OpenFed: A Comprehensive and Versatile Open-Source Federated Learning Framework

Dengsheng Chen[1], Vince Junkai Tan[2], Zhilin Lu[3], Enhua Wu[4,5,6], Jie Hu[5,6*†]
[1] Meituan [2] Bytedance Inc. [3] Tsinghua University [4] University of Macau
[5] State Key Lab of Computer Science, ISCAS [6] University of Chinese Academy of Sciences
chendengsheng@meituan.com, vince.tan.jun.kai@gmail.com, luzl18@mails.tsinghua.edu.cn,
ehwu@um.edu.mo, hujie@ios.ac.cn

Abstract

Recent developments in Artificial Intelligence techniques have enabled their successful application across a spectrum of commercial and industrial settings. However, these techniques require large volumes of data to be aggregated in a centralized manner, forestalling their applicability to scenarios wherein the data is sensitive or the cost of data transmission is prohibitive. Federated Learning alleviates these problems by decentralizing model training, thereby removing the need for data transfer and aggregation. To advance the adoption of Federated Learning, more research and development needs to be conducted to address some important open questions. In this work, we propose OpenFed, an open-source software framework for end-to-end Federated Learning. OpenFed reduces the barrier to entry for both researchers and downstream users of Federated Learning by the targeted removal of existing pain points. For researchers, OpenFed provides a framework wherein new methods can be easily implemented and fairly evaluated against an extensive suite of benchmarks. For downstream users, OpenFed allows Federated Learning to be plugged and play within different subject-matter contexts, removing the need for deep expertise in Federated Learning. The source code of OpenFed is publicly available online at https://github.com/FederalLab/OpenFed.

1. Introduction

Recent developments in Artificial Intelligence (AI) and Machine Learning (ML) techniques have advanced the state-of-the-art across many application domains [28]. In particular, Deep Learning (DL) has revolutionized the field and deep neural networks are now the defacto standard for computer vision [11], natural language processing [26], audio and speech processing [41], and reinforcement learning [4], among others. In the broader historical context of AI research, deep learning simply represents the next milestone in the Bitter Lesson [1] - "general methods that leverage computation are ultimately the most effective". Deep learning enables today's vast computational resources to be unleashed on equally vast amounts of data, with the best-known example perhaps being ImageNet's 14 million hand-annotated images [29]. The typical setup involves centralizing the said data onto a database, which is connected to a high-performance server or cluster of servers. The training process can then take place, and the deep neural network is exposed to these examples and their corresponding desired outputs and therefore "learns" the assigned task.

While this paradigm has proven to be broadly effective, there are some use cases where it is not straightforwardly applicable [3, 22]. The very first step of centralizing the data onto a database can be challenging or impossible for two main reasons: the data is personal or confidential and therefore the user would not consent to it being transmitted, or the data originates from edge devices and transmission costs are prohibitive.

Federated Learning (FL) is a family of ML techniques proposed to address these challenges [24]. Rather than centralizing the data and then training the model on a server, in FL, model training is decentralized to the data sources themselves, obviating data transfer entirely [32, 39]. Although FL has already been applied successfully in some industrial and commercial settings, it is in fact still an area of active research. Many open sub-problems need to be addressed before FL can be considered ready for broader application [21].

As a research topic, FL suffers from several stumbling blocks. Firstly, fair comparisons are difficult to perform. Much research is built upon synthetically generated datasets. For these datasets, typically only the statistical

*Corresponding author
†Project lead

[1] http://www.incompleteideas.net/IncIdeas/BitterLesson.html

properties can be practically reported. This introduces a significant element of randomness to experimental results. Secondly, in applied ML research, it is common to make use of existing frameworks which may not have been built with FL in mind. Because these frameworks are industry standards for their respective domains, it is not feasible to abandon them just to make use of FL techniques. Finally, to be effective in FL research requires a broad skill set. Whether to replicate existing work or to propose new methods, researchers need strong software engineering skills to implement the (pseudo) distributed setting that FL takes place in, a deep understanding of the ML principles that undergird FL techniques, as well as subject-matter expertise in the domain in question.

