný, B. McMahan, and D. Ramage. “Federated optimization: Distributed optimization beyond the datacenter”. In: *arXiv preprint arXiv:1511.03575* (2015).

[3] B. McMahan and D. Ramage. “Federated learning: Collaborative machine learning without centralized training data”. In: *Google Research Blog* 3 (2017).

[4] 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*. PMLR. 2017, pp. 1273–1282.

[5] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas. “Federated learning of deep networks using model averaging”. In: *arXiv preprint arXiv:1602.05629* (2016).

[6] 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”. In: *arXiv preprint arXiv:1912.04977* (2019).

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. “Federated learning with non-iid data”. In: *arXiv preprint arXiv:1806.00582* (2018).

[8] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach. “A generic framework for privacy preserving deep learning”. In: *arXiv preprint arXiv:1811.04017* (2018).

[9] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. “Leaf: A benchmark for federated settings”. In: *arXiv preprint arXiv:1812.01097* (2018).

[10] S. Hu, Y. Li, X. Liu, Q. Li, Z. Wu, and B. He. *The OARF Benchmark Suite: Characterization and Implications for Federated Learning Systems.* 2020. arXiv: 2006.07856 [cs.LG].

[11] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang. “Real-world image datasets for federated learning”. In: *arXiv preprint arXiv:1910.11089* (2019).

[12] T. Hao, Y. Huang, X. Wen, W. Gao, F. Zhang, C. Zheng, L. Wang, H. Ye, K. Hwang, Z. Ren, et al. “Edge AIBench: towards comprehensive end-to-end edge computing benchmarking”. In: *International Symposium on Benchmarking, Measuring and Optimization*. Springer. 2018, pp. 23–30.

[13] L. Liu, F. Zhang, J. Xiao, and C. Wu. “Evaluation Framework For Large-scale Federated Learning”. In: *arXiv preprint arXiv:2003.01575* (2020).

[14] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons. “The non-iid data quagmire of decentralized machine learning”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4387–4398.

[15] *PyFed library.* https://github.com/okazaki0/PyFed. Accessed: 2021-02-17.

[16] A. Krizhevsky et al. “Learning multiple layers of features from tiny images”. In: (2009).

[17] Y. LeCun. “The MNIST database of handwritten digits”. In: *http://yann. lecun. com/exdb/mnist/* (1998).

[18] H. Xiao, K. Rasul, and R. Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[19] A. Go, R. Bhayani, and L. Huang. “Twitter sentiment classification using distant supervision”. In: *CS224N project report, Stanford* 1.12 (2009), p. 2009.

[20] *William Shakespeare. The Complete Works of William Shakespeare.* http://www.gutenberg.org/ebooks/100. Accessed: 2021-02-17.