Towards Volunteer Deep Learning: Security Challenges and Solutions

Divi De Lacour, Marc Lacoste, Mario Südholt, Jacques Traoré

Towards Volunteer Deep Learning: Security Challenges and Solutions

Divi De Lacour[1,2], Marc Lacoste[1], Mario Südholt[2], Jacques Traoré[1]
Orange Innovation[1], IMT Atlantique/Inria[2]
{divi1.delacour,marc.lacoste,jacques.traore}@orange.com,mario.sudholt@imt-atlantique.fr

Abstract

Recently, the costs of training deep learning models have dramatically increased. Volunteer Computing (VC) enables cost-effective gathering of large amounts of idle computing resources, but has been little applied to deep learning. Volunteer Deep Learning (VDL) applies VC to train deep learning models while keeping costs low. In this paper we review current VDL concepts, projects, challenges and opportunities. We analyze the VDL security challenges and countermeasures. We propose guidelines towards a VDL framework implementation based on decentralized federated learning.

Keywords: Volunteer Deep Learning, Volunteer Computing, Federated Learning, Decentralized Learning, Security, Byzantine Resilience

1 Introduction

Over the last decades, different combinations of paradigms such as cloud computing, collaborative architectures, artificial intelligence and distributed computing have been intensively investigated to make systems smarter and cost efficient. Applications span multiple research fields [5, 14, 35], including medicine [8, 16], education [36], astronomy [10], agriculture, energy efficiency [3] or smart mobility.

Deep Learning. More recently, the spectacular development of Large Language Models (LLM) has confirmed deep learning (DL) as a major step towards smarter systems. DL models have shown to yield higher accuracy with larger numbers of parameters and training data [27]. Though, large DL model training remains expensive and requires a great amount of data and computing power.

Volunteer Computing. To reduce costs, academia developed the *Volunteer Computing* (VC) paradigm [34]. Explored in projects such as Folding@home [8] or BOINC [1], VC aims to use idle resources of computers (e.g., CPU scavenging) for scientific tasks (e.g., protein folding simulations, prime number research) that require massive computing power or to operate over large amounts of data. Crowdsourcing [20] is a related approach that uses volunteers to produce data, e.g., to label data sets. The Amazon Mechanical Turk (MTurk)[1] is a commercial example of crowdsourcing against a remuneration. Unfortunately, VC approaches have seldom been used to train DL models.

Volunteer Deep Learning: combining VC and DL. To reconcile both worlds, we propose the term *Volunteer Deep Learning (VDL)* to refer to situations when VC is applied to train DL models [18, 19, 37].
Those approaches leverage Federated Learning (FL) [12], used to train DL models on confidential data directly on the devices holding the data.
Compared with centralized DL training, VDL presents several critical challenges: 1) **Synchronisation** requires efficient communications and to take into account device heterogeneity, 2) **Security** countermeasures are needed to guarantee training integrity and privacy, 3) **Deployment** guidelines should be defined to support a high diversity of hardware and software. While current works has focused on improving performance, unfortunately, security and deployment remain little studied.
Therefore, we felt it necessary to provide a bird's eye view of the VDL paradigm, with a focus on security challenges and solutions, and to provide first elements of a VDL framework, exploring the potential of decentralized FL architectures.
We provide in this paper the following contributions. We review the current VDL concepts, projects, challenges and opportunities. We survey the security challenges of VDL and its countermeasures. We propose guidelines towards a VDL framework implementation.
The paper is structured as follows. Section 2 provides background on VC and VDL. Section 3 presents the VDL security challenges and countermeasures. Section 4 gives an overview of the VDL ecosystem and opportunities. Section 5 proposes guidelines towards implementing a secure VDL framework.

2 Background

2.1 Volunteer Computing

Overview. VC [34] approaches are based on a form of volunteering where each participant shares unused resources of his computer (see Figure 1). VC approaches have been the subject of an important literature. The most popular VC projects are based on a gamification of participation (e.g., high-score pages). VC approaches where participants are being paid remain scarce.