In this work, we propose OpenFed, a software framework that targets the stumbling blocks identified above in order to accelerate FL research and development. For FL researchers, OpenFed provides comprehensive implementations of existing methods and their corresponding performances across a suite of standardized datasets, thereby enabling convenient and fair comparisons for newly proposed methods while reducing the effort required to implement them. For FL users, OpenFed's support for common third-party frameworks allows state-of-the-art FL techniques to be integrated into practical applications without the need for practitioners to fully understand the underlying implementation.

2. Aims and contributions

For both academic and industrial applications of FL, it is essential that the software framework used is versatile and flexible. There already exist several influential FL frameworks, each with its own focus. However, these frameworks have left some challenges unmet.

Firstly, the full diversity of FL topologies is not commonly supported. For example, TensorFlow Federated [16], PySyft [42], and LEAF [6] support only the centralized topology. Furthermore, these libraries do not provide convenient interfaces for the flexible exchange of auxiliary information or the customization of the training procedure, thereby limiting the algorithmic innovations possible.

Secondly, it is challenging to perform fair comparisons. Newly proposed FL algorithms are often implemented based on different libraries and dataset configurations. As a result, it is difficult for researchers to compare the performances of the algorithms fairly.

Finally, it is not easy to transition FL algorithms to applied scenarios. Existing FL frameworks have not sufficiently catered for downstream integration. For instance, it is not trivial to employ even the famous FedML [15] to train models from well-established libraries like HuggingFace [33].

In order to address the above challenges, we designed a novel FL framework named OpenFed based on the PyTorch [27] ecosystem. OpenFed provides an expandable toolkit for FL algorithm development across diverse topologies. Comparisons between OpenFed and existing mainstream FL libraries are given in Table 1. The main contributions of the OpenFed framework are listed as follows.

Diverse topologies OpenFed provides powerful automatic topology analysis and construction tools. By introducing the concept of a federated group, we are able to decompose an entire FL topology into its atomic units. Novel FL algorithms for split, vertical, hierarchical, or decentralized topologies can then be implemented as though they were the standard centralized topology. In addition, OpenFed provides an interface for auxiliary information to be exchanged across compute nodes.

Comprehensive and standardized FL algorithms and benchmarks OpenFed provides standardized datasets, algorithms, and benchmarks. Researchers proposing novel algorithm designs can conveniently and comprehensively compare their new ideas against existing solutions. OpenFed also provides rich configuration possibilities (e.g. server/client optimizer, sampling strategy of partial-activated clients, non-i.i.d. distribution of dataset partition), allowing researchers to better validate how well different algorithms generalize to specific situations.

User-friendly API design Mainstream deep learning (DL) frameworks, such as PyTorch and TensorFlow, lack ready-made APIs for FL. OpenFed carefully inherits the API design from the PyTorch library, which is commonly used by the DL community. This reduces the barrier to entry for the larger DL community to participate in FL research and development. Furthermore, OpenFed provides rich built-in features including weighted gradient descent, federated averaging, diverse data augmentation, local early stopping, etc. These FL modules offer an immersive development experience to OpenFed users.

Third-party library support The relative maturity of DL has resulted in the rise and success of open-source secondary libraries. These libraries provide state-of-the-art implementations for their respective domain areas. For example, HuggingFace [33] for natural language processing, MONAI [9] for medical analysis, and MMCV [8] for computer vision. However, few of this support FL natively. Furthermore, because these libraries are increasingly widely adopted, it is unlikely for newly proposed libraries that offer only FL as an advantage to gain much traction. Therefore, OpenFed is designed to be plug-and-play compatible with these third-party libraries.

