DISCO: A Browser-Based Privacy-Preserving Framework for Distributed Collaborative Learning

Julien Tuấn Tú Vignoud [1] **Valérian Rousset** [1] **Hugo El Guedj** [1] **Ignacio Aleman** [1] **Walid Bennaceur** [1] **Batuhan Faik Derinbay** [1] **Eduard Ďurech** [1] **Damien Gengler** [1] **Lucas Giordano** [1] **Felix Grimberg** [1] **Franziska Lippoldt** [1] **Christina Kopidaki** [1] **Jiafan Liu** [1] **Lauris Lopata** [1] **Nathan Maire** [1] **Paul Mansat** [1] **Martin Milenkoski** [1] **Emmanuel Omont** [1] **Güneş Özgün** [1] **Mina Petrović** [1] **Francesco Posa** [1] **Morgan Ridel** [1] **Giorgio Savini** [1] **Marcel Torne** [1] **Lucas Trognon** [1] **Alyssa Unell** [1] **Olena Zavertiaieva** [1] **Sai Praneeth Karimireddy** [1] **Tahseen Rabbani** [2] **Mary-Anne Hartley** [1] **Martin Jaggi** [1]

Abstract

Data is often impractical to share for a range of well considered reasons, such as concerns over privacy, intellectual property, and legal constraints. This not only fragments the statistical power of predictive models, but creates an accessibility bias, where accuracy becomes inequitably distributed to those who have the resources to overcome these concerns. We present **DISCO**[1]: an open-source **Dis**tributed **Co**llaborative learning platform accessible to non-technical users, offering a means to collaboratively build machine learning models without sharing any original data or requiring any programming knowledge. DISCO's web application trains models locally directly in the browser, making our tool cross-platform out-of-the-box, including smartphones. The modular design of DISCO offers choices between federated and decentralized paradigms, various levels of privacy guarantees and several approaches to weight aggregation strategies that allow for model personalization and bias resilience in the collaborative training.

[1]School of Computer and Communication Sciences, EPFL (Ecole polytechnique fédérale de Lausanne), Lausanne, Switzerland [2]University of Chicago, USA. Correspondence to: Julien Vignoud <julien.vignoud@epfl.ch>, Martin Jaggi <martin.jaggi@epfl.ch>.

Proceedings of the ICML 2025 Workshop on Championing Open-source Development in Machine Learning (CODEML '25).

[1]Code repository is available at https://github.com/epfml/disco/ and a showcase web interface at https://discolab.ai/

1. Introduction

By 2025, global data is estimated to reach 168 zettabytes, with a projected 25 petabytes of genomic data (Banks, 2020) and 50 petabytes of hospital data (Organization & University, 2019) added annually. As the scale of big data increases and its granularity deepens, so too does its potential power, value, risk and the legal constraints of sharing it (Horvitz & Mulligan, 2015). Beyond privacy concerns, sharing data also involves issues of ownership, sovereignty, intellectual property, and fairness (Ballantyne, 2020).

However, when data is not shared, the problems encountered are equally challenging. Its statistical power becomes fragmented, risking poor generalization (Zhao et al., 2018). This disproportionately affects lower-resource sites that lack the capacity to ensure robust data collection or collaborative data security (Zech et al., 2018). Protectively siloing data also creates interoperability drift and a burdensome harmonization task, further disincentivizing collaboration (Crowson et al., 2022).

Indeed, secure centralization of the data requires significant resources, and laborious applications for ethical approval, which are not granted in perpetuity, and thus do not align with the nature of the exponentially amassing live streams of big data.