[1] https://www.mturk.com/

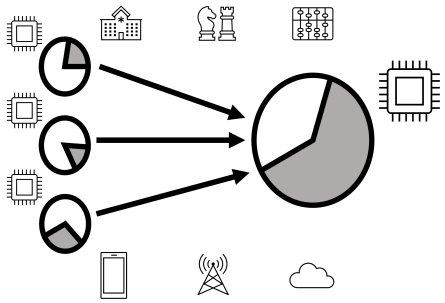
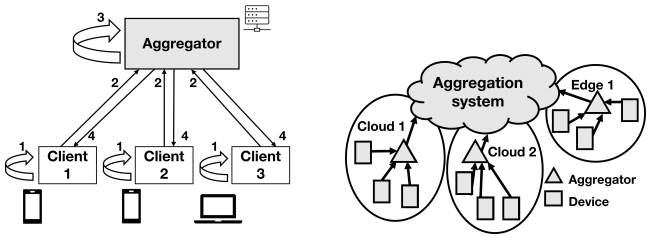


Figure 1. Volunteer Computing



(a) Federated Learning **(b)** Volunteer Deep Learning

Figure 2. Federated Learning vs. Volunteer Deep Learning for model training across the Cloud-Edge

VC projects are usually based on client-server architectures as in FL, with tasks easy to parallelize (e.g., searching for prime numbers). Communications can then take place on high-latency and low-bandwidth connections.
One notable difference between VC and grid computing is that VC participants are highly untrusted and unreliable. Therefore, tasks often rely on replication mechanisms.
VC projects face three types of challenges:

- **Participation**: gather a maximum number of participants and of computing resources.
- **Performance**: exploit efficiently available resources.
- **Security**: guarantee confidentiality, integrity, and availability.

Participation challenges include:

- **Encouraging participation**: attract participants that will volunteer resources to the VC project. This challenge is critical as VC project resources directly depend on participation.
- **Heterogeneity**: make a project available on a diversity of hardware and software and take into account their capacity. Clients may each have different devices.

The reasons for participation have been explored in-depth, highlighting the different categories of participants, their behaviors and incentives [34].

Performance challenges include:

- **Resilience**: the system should support random client failures. Clients in VC projects are subject to high disconnection rates (churn).
- **Scaling**: the system should support a high number of participants. VC projects aim to gather massive computing power through an important number of low-capacity participants.
- **Task scheduling**: dispatching tasks to volunteers efficiently in terms of requirements and performance (latency, bandwidth).

Security challenges include:

- **Confidentiality** of data, participants, and in some cases, of task results.
- **Byzantine threats**: protect tasks against a fraction of malicious participants providing spurrious replies, threatening computing integrity.
- **Sybil attacks** [4]: i.e., a single entity creates multiple identities to increase its voting rights. Sybil attacks can be a first step for Byzantine threats.
- **Availability** of resources (e.g., memory, computing units) and of the distributed system (e.g., central server, P2P network).

So far, most works focused on guaranteeing integrity of results. Folding@home adopts a closed-source approach to limit retro-engineering. It also adopts replication: the same task is sent to multiple clients to detect malicious replies. Cheating has been detected in SETI@home, i.e., participants claim credit for data that has not been processed. Mitigation is similar to the Folding@home approach.

2.2 VDL: Volunteer Computing for Deep Learning

Overview. DL is used to solve a variety of tasks, including in computer vision or natural language processing, e.g., object recognition, text generation [27, 38]. In FL [12, 25], a server sends a model to be trained to clients on their local data. Clients send back the trained models to the server for aggregation (see Figure 2a). FL can be considered as a form of VC to train DL models - more specifically when the training data set is not shared between clients.
The aim of VDL is to train large DL models using all available resources in the IoT-Cloud continuum. Examples include unused devices and idle resources of personal computers or cheapest clouds (see Figure 2b).
VDL projects include [11, 18, 19, 22, 30, 37]. [30] is one of the most recent works in this area. [18] includes resilience mechanisms and AWS spot instances to reduce computing costs (cloud optimization). Its client-server architecture is implemented with the VC framework BOINC.
For training, two VDL projects are up to our knowledge implemented and ready-to-deploy. Hivemind [7, 39] is the most famous project. Disco [33] proposes to participate in the training of DL models directly in the web browser, in a federated or decentralised manner - without need to install