		CrypTen [13]	FATE [35]	PaddleFL [23]	PySyft [42]	FedML [15]	FFL-ERL [31]	LEAF [6]	TensorIO [10]	TFF [16]	OpenFed (ours)
Diverse computing paradigms	Standalone simulation		✓	✓	✓	✓		✓		✓	✓
	Distributed computing	✓	✓	✓	✓	✓				✓	✓
	On-device training					✓	✓		✓		✓
Flexible & generic API design	Message flow					✓					✓
	Customize data distribution							✓			✓
	Unified paradigm		✓	✓	✓					✓	✓
	Third-party support				✓						✓
Topology customization	Split			✓	✓	✓					✓
	Vertical		✓	✓		✓					✓
	Hierarchical										✓
	Decentralized					✓					✓
	Centralized	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Federated optimizer	Aggregation customization					✓					✓
	Gradient accumulation										✓
	Parameter penalization					✓					✓
	State synchronization										✓
Benchmarks	Computer vision		✓	✓	✓	✓		✓	✓	✓	✓
	Natural language processing		✓	✓		✓		✓		✓	✓
	Reinforcement learning										✓
	Medical analysis				✓						✓

Table 1. Comparison between OpenFed and existing FL frameworks.

3. Architecture design

As depicted in Figure 1, the FL workflow is based on the distributed computing setup [38], with a server and a number of local devices. Since user data never leaves the local device, the FL workflow inherently protects personal privacy.

Each local device downloads a copy of the most recent global model. This model can then be independently personalized using private data, resulting in improved performance for the user. These personalized models are then uploaded from local devices to the main server. An aggregator at the server fuses all the personalized models into a newer and better-performing global model. Similarly, the on-device personalized model is also continuously improving with every round of this FL training loop. Note that the prediction task is typically also independently conducted on each local device.

We designed the OpenFed architecture based on the FL workflow described earlier. The crucial architectural features of OpenFed are as follows.

Flexible federated groups abstraction The connections between servers and local devices in an FL system can be complicated. Furthermore, the effort to implement these connections is often orthogonal to the algorithmic work that researchers primarily deal with. Therefore, OpenFed includes an automatic topology analysis strategy in which different FL connections are expressed by the same atomic unit which is called a federated group. As Figure 2 shows, the proposed strategy has three levels. Device nodes are added to the topology graph based on their connection modes. The topology graph is then parsed into a set of federated groups. The key insight is that every topology - even ones that appear highly irregular or complex - can be decomposed into a set of simpler groups that are individually just centralized topologies. By this mechanism, OpenFed enables centralized FL algorithms to be applied to other topologies.

The robust distributed training scheme FL algorithms naturally assume a distributed training scenario since the servers and the devices are physically separated. OpenFed offers an interface to simulate dozens of federated groups on a single machine. This lowers the resource requirement to participate in and contribute to FL research. Furthermore, the distributed design in OpenFed allows complicated auxiliary information exchange within each federated group. For example, algorithmic hyper-parameters like learning rate, the FL round number, training instance, etc. can be eas-