To fully harness the potential of massively distributed data, the learning process can be similarly distributed to the data sources. Collaborative learning thus involves exchanging updates from local learning steps, typically in the form of gradients, which are then aggregated into a common model. In federated learning (FL), the gradient aggregation process is coordinated by a central server, whereas in decentralized learning (DL), it is delegated to individual nodes. While the concept of parallel computing is not new, the term 'federated learning' was formalized in 2016 (Konečnỳ et al., 2016; McMahan et al., 2017). It is understood as a set of challenges faced when deriving machine learning models

Table 1: Feature comparison of existing collaborative training platforms.
[1]DP = Differential Privacy. [2]Supports Decentralized Learning.

	DP[2]	Encryption	Open Src	Browser client	Decentralized[3]	Code-free training
SensiX (Min et al., 2020)						
HomoPAI (Li et al., 2020a)		✓				
IntegrateAI (Crowson et al., 2022)	✓	✓				
Nvidia FLARE (Roth et al., 2022)	✓	✓	✓			
Flower (Beutel et al., 2020)	✓	✓	✓			
FATE (Liu et al., 2021)		✓	✓			
PySyft (Ziller et al., 2021)	✓	✓	✓			
TF Federated (McMahan et al., 2017)	✓	✓	✓			
DISCO (this work)	✓	✓	✓	✓ *(incl. mobile)*	✓	✓

from multiple nodes (Kairouz et al., 2021).

Although collaborative learning appears to address most challenges of data sharing, several potential issues and vulnerabilities can hinder its practical implementation and scalability. For instance, while FL addresses the issue of data sovereignty and ownership, it places the intellectual property of the model and gradients at the discretion of a central entity. This central server also potentially represents a single point of failure, which can become a bottleneck in high-communication settings (Lian et al., 2017). While DL can resolve this issue, it comes with its own constraints such as limited computational power in end-nodes (e.g., smartphones).

Neither FL nor DL are immune to attacks on privacy (Mothukuri et al., 2021; Pasquini et al., 2022; Geiping et al., 2020; Wen et al., 2022) and require a battery of features to stave off threats to GDPR compliance (Truong et al., 2021). Various kinds of privacy-enhancing techniques can be combined with collaborative training, including differential privacy (Abadi et al., 2016; Wei et al., 2020), secure multiparty computation (Bonawitz et al., 2017; Mohassel & Zhang, 2017; Zhou et al., 2024), and homomorphic encryption (Zhang et al., 2020), etc.

For collaborative learning to be practically feasible and realize its potential to improve fairness, representation, and accessibility, an effort must be made to create a customizable suite of easy-to-use, open-source resources. In this work, we propose DISCO: an open-access, open-source **Dis**tributed **Co**llaborative learning platform, offering users a means to collaboratively build machine learning models without sharing any original data. The modular design of DISCO allows users to leverage a flexible combination of features according to their requirements. This includes choices regarding federated and decentralized communication protocols, various levels of privacy guarantees, and several approaches to gradient aggregation strategies.

While other FL platform implementations exist, none support DL or offer the same level of accessibility to non-technical users that DISCO provides through its code-free, installation-free client. We show a comparison of existing frameworks in Table 1 and describe them individually in Appendix A. The main contributions of DISCO are:

- An in-browser and code-free web application supporting local deep learning model training, including on smartphones. Its intuitive user interface eliminates the need for data owners to possess technical expertise required to install software and implement collaborative learning scripts, a common requirement for other platforms.
- Support for decentralized learning, enabling participants to train models collaboratively without sharing weight updates with a central server. Furthermore, DISCO can leverage secure multi-party computation to enhance model privacy against other participants.

2. The DISCO library

DISCO is a novel open-source software for collaborative machine learning. It is the first in-browser collaborative training platform supporting both federated and decentralized learning, and provides a code-free interface allowing users to create and join collaborative machine learning tasks, termed "**DISCOllaboratives**" and are described in Section 2.2. DISCO enables participants to jointly train deep learning models without sacrificing data ownership by sharing model weight updates — rather than raw data — through collaborative learning. It also provides users with a number of options to accommodate their specific training and privacy requirements.