a software. For inference, Petals [6] runs in a cooperative manner DL models too large to fit on a single computer.
The key challenge for VDL is to train large models with minimal performance overhead and securely on multiple clients. Training should also be privacy-preserving, scalable and resilient. We discuss VDL performance challenges and security challenges below and in Section 3 respectively.

Performance challenges. Task scheduling is hard in VDL as tasks are concurrent, but not independent. Local models should be synchronized regularly (e.g., at each training step). This requirement can make training quite inefficient on a high-latency and low-bandwidth network. Handling straggler nodes is also challenging for resilience and efficiency.

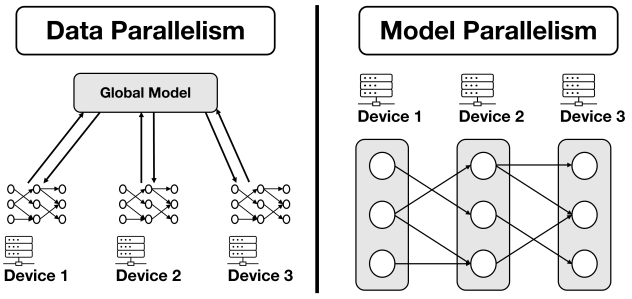


Figure 3. Model and Data Parallelism

In terms of design solutions to such communication and concurrency issues in a decentralized setting, distributed learning can be based on two types of parallelism: *model* and *data parallelism* (see Figure 3).
Most approaches are based on *data parallelism*: a batch of training data is distributed on multiple machines. This allows distributing the computing load on multiple computing units in parallel. Each client processes its own batch of data separately and then aggregates its model with other clients. For large models, the whole model cannot fit in the memory of a single machine. *Model parallelism* should then be adopted, where a client holds a fraction of the whole model in its memory (usually divided by layers).
Model parallelism is usually implemented using techniques such as Gpipe [29]. SWARM [23] proposes a similar approach to distribute layers on different devices, taking into account performance and distance (network latency, bandwidth).
Other techniques such as gradient accumulation allow to reduce the memory footprint. Memory transfer from GPU VRAM to RAM are possible, with slower training speed. For large models, other compression techniques like quantization and pruning can be added by reducing weight size and eliminating useless neurons. Impact on model accuracy is low, but memory footprint may be reduced by an order of magnitude [40].

3 VDL Security

We now review VDL security challenges and counter-measures.

3.1 Security Challenges

Many threats on DL [24] and FL [9] may also apply to VDL. We focus on the following key VDL security challenges: *integrity*, as training should produce a correct DL model; and *privacy*, as training data should be kept private.

Integrity. VDL does not require an exact model. It can even be hard to get the same model every time due to floating-point operations roundings that can vary between different systems. Models can be victims of attacks such as *backdoor attacks* [13]: the model performance is degraded on a specific class of samples. Counter-measures may only filter part of malicious updates sent by clients. Different levels of integrity may be achieved depending of the update impact on the overall model. The challenge is then to evaluate model integrity – a first broad metric being model accuracy.

Privacy. Keeping training data private is a main driver for FL [12, 25]. But assessing the privacy guarantees of FL schemes remains difficult. While some VC projects training data can be public, a common applicable approach for VDL privacy is based on *differential privacy (DP)* [17]. DP provides a probabilitistic indistinguishability guarantee for processing a specific data sample during training. This means that a specific training sample cannot be extracted from the trained model – the training data set probability distribution can still be leaked. Privacy requirements may also include confidentiality of the neural network architecture and weights.