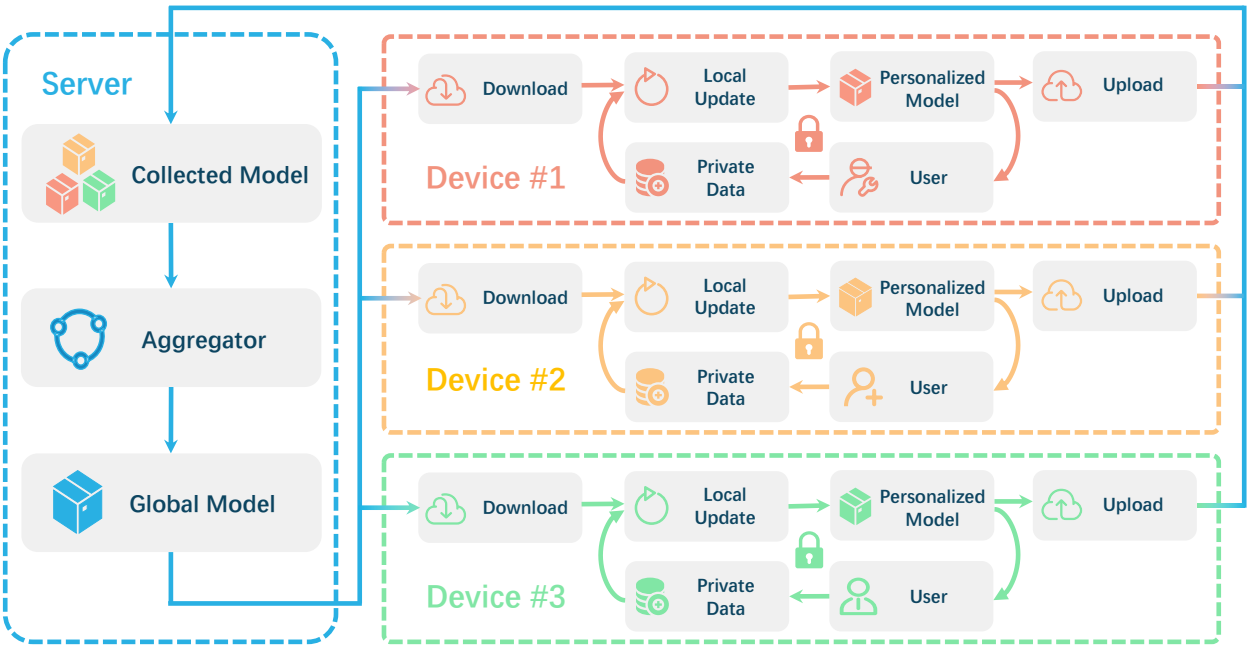


Figure 1. An illustration of the FL workflow based on the OpenFed framework. A server collects personalized models from different users, and the aggregator combines these to create a global model. This model is sent to all the users and is separately personalized on each device. The cycle then repeats, making up the main body of the FL workflow. Note that private data is only ever accessed by the local device to guarantee user privacy.

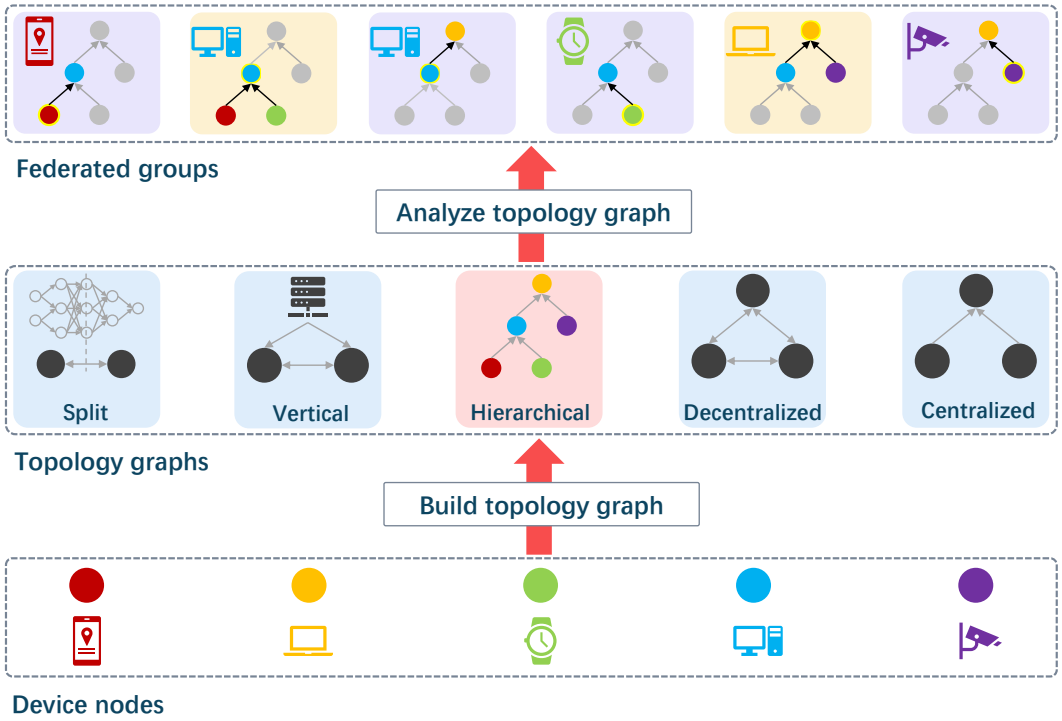


Figure 2. Three-level abstraction of the OpenFed design. Bottom level: nodes abstracted from various physical devices. Middle level: topology graph generated from the connection between different nodes. Top-level: federated groups deriving from the specific topology. Although one device might belong to several federated groups, communication among different devices is limited to within each group.

ily shared. This enables the implementation of even highly exotic algorithms.