DISCO is designed for two primary use cases. The first is *public*, large-scale collaborative training, where DISCOllaboratives are open to anyone to join by contributing readily accessible data, and new DISCOllaboratives can be created by any user. For instance, users can leverage DISCO to train spam detection systems by contributing their personal spam emails, speech recognition systems by contributing

voice recordings based on provided transcripts or for image classification by labeling medical or non-medical images. A public DISCO instance[4] already showcases this use case.

The second use case envisioned is a *private* intra-organizational training, for instance, within a network of hospitals—where the generalization of medical models would significantly benefit from inter-hospital collaboration, but hindered due to patient data confidentiality and data sharing laws. In such scenarios, training is restricted to selected entities aiming to train models collaboratively without direct data sharing, typically deploying a custom private DISCO instance with whitelisted access.

Consequently, DISCO is not a single website; numerous instances can be deployed, both public and private, that accommodate for its diverse use cases.

2.1. Architecture

DISCO features a modular architecture comprising several key components: a core library complemented by 2 platform-specific packages, a server for communication orchestration, and two distinct user interfaces - a web application and a command line interface. This design promotes reusability and composability. To enable in-browser operation and ensure that DISCO remains easy to customize and adapt, all modules - including the machine learning stack and backend components - are implemented in JavaScript/TypeScript. Each of these modules is also open-source, allowing users to further customize DISCO according to their specific needs.

Core library. The `discojs` core library is platform agnostic, designed to run in any JavaScript/TypeScript engine. It forms the foundational codebase of DISCO, containing type definitions utilized throughout the system. Key components include a `Dataset` class for efficient data loading and composition, a `Trainer` to manage local model training synchronized with server updates, various `Aggregator` implementations offering different levels of security, and a collection of `Model`. The library's generic design permits the use of any framework for model definition; currently, models are implemented using the TensorFlow.js library (Smilkov et al., 2019).

Two platform-specific packages complement the library for browser and Node.js environments (respectively `discojs-web` and `discojs-node`), providing helper functions that facilitate its use and leverage platform-specific technologies, such as a compiled TensorFlow library on Node.js and `Canvas` rendering in the browser.

Clients. The web application is DISCO's code-free user interface, through which users join DISCOllaboratives. All computation occurs directly within the browser, with no off-loading to local programs. This design ensures broad accessibility: no installation is required, and the platform can even run on mobile devices. The web application primarily targets non-technical users; no domain-specific knowledge is required and users can join training sessions by connecting their data and initiating the process. The interface also supports the creation of new DISCOllaboratives and using trained models for inference.

As an alternative to the web interface, DISCO provides a Command Line Interface (CLI) relying on Node.js, enabling technical users to interact with DISCO headlessly and facilitating large-scale experiments. Both CLI and browser-based users can participate in the same training session.

Server. The server, deployed by technical maintainers, hosts available DISCOllaboratives, orchestrates training sessions, and handles federated weight aggregation. In DL, it is also utilized for peer discovery and the pacing of training rounds.

2.2. DISCOllaboratives: collaborative training tasks

Creation. DISCOllaboratives can be set up via the code-free UI or programmatically. Each task includes a description of the training goal, the base model architecture (currently in TensorFlow.js format), the expected format of data, and customizable hyperparameter settings. These hyperparameters include epochs, batch size, differential privacy settings among others. DISCO's public instance showcases several pre-defined tasks. These DISCOllaboratives include classical LLM training and MNIST classification or COVID-19 diagnosis from lung ultrasounds.

Training. Participating users are first shown a description of the DISCOllaborative and prompted to connect the data they wish to use for training. Importantly, **connected data is not uploaded anywhere and remains on the local device**, the UI reads this data solely for local model training. After connecting their data, users can directly join the collaborative training session. Note that users can also decide to train locally on their own. The training is paused if there are insufficient participants in the session.