Sybil threats. In Sybil attacks, an attacker creates multiple identities to increase its voting power [4]. For VC systems, such attacks are generally ignored: countermeasures such as IP address bans are considered sufficient.

VC security vs. cloud security. Security challenges differ between traditional VC and cloud optimization. In the cloud optimization case, preemptible instances are deployed across multiple clouds. Unlike traditional VC, training integrity and data privacy can be considered guaranteed, as the VDL platform administrators control the devices that take part in the training process. Also, cloud instances are considered honest and privacy-preserving.

3.2 Integrity Countermeasures

For distributed machine learning, training integrity is closely related to *Byzantine resilience* [31], as malicious models may be introduced in such a decentralized setting. Most solutions against Byzantine clients such as FLAME[28] are based on filtering the most important outliers. They convert the model weights into vectors and compare them, e.g., keeping the median of vectors instead of the average. FLAME also adds

noise (DP) to further reduce the effectiveness of backdoor attacks. Other approaches are based on redundancy by making multiple clients perform the same tasks or storing intermediate results. However, they breach training data privacy, as clients have to share their training data for safety checks.
To guarantee *aggregation integrity*, most solutions are secure aggregation schemes [21] that may get in conflict with Byzantine protections. For FL, this challenge is little explored as the central server is considered trusted, under the direct control of the organizers.

3.3 Privacy Countermeasures

Privacy countermeasures include: 1) *differential privacy* to prevent leak of private data from the model weights, and 2) *secure aggregation protocols* to prevent the aggregating server from accessing weights of individual models.
DP [17] improves FL and VDL privacy by adding random noise to the model weights. DP can be applied by clients to protect their training data from aggregators; or by aggregators to protect training data from other clients at the next training round. DP increases privacy but reduces the accuracy of the model.
Secure aggregation protocols [21] are cryptographic schemes where clients encrypt their model before sending it to the server for aggregation. The server is only able to get access to the weights of the aggregated model and not to the weights of the individual models sent by clients. Secure aggregation schemes require high processing power. An algorithm that is scalable, fault tolerant, while supporting Byzantine protection and DP-enabled privacy still remains to be found.

3.4 Privacy and Integrity

Trusted Execution Environments (TEE) are also part of applicable counter-measures to achieve both privacy and integrity [15, 26]. The TEE encrypted memory and integrity guarantees may help to analyze training data, train or aggregate models in a secure and privacy-preserving manner. TEE remote attestation may also be useful for devices to verify that tasks take place on secure nodes. How to compose TEE with other privacy-enhancing technologies such as distributed protocols, DP, and cryptographic scheme is still an open research challenge.

4 VDL Potential

We review some opportunities for VDL and analyze its current ecosystem.

4.1 A SWOT Analysis

We use the SWOT (Strengths, Weaknesses, Opportunities, Threats) methodology to provide a bird's eye view of the potential of VDL (see Table 1).

Strengths	Weaknesses
✦ Potential participants	✦ Environmental impact
✦ Networking	✦ Dependence on popularity
Opportunities	**Threats**
✦ Reduce training costs	✦ Intellectual Property
✦ Encourage data sharing	✦ Security and Privacy
✦ Collaborative training	✦ Unethical models

Table 1. A SWOT Analysis for VDL

Strengths. The number of *potential participants* is extremely important: many unused devices (e.g., smartphones, video game consoles, PCs) feature high computing power and GPUs usable for DL tasks. On the *network* side, high bandwidth and low latency connections are now easily available to most participants.

Weaknesses. The *environmental impact* should be assessed: VDL consumes a lot of bandwidth, with therefore a significant impact in terms of energy efficiency. A VDL project is also *dependent on popularity* in order to attract the participants necessary for success.

Opportunities. The main VDL opportunity is to *reduce and/or mutualize training costs of large models:* when cloud capacities are insufficient, costs can be reduced using preemptible instances and choosing best-of-breed cloud providers. Also, VDL *encourages data sharing*: as in FL, participants sharing their data enables to produce more optimal models than training models individually. Finally, VDL enables *collaborative model training:* typically between stakeholders that do not wish or cannot centralize computing, e.g., hospitals, that use organization-specific anonymization techniques but wish to access a common model.