Specialized functional modules The OpenFed architecture is composed of several functional modules. Researchers can pick and choose corresponding modules as required for their experiments. By using OpenFed's predefined modules like the aggregator, collaborator, pipe and optimizer, FL research can be accelerated. In addition, security and privacy-related functional modules are also supported in OpenFed, implementing defenses against common attacks like data and model poisoning [12, 25].

4. Real-life application scenarios and benchmarks

FL has already been employed in many different applications; here we discuss a few representative scenarios, as illustrated in Figure 3. These scenarios also motivate OpenFed's support for third-party libraries commonly used in industry. Specifically, OpenFed-CV and OpenFed-RL address the perception and decision-making components respectively for self-driving systems. OpenFed-Medical supports common medical use cases. Finally, OpenFed-NLP can be used for the text-based recommendation systems described.

Self-driving Self-driving systems can typically be broken down into two major components: perception, and decision-making [3]. The perception system is generally further divided into subsystems responsible for self-localization, static obstacle mapping, moving obstacle detection and tracking, road mapping, and traffic signal detection, and recognition, among others. The decision-making system likewise comprises tasks such as route planning, path planning, behavior selection, motion planning, and control. These tasks are highly complex and each requires not only large volumes of data but also data of a sufficiently wide diversity to account for the natural diversity of street scenes.

A promising source of data is the fleet of vehicles already sold - in the course of their everyday operation, these vehicles would collectively accumulate data approaching the scale and variety needed. Due to privacy concerns and the high costs of transmitting rich multimedia data, the traditional model of data aggregation and centralized training is unfeasible. FL addresses these problems directly - for example, a previous case study demonstrated that using FL for the steering wheel prediction problem reduces bandwidth by 60% and training time by 70% with no loss to the model performance [40].

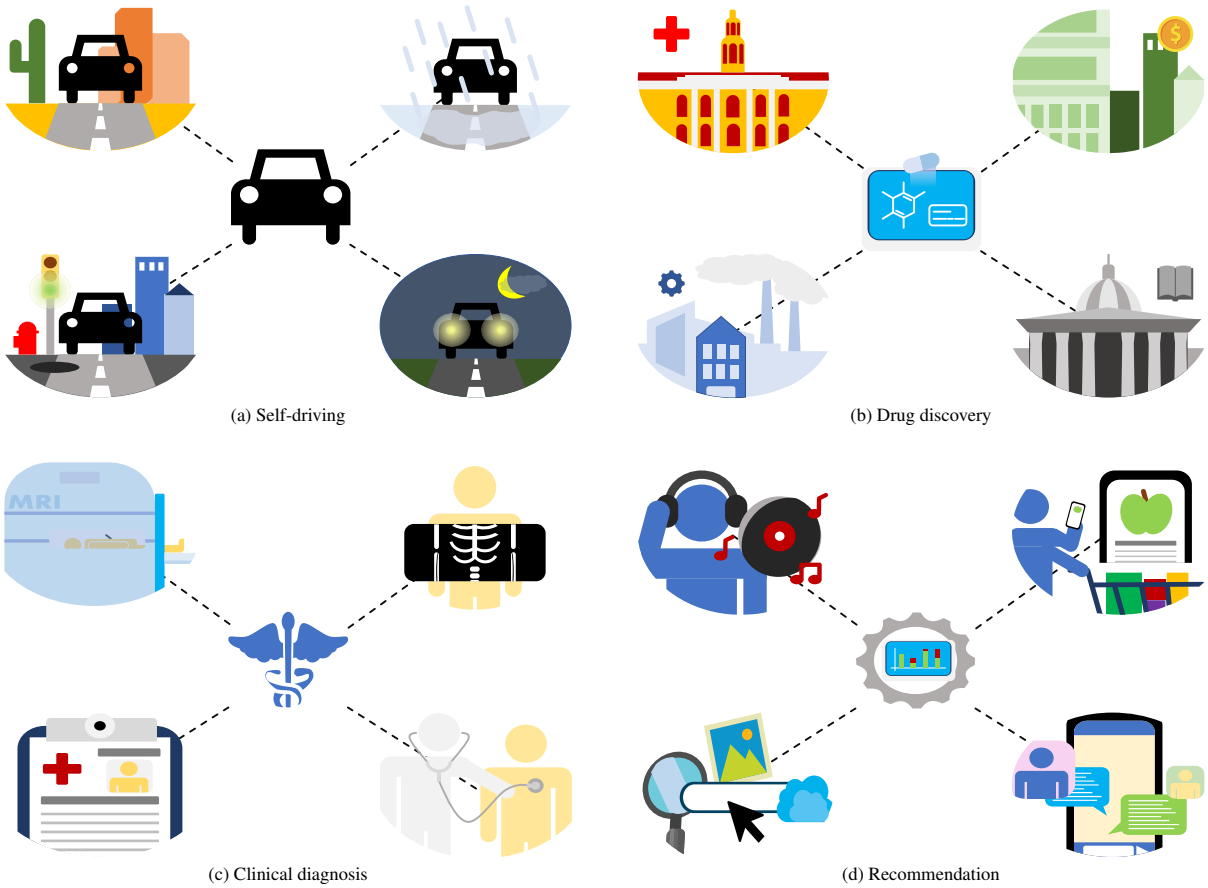
Drug discovery According to Pharmaceutical Research and Manufacturers of America, it takes on average 10 years and $2.6 billion for a new medicine to reach the market [1]. One part of this process that machine learning can accelerate is drug discovery. For example, quantitative structure-activity relationship (QSAR) is a machine learning method for predicting the relationship between chemical structures and resultant biological activities. One challenge in improving the performance of QSAR models is data availability. Data from different institutions cannot be freely shared and aggregated due to commercial and legal reasons. FL has been demonstrated to work well for QSAR [7, 34], and can overcome the data availability problem.