Any user participating in a task can monitor their local model's performance and view information regarding the number of actively participating users in the session. Figure 1 depicts the training and test curves for a model that is globally aggregated every two epochs (over a total of 20 epochs). Note that these accuracies reflect performance on the individual client's data; consequently, these graphs will differ from client to client. After training, users can export the resulting model or utilize it within the DISCO platform.

[4] https://discolab.ai/

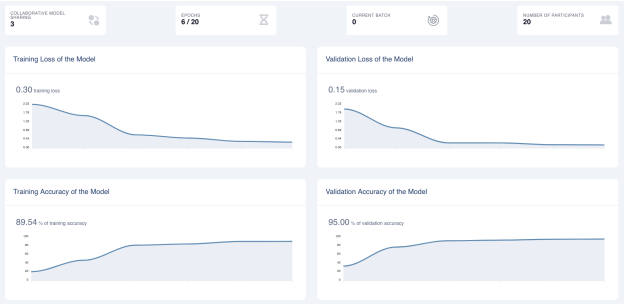

Figure 1: Performance visualization for a federated MNIST classification task.

2.3. In-browser Deep Learning

Performing deep learning operations directly within the user's web browser is foundational to DISCO's accessibility. This is primarily achieved through TensorFlow.js (Smilkov et al., 2019), an open-source machine learning library maintained by Google specifically developed for JavaScript environments. TensorFlow.js supports multiple computational backends that implement tensor storage and mathematical operations. The WebGL backend is the most powerful one and leverages the system's GPU when available for hardware acceleration. With WebGL, tensors are stored as textures, and mathematical operations are implemented as WebGL shaders. TensorFlow.js can fall back on the CPU or WebAssembly backends. When using DISCO's CLI rather than the web application, TensorFlow.js can leverage its Node.js backend for hardware acceleration.

TensorFlow.js natively supports tabular data and images. DISCO extends its capabilities to language modeling tasks by implementing the Transformers block in Tensorflow.js, resulting in a custom GPT2 implementation. Tokenization is performed using the Transformers.js library (Lochner, 2023).

Transformers.js - and ONNX Runtime Web (developers, 2021) on which it relies - are promising frameworks for browser-based ML, notably due to early WebGPU support (the successor to WebGL) and the possibility to convert existing pre-trained language models to Transformer.js. However, in-browser training is not yet supported by either libraries and to this date TensorFlow.js remains the most popular library implementing in-browser automatic differentiation for back-propagation.

2.4. Privacy-preserving training

To implement differential privacy, task creation options enables adding Gaussian noise (McMahan et al., 2018) to gradient updates by setting a noise scale, as well as clipping gradients (Abadi et al., 2016) by adjusting a clipping radius. These measures aim to limit malicious or honest-but-curious reconstruction of cross-client training data statistics.

DL also supports secure multi-party computation (SMPC) (Zhou et al., 2024) for aggregating weight updates, thereby keeping them private from other participants. SMPC relies on additive secret sharing, wherein a secret - here a local weight update - is divided into shares. Each share is generated by adding a random value to a specific portion of the original secret. The shares are distributed among the parties, and the secret can only be reconstructed by combining a sufficient number of shares. Furthermore, support for Byzantine-robust aggregation methods (Karimireddy et al., 2021) is currently under development.

2.5. Collaborative Learning in DISCO

Federated learning. DISCO relies on a central server to manage the aggregation of model updates and the distribution of the updated model architecture in each round. After training the model locally in the web application, each client uploads its model update (as a gradient) to the server. The server aggregates all updates using the classical FEDAVG algorithm (McMahan et al., 2017) and sends a global update back to all clients. Implementing robust aggregation methods, such as FEDPROX (Li et al., 2020b) is currently underway. DISCO clients establish WebSocket connections to communicate with the server, allowing the latter to be notified when clients leave the training session.