Threats. In any VDL project, the question of *Intellectual Property (IP)* is critical: who owns the trained model and/or training data can be a legal challenge. *Security* and *privacy* is another key challenge as participants could degrade the performance of the models, introduce backdoors, or information on training data could be extracted from the model. VDL could also be used to train *unethical models*.

4.2 Ecosystem

VC seems to be mostly explored in academia, with well-established projects such as BOINC [1] from UC Berkeley. Such non-profit projects attract participation by garnering public interest on hot topics such as cure of rare diseases or search for extra-terrestrial life.
Up to our knowledge, VC seems so far little investigated by FAANG or BATX companies. Those players are major cloud stakeholders from which they make a significant part of their income. They also profit from data collection.

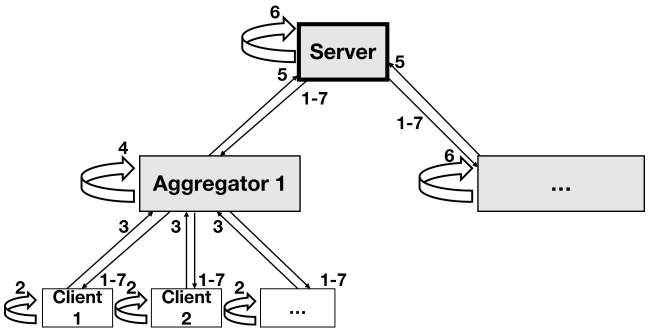


Figure 4. FDFL training process [32]

5 Towards a VDL Framework

VDL requires a framework to support new projects. Ideally, no networking or security background should be necessary.

5.1 Requirements

In a first approach, a VDL framework should run on a high diversity of devices and cloud platforms, while making the most of their performance and handling failures. It should also provide security and privacy guarantees. One direction for future work is to provide a detailed requirements analysis for building a VDL platform and framework [2].

5.2 Architecture

As FL can be implemented with composable security components and a common data set (or not) to train DL models in a collaborative manner, a FL architecture could be a starting point for a VDL architecture. As in centralized systems, the server is a single point of failure. It also limits the number of participants in terms of load. To tackle those challenges, a fully decentralized learning approach is preferred.
FDFL [32] proposes a fully decentralised FL extension to improve FL scalability and resilience while preserving convergence speed. First, intermediate aggregators are placed between the server and clients to reduce the load on the server and improve scalability (see Figure 4). Second, resilience is improved by introducing an election process on a peer-to-peer network to regularly designate new aggregators and a server. This allows to replace the aggregators or server in case of failure. Security countermeasures have to be added and are currently under implementation.
Figure 5 sketches a possible VDL framework architecture. The system is managed using a configuration file that allows participation of any entity without need for a central server to supervise training. The file can be a template that may be specialized (e.g., training hyper-parameters, security configuration). This approach improves project re-usability such as starting from a well-known security configuration. Security components are selected among a list of pre-defined alternatives (e.g., type of DP to be applied) or downloaded at start-up time. It allows to list alternative security components if the first one is unavailable. The database can be a private internal database (classic FL approach) or a public data set shared on cloud storage or a P2P network. The VDL framework should be self-managed in terms of configuration, optimization, resilience and protection [2].

5.3 Implementation

The framework is foreseen to be implemented in JavaScript for ease of deployment [33] – it can be run onto any device with a JavaScript run-time (e.g., nodeJS server, smartphone web browser, video game console). This allows to develop a single framework that can run with user-level rights.
Compute-intensive tasks can be run using WASM for an improved performance and safety. Interface with GPUs is also easy using a DL framework such as TensorflowJS (using the CUDA API on cloud servers, or WebGL on mobile devices). In a decentralized setting, the P2P network should help participants optimize aggregation by choosing peers close in terms of latency and bandwidth.