Furthermore, FL for drug discovery has moved beyond the theoretical. The MELLODDY project is a federated learning platform to accelerate drug discovery for 10 major pharmaceutical companies, allowing them "for the first time to collaborate in their core competitive space" [2].

Clinical diagnosis Clinical diagnoses are often made with context provided by a diverse range of sources, from patient medical history to different types of imagery. This use of multi-modal information is required to give a holistic assessment of the patient's condition. Vertical FL, also known as Heterogeneous FL, allows different types of information across different data sources to be combined securely and with privacy protection. This is critical for such a system to be deployed in practice, as medical data and records are highly sensitive.

Previous work has demonstrated the need for more diverse data in this clinical diagnosis setting [43]. Medical data can be highly heterogeneous across institutions - it was found that models trained using data from single institutions perform substantially worse on test examples from other institutions. By using FL, models can benefit from a greater diversity of cases and thereby perform better even on validation data from their own institution [30].

Recommendation FL has been successfully applied to recommendation systems such as browser history suggestion [14], keyboard query suggestion [36], and mobile keyboard prediction [20]. The mobile keyboard prediction study provides a rare real-world large-scale comparison between an FL system and the best-performing centralized alternative. The centralized model benefited from the relative maturity of its traditional machine learning setup - including best practices for the algorithm design, parameter selection, and training methodology. However, it was limited to users who had opted in and allowed their data to be recorded. In this setup, it was found that the FL model did not simply match the centralized model performance, but in fact, exceeded it in terms of both the top-1 and top-3 prediction metrics by 1%.



(a) Self-driving (b) Drug discovery

(c) Clinical diagnosis (d) Recommendation

Figure 3. Real-life use-cases of FL. In (a), a better self-driving system can be trained by combining the varied experiences of individual vehicles across the full diversity of road scenes. In (b), FL enables pharmaceutical companies to collaborate in an otherwise highly competitive space, thereby accelerating drug discovery. In (c), more accurate clinical diagnoses can be made by considering multiple modalities of information. In (d), FL can make for higher quality recommendations through on-device personalization.

5. Discussion

We present OpenFed, a comprehensive and versatile open-source framework for FL with a variety of benchmarks across diverse federated tasks. In this section, we further discuss the strengths and weaknesses of OpenFed as well as future work directions from the perspective of facilitating academic and industrial research and development.