Decentralized learning. In contrast, nodes in DL do not share weights with a central server; instead, they exchange them directly, a process that occurs within the clients' web browsers. Specifically, DISCO nodes establish peer-to-peer connections using WebRTC (Web Real-Time Communication) which is the only way to ensure direct connections between peers within the browser environment. WebRTC's capabilities, such as NAT traversal (including hole punching for firewalls) and support for re-establishing dropped connections, allow DISCO to operate across diverse network configurations.

Although the server never accesses weight updates in DL, it remains essential for orchestrating the training session. This orchestration includes assigning client IDs, designating peers for participation in weight-sharing rounds, and acting as a WebRTC *signaling server* to facilitate peer discovery and connection establishment.

At each weight sharing round, nodes first notify the server of their intent to join. Each node then trains locally for a predetermined number of epochs before informing the server of its readiness to exchange weights. Once the number of peers ready to exchange weights reaches a threshold defined during task creation, the server shares the list of these peers with the nodes participating in the round. Peers then establish peer-to-peer connections to exchange weight updates, forming a fully connected topology. Weight updates can then be aggregated either by simple averaging or

by leveraging secure multi-party computation as described in Section 2.4.

The availability of decentralized training is a unique feature of DISCO, not currently supported by comparable collaborative learning platforms (see Table 1).

3. Conclusion

We introduced DISCO, a novel open-source platform for collaborative machine learning that preserves data sovereignty. By supporting both federated and decentralized learning, and integrating flexible privacy safeguards, DISCO enables model training without the need to centralize raw data. Its no-install, no-code interface—including support for mobile browsers—lowers technical barriers to entry, fostering broader participation in AI development. In doing so, DISCO contributes to more inclusive, equitable, and privacy-preserving machine learning ecosystems.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318. ACM, 2016.

Ballantyne, A. How should we think about clinical data ownership? *Journal of medical ethics*, 46(5):289–294, 2020.

Banks, M. A. Sizing up big data. *Nature Medicine*, 26(1):5–6, 2020. ISSN 1546-170X. doi: 10.1038/s41591-019-0703-0. URL https://doi.org/10.1038/s41591-019-0703-0.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Kwing, H. L., Parcollet, T., Gusmão, P. P. d., and Lane, N. D. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191. ACM, 2017.

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

Crowson, M. G., Moukheiber, D., Arévalo, A. R., Lam, B. D., Mantena, S., Rana, A., Goss, D., Bates, D. W., and Celi, L. A. A systematic review of federated learning applications for biomedical data. *PLOS Digital Health*, 1:1–14, 05 2022. doi: 10.1371/journal.pdig.0000033. URL https://doi.org/10.1371/journal.pdig.0000033.

developers, O. R. Onnx runtime. https://onnxruntime.ai/, 2021.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

Horvitz, E. and Mulligan, D. Data, privacy, and the greater good. *Science*, 349(6245):253–255, 2015. doi: 10.1126/science.aac4520. URL https://www.science.org/doi/abs/10.1126/science.aac4520.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning, 2021.

Karimireddy, S. P., He, L., and Jaggi, M. Learning from history for byzantine robust optimization, 2021. URL https://arxiv.org/abs/2012.10333.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Li, Q., Huang, Z., Lu, W.-j., Hong, C., Qu, H., He, H., and Zhang, W. Homopai: A secure collaborative machine learning platform based on homomorphic encryption. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1713–1717. IEEE, 2020a.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020b.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent, 2017.

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

Lochner, J. Transformers.js: State-of-the-art machine learning for the web. https://github.com/huggingface/transformers.js, 2023.

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

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models, 2018. URL https://arxiv.org/abs/1710.06963.

Min, C., Mathur, A., Montanari, A., Acer, U. G., and Kawsar, F. Sensix: A platform for collaborative machine learning on the edge, 2020.

Min, C., Mathur, A., Acer, U. G., Montanari, A., and Kawsar, F. Sensix++: Bringing mlops and multi-tenant model serving to sensory edge devices. *ACM Transactions on Embedded Computing Systems*, 22(6):1–27, 2023.