6 Conclusion

VDL shows potential to reduce DL training costs, using VC to gather idle resources from a high number of participants in edge devices or multiple clouds. Despite opportunities, protection remains a main barrier, with security challenges and counter-measures similar to FL. A decentralized, scalable and resilient FL framework such as FDFL could provide a first step towards a VDL framework.
Next steps are to design and implement a multi-platform VDL framework for the IoT-Cloud continuum. This framework should provide automated adaptation to hardware capacity and support failures. Composition of security components to guarantee training integrity should be explored, especially its impact on training performance and on the models. Further studies should also quantify the environmental and economical impact of VDL.

References

[1] D.P. Anderson. 2020. BOINC: A Platform for Volunteer Computing. *J. Grid Computing* 18 (2020), 99–122.
[2] D.P. Anderson, C. Christensen, and B. Allen. 2006. Designing a Runtime System for Volunteer Computing. In *ACM/IEEE Conference on Supercomputing (SC)*.
[3] R. Couillet, D. Trystram, and T. Ménissier. 2022. The Submerged Part of the AI-Ceberg [Perspectives]. *IEEE Signal Processing Magazine* 39, 5 (2022), 10–17.
[4] J. R. Douceur. 2002. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems*.
[5] The Economist. 2023. https://www.economist.com/science-and-technology/2023/09/13/how-scientists-are-using-artificial-intelligence Accessed: 2023-09-22.
[6] A. Borzunov et al. 2022. Petals: Collaborative Inference and Fine-tuning of Large Models. (2022). arXiv:2209.01188.
[7] A. Borzunov et al. 2022. Training transformers together. In *NeurIPS 2021 Competition and Demonstration Track*. PMLR.

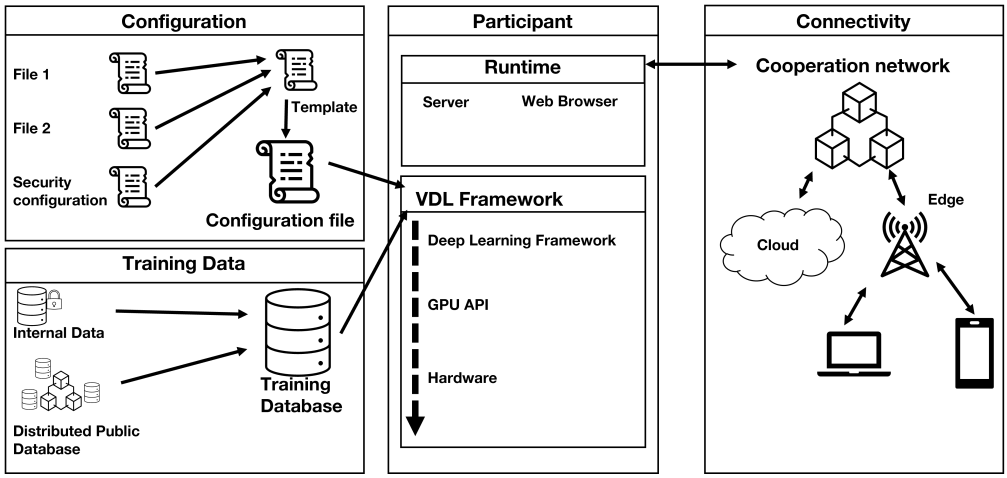


Figure 5. VDL framework using FDFL: high-level overview

[8] A. L. Beberg et al. 2009. Folding@home: Lessons from eight years of volunteer distributed computing. In *IPDPS*.

[9] E. Hallaji et al. 2023. Federated and Transfer Learning: A Survey on Adversaries and Defense Mechanisms. In *Federated and Transfer Learning*, R. Razavi-Far et al. (Ed.). Springer, 29–55.

[10] E. Korpela et al. 2001. SETI@home-massively distributed computing for SETI. *Comput. Sci. Eng.* 3 (2001), 78–83.

[11] E. Kijsipongse et al. 2018. A hybrid GPU cluster and volunteer computing platform for scalable deep learning. *J. Supercomput.* 74 (2018), 3236–3263.