Comparison with prior FL frameworks As shown in Table 1, although there exist other FL frameworks, most of them focus on specific uses and are not meant for general purposes. A gap exists for a framework to accelerate underlying FL algorithm research. OpenFed bridges this gap. The main advantages of OpenFed compared to existing frameworks are summarized as follows:

- **Inherited design structure.** In general, most existing frameworks are based on either TensorFlow or PyTorch. However, they typically do not follow the programming structure of their underlying frameworks. Instead, they propose their own proprietary designs. Uniquely, our implementation completely follows the design philosophy of our base, PyTorch. The relatively large DL community should thus find OpenFed to be immediately familiar. Researchers using OpenFed can thus concentrate on exploring and implementing novel algorithms, while developers will find it easy to apply these algorithms to their tasks.

- **Modular algorithm implementations.** Most frameworks support only the simplest FL optimization algorithm, FedAvg [17, 18]. Although some frameworks also include other more complex algorithms, the implementations tend to be more functional. In comparison, the OpenFed design clearly demarcates where each step of the FL workflow begins and ends. We summarize and standardize the workflow by describing it with four independent steps, i.e. parameter aggregation, gradient accumulation, parameter penalization,

and state synchronization. Different FL algorithms can thus be clearly implemented by making specific adjustments within only the required steps. This philosophy not only facilitates researchers to realize standardized and maintainable code but also makes explicit where the differences are between different algorithms.

- **Flexible topology support.** Most frameworks support only centralized topologies and provide limited support for some of the more complex ones. Research on different topologies is an important topic in FL. Our concept of the federated group allows OpenFed to flexibly implement any topology.

Future work As FL continues to gain traction, we anticipate that new features will become necessary for OpenFed to continue to meet the requirements of academia and industry.

- **Topology benchmarks.** At present, all benchmarks default to the centralized topology for algorithm evaluation, and no relevant large-scale experiments have been conducted on the impact of different topologies on FL performance. We will continue to promote this work as OpenFed matures.

- **Industry deployment.** OpenFed runs in a Python runtime environment, which makes it challenging for deployment on resource-constrained devices. Therefore, more efficient runtime environments such as C/C++ must be supported. Furthermore, we will continuously increase the number of third-party libraries that OpenFed supports [5].

- **Cryptography.** Although there are many mature symmetric/asymmetric encryption algorithms in the communication field, these algorithms are often too complex to encrypt large amounts of data and consume too much computation. An open problem is therefore to design an effective encryption algorithm under the condition of limited computing resources according to the characteristics of FL.

- **Poisoning and adversarial attacks.** The data used for training in FL often involves personal and private information. There has been researched work demonstrating vulnerabilities in FL, for example, it has been shown that the content of training data can be inferred (to some degree) through the exposed gradients [19, 37]. OpenFed requires more defense mechanisms and algorithms to counteract these attacks.

Limitations. With the growing importance of privacy protection and data security, FL has become increasingly adopted across different fields. However, there is still generally a performance gap between FL and traditional distributed learning. As such, in scenarios where the data is not particularly sensitive, researchers and developers still prefer traditional distributed learning. We hope that OpenFed can address the pain points of FL research and stimulate the enthusiasm of researchers new and experienced, ultimately narrowing the performance gap.

6. Conclusion

We present a free, open-source software framework for FL and end-to-end encrypted inference, which we showcased in several relevant real-life case studies. OpenFed is comprehensive and versatile and can be used both for research and development as well as for commercial and industrial applications. Ongoing work will enable the large-scale deployment of OpenFed, the validation of our findings on diverse cross-institutional datasets, and further the widespread utilization of our framework.