Mohassel, P. and Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pp. 19–38. IEEE, 2017.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2020.10.007. URL https://www.sciencedirect.com/science/article/pii/S0167739X20329848.

Organization, W. and University, C. *Global Innovation Index 2019: Creating Healthy Lives — The Future of Medical Innovation*. WIPO Publication. World Intellectual Property Organization, 2019. ISBN 9791095870142. URL https://books.google.com/books?id=6uilDwAAQBAJ.

Pasquini, D., Raynal, M., and Troncoso, C. On the privacy of decentralized machine learning, 2022. URL https://arxiv.org/abs/2205.08443.

Roth, H. R., Cheng, Y., Wen, Y., Yang, I., Xu, Z., Hsieh, Y.-T., Kersten, K., Harouni, A., Zhao, C., Lu, K., et al. Nvidia flare: Federated learning from simulation to real-world. *arXiv preprint arXiv:2210.13291*, 2022.

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S. N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F. B., and Wattenberg, M. Tensorflow.js: Machine learning for the web and beyond, 2019. URL https://arxiv.org/abs/1901.05350.

Truong, N., Sun, K., Wang, S., Guitton, F., and Guo, Y. K. Privacy preservation in federated learning: An insightful survey from the GDPR perspective. *Computers & Security*, 110:102402, nov 2021. ISSN 0167-4048. doi: 10.1016/J.COSE.2021.102402.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

Wen, Y., Geiping, J., Fowl, L., Goldblum, M., and Goldstein, T. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv preprint arXiv:2202.00580*, 2022.

Yang, L., Tao, F., Tianjian, C., Qian, X., and Qiang, Y. An industrial grade federated learning framework. *The Journal of Machine Learning Research*, pp. 10320–10325, 2021.

Zech, J. R., Badgeley, M. A., Liu, M., Costa, A. B., Titano, J. J., and Oermann, E. K. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. *PLOS Medicine*, 15:1–17, 11 2018. doi: 10.1371/journal.pmed.1002683. URL https://doi.org/10.1371/journal.pmed.1002683.

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pp. 493–506, 2020.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zhou, I., Tofigh, F., Piccardi, M., Abolhasan, M., Franklin, D., and Lipman, J. Secure multi-party computation for machine learning: A survey. *IEEE Access*, 12:53881–53899, 2024. doi: 10.1109/ACCESS.2024.3388992.

Ziller, A., Trask, A., Lopardo, A., Szymkow, B., Wagner, B., Bluemke, E., Nounahon, J.-M., Passerat-Palmbach, J., Prakash, K., Rose, N., et al. Pysyft: A library for easy federated learning. *Federated Learning Systems: Towards Next-Generation AI*, pp. 111–139, 2021.

A. Related work

SensiX (Min et al., 2020) and its successor SensiX++ (Min et al., 2023) are multi-device, multi-modal runtimes tailored for edge-based collaborative learning. Clients served by this system component include sensory devices such as earbuds, smartwatches and smartphones. SensiX lies between the sensors and sensory model(s), adaptively pooling data from subsets of clients depending on data quality, energy costs, device availability, and desired task. SensiX is demonstrated for a variety of sensory tasks such as human activity recognition and audio keyword spotting, but only in a setup consisting of trusted wireless devices. In particular, privacy is not considered: the centralized SensiX controller is given unrestricted access to sensor data. Furthermore, as opposed to federated learning, the sensory model outputs do not improve the on-device models of the sensors. The SensiX library is not accessible to the public.