[12] H. B. McMahan et al. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

[13] H. Wang et al. 2020. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *NeurIPS*.

[14] H. Wang et al. 2023. Scientific discovery in the age of artificial intelligence. *Nature* 620, 7972 (2023), 47–60.

[15] I. Messadi et al. 2022. SplitBFT: Improving Byzantine Fault Tolerance Safety Using Trusted Compartments. In *Middleware*.

[16] K. Singhal et al. 2023. Large language models encode clinical knowledge. *Nature* 620 (2023), 172–180.

[17] M. Abadi et al. 2016. Deep learning with differential privacy. In *CCS*.

[18] M. Atre et al. 2021. Distributed deep learning using volunteer computing-like paradigm. In *IPDPS Workshops*.

[19] M. Diskin et al. 2021. Distributed deep learning in open collaborations. In *NeurIPS*.

[20] M. Hirth et al. 2011. Anatomy of a Crowdsourcing Platform - Using the Example of Microworkers.com. In *IMIS*.

[21] M. Mansouri et al. 2023. SoK: Secure Aggregation Based on Cryptographic Schemes for Federated Learning. In *PoPETs*.

[22] M. Ryabinin et al. 2021. Moshpit SGD: Communication-Efficient Decentralized Training on Heterogeneous Unreliable Devices. In *NeurIPS*.

[23] M. Ryabinin et al. 2023. SWARM Parallelism: Training Large Models Can Be Surprisingly Communication-Efficient. *arXiv preprint arXiv:2301.11913* (2023).

[24] M. Xue et al. 2020. Machine Learning Security: Threats, Countermeasures, and Evaluations. *IEEE Access* 8 (2020), 74720–74742.

[25] P. Kairouz et al. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210.

[26] S. Prakash et al. 2022. Byzantine-Resilient Federated Learning with Heterogeneous Data Distribution. (2022). arXiv:2010.07541.

[27] T. Brown et al. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

[28] T. D. Nguyen et al. 2022. FLAME: Taming Backdoors in Federated Learning. In *USENIX Security*.

[29] Y. Huang et al. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *NeurIPS*.

[30] Z. Tang et al. 2023. FusionAI: Decentralized Training and Deploying LLMs with Massive Consumer-Level GPUs. In *LLM-IJCAI*.

[31] R. Guerraoui, N. Gupta, and R. Pinot. 2023. Byzantine Machine Learning: A Primer. *ACM Comput. Surv.* (2023).

[32] D. De Lacour, M. Lacoste, M. Südholt, and J. Traoré. 2023. Towards Scalable Resilient Federated Learning: A Fully Decentralised Approach. In *PerCom Workshop PeRConAI*.

[33] EPFL Machine Learning and Optimization Laboratory. 2023. DISCO - Distributed Collaborative Machine Learning. https://github.com/epfml/disco Accessed: 2023-09-22.

[34] T. M. Mengistu and D. Che. 2019. Survey and Taxonomy of Volunteer Computing. *ACM Comput. Surv.* 52, 3 (2019).

[35] United Nations. 2023. Sustainable Development Goals. https://unstats.un.org/sdgs Accessed: 2023-09-22.

[36] OpenAI. 2023. Duolingo Max. https://openai.com/customer-stories/duolingo Accessed: 2023-09-22.

[37] M. Ryabinin and A. Gusev. 2020. Towards Crowdsourced Training of Large Neural Networks using Decentralized Mixture-of-Experts. In *NeurIPS*.

[38] P. Shinde and S. Shah. 2018. A Review of Machine Learning and Deep Learning Applications. In *ICCUBEA*.

[39] Learning@home team. 2020. Hivemind: a Library for Decentralized Deep Learning. https://github.com/learning-at-home/hivemind Accessed: 2023-09-22.

[40] F. Tung and G. Mori. 2020. Deep Neural Network Compression by In-Parallel Pruning-Quantization. *TPAMI* 42, 03 (2020), 568–579.