References

[1] Biopharmaceutical research & development: The process behind new medicines. 2015. 5

[2] Melloddy: Machine learning ledger orchestration for drug discovery. 2019. 5

[3] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020. 1, 5

[4] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. 1

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019. 7

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018. 2, 3

[7] Shaoqi Chen, Dongyu Xue, Guohui Chuai, Qiang Yang, and Qi Liu. Fl-qsar: a federated learning-based qsar prototype for collaborative drug discovery. *Bioinformatics*, 36(22-23):5492–5498, 2020. 5

[8] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv, 2018. 2

[9] MONAI Contributors. MONAI: Medical Open Network for AI. https://monai.io/, 2020. 2

[10] Tensor/IO Contributors. Tensor/IO: a lightweight, cross-platform library for on-device machine learning. https://github.com/doc-ai/tensorio, 2019. 3

[11] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, and Xin Li. Computer vision algorithms and hardware implementations: A survey. *Integration*, 69:309–320, 2019. 1

[12] Yan Feng, Xue Yang, Weijun Fang, Shu-Tao Xia, and Xiaohu Tang. Practical and bilateral privacy-preserving federated learning. 2020. 5

[13] David Gunning, Awni Hannun, Mark Ibrahim, Brian Knott, Laurens van der Maaten, Vinicius Reis, Shubho Sengupta, Shobha Venkataraman, and Xing Zhou. Crypten: A new research tool for secure machine learning with pytorch. *Blog post available at https://ai. facebook. com/blog/crypten-a-new-research-tool-for-secure-machine-learning-with-pytorch*, 2019. 3

[14] Florian Hartmann, Sunah Suh, Arkadiusz Komarzewski, Tim D Smith, and Ilana Segall. Federated learning for ranking browser history suggestions. *arXiv preprint arXiv:1911.11807*, 2019. 5

[15] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020. 2, 3

[16] A Ingerman and K Ostrowski. Introducing tensorflow federated. *External Links: Link Cited by*, 4, 2019. 2, 3

[17] Pengfei Jiang and Lei Ying. An optimal stopping approach for iterative training in federated learning. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2020. 6

[18] 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. *arXiv preprint arXiv:1912.04977*, 2019. 6

[19] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2019. 7

[20] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6341–6345. IEEE, 2019. 5

[21] 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, 2020. 1

[22] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020. 1

[23] Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115, 2019. 3

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1

[25] Florian Nuding and Rudolf Mayer. Poisoning attacks in federated learning: An evaluation on traffic sign classification. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 168–170, 2020. 5

[26] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):604–624, 2020. 1

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 2

[28] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018. 1

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1

[30] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, and S. Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1), 2020. 5

[31] Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. Functional federated learning in erlang (ffl-erl). In *International Workshop on Functional and Constraint Logic Programming*, pages 162–178. Springer, 2018. 3

[32] Xiaofei Wang, Chenyang Wang, Xiuhua Li, Victor CM Leung, and Tarik Taleb. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*, 7(10):9441–9455, 2020. 1

[33] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 2

[34] Zhaoping Xiong, Ziqiang Cheng, Xiaohong Liu, Dingyan Wang, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Facing small and biased data dilemma in drug discovery with federated learning. *BioRxiv*, 2020. 5

[35] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019. 3

[36] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018. 5

[37] Abbas Yazdinejad, Reza M Parizi, Ali Dehghantanha, and Kim-Kwang Raymond Choo. P4-to-blockchain: A secure blockchain-enabled packet parser for software defined networking. *Computers & Security*, 88:101629, 2020. 7

[38] Hanene Ben Yedder, Ben Cardoen, and Ghassan Hamarneh. Deep learning for biomedical image reconstruction: A survey. *Artificial Intelligence Review*, 54(1):215–251, 2021. 3

[39] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 393–399, 2020. 1

[40] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. Real-time end-to-end federated learning: An automotive case study. *arXiv preprint arXiv:2103.11879*, 2021. 5

[41] Hao Zhu, Man-Di Luo, Rui Wang, Ai-Hua Zheng, and Ran He. Deep audio-visual learning: A survey. *International Journal of Automation and Computing*, pages 1–26, 2021. 1

[42] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. Pysyft: A library for easy federated learning. In *Federated Learning Systems*, pages 111–139. Springer, 2021. 2, 3

[43] Alexander Ziller, Dmitrii Usynin, Rickmer Braren, Marcus Makowski, Daniel Rueckert, and Georgios Kaissis. Medical imaging deep learning with differential privacy. *Scientific Reports*, 11(1):1–8, 2021. 5