HomoPAI (Li et al., 2020a) is a secure collaborative protocol built on top of Alibaba Cloud's Platform for AI (PAI). The system enables multiple parties to train a central via model through submission of encrypted local data. Homomorphic encryption (HE) is performed on both user data and the model to enable training securely – only the final model is decrypted and sent out to users. Peer-to-peer (i.e., decentralized) learning is also available. HomoPAI also makes use of MPI to conduct training in parallel. Similar to FL, the protocol attempts to prevent exposure of raw client data. However, in HomoPAI, data is submitted to the central server, as opposed FL, where only gradients are uploaded. A demo is provided for logistic regression, though deep neural networks are untested (with only pseudo-homomorphic approximations of activation functions such as the ReLU. It is stated that accuracies are suboptimal when compared to training over raw data, whereas modern FL matches single-machine training over a wide cluster of tasks. Furthermore, conducting operations over HE data, especially at the scale of a deep network is computationally expensive. The library is not open-source and it is unclear if it still being maintained.

integrateAI (`https://www.integrate.ai/`) is an online federated learning platform. The motivation is to enable pools of trusted collaborators to jointly train a model according to the principles of centralized FL. The motivation and design of integrateAI is quite similar to DISCO, with features such as DP gradients, cloud-based or local training, and custom dataset loading. However, unlike DISCO, the integrateAI platform is closed-source (API-based), doesn't work on mobiles or browser, and does not support decentralized learning.

NVIDIA FLARE (NVIDIA Federated Learning Application Runtime Environment) (Roth et al., 2022) is a domain-agnostic, open-source federated SDK. FLARE supports various aggregation functions (FedAvg, FedProx), custom datasets, arbitrary architectures, and privacy-preserving model updates via various DP mechanisms and homomorphic encryption (HE). However, FLARE does not support decentralized learning and as an SDK, it is intended for application development and requires engineered setup, as opposed to than drop-in browser-based deployment like DISCO.

Flower (Beutel et al., 2020) Flower is a federated learning programming library that provides a unified approach to federated learning by remaining agnostic machine learning frameworks, meaning it is compatible with popular frameworks such as PyTorch, TensorFlow, Keras, scikit-learn, and Hugging Face. It is also platform-independent, allowing interoperability across different operating systems and hardware, which is beneficial for heterogeneous edge device environments. Flower emphasizes usability, enabling users to create a complete federated learning system with a small amount of Python code.

FATE (Federated AI Technology Enabler) (Yang et al., 2021) is focused on the industry-scale federated learning. FATE supports horizontal and vertical FL, linear/deep/tree-based models, and transfer learning. For gradient-boosted tree models, FATE supports the SecureBoost (Cheng et al., 2021) algorithm for enhanced privacy, along with model-agnostic approaches such as secure multi-party computation (MPC), HE, and general DP mechanisms. FATE-Flow is a multi-party federated task security scheduling platform which includes resource coordination, real-time job monitoring, multi-party cooperation authority management, CLI and Restful APIs. FATE does not support decentralized learning nor have a browser platform.

PySyft (Ziller et al., 2021) is a large open-source library for federated learning. Data may be held locally or distributed by a central server across workers (for an internal and/or trusted network). PySyft enables both encrypted training and differentially private/secure gradient exchange. The platform relies on a "datasite" which is a centralized server clients may log into. Datasites may be launched using Docker or Kubernetes. A defining feature of PySyft is its interoperability: it supports PyTorch and TensorFlow training and mobile-friendly programming languages such as Kotlin and Swift. Decentralized learning is not supported as only a single node can send back results, and while it has a JavaScript version,

`syft.js`, it hasn't been updated in years so currently it does not support a browser application.

TensorFlow Federated (TFF) (`https://www.tensorflow.org/federated`) is an open-source framework within the TensorFlow ecosystem for centralized federated learning. TFF is organized into a high-level layer for general implementation of federated training to existing models and a low-level layer for the construction of communication operators, aggregation functions, and foundational algorithms. Support for differential privacy is provided, and parts to build secure aggregation. Furthermore, update and model compression is supported for low-resource systems. Decentralized learning is unavailable, and the framework is intended for integrated engineering for existing TensorFlow routines as opposed to drop-train training for edge clients. It is solely in Python thus can't be run in a browser.