MatchCurv: Communication-Efficient Decentralized Federated Learning in Heterogeneous Environments

Harsha Praneeth Dussa, Janek Haberer, Olaf Landsiedel
Kiel University, Germany
{janek.haberer,olaf.landsiedel}@cs.uni-kiel.de

ABSTRACT

Federated learning offers a privacy-preserving method for training machine learning models. Yet, traditional centralized federated learning has drawbacks like single points of failure and communication bottlenecks. While decentralized federated learning has been proposed to overcome these limitations, challenges such as statistical and system heterogeneity remain.

Whereas most works focus on solving only one of these challenges, this paper introduces MatchCurv, a decentralized federated learning framework designed to handle statistical and system heterogeneity while improving communication efficiency. In our evaluation, for example, a multi-layer perceptron with two hidden layers of 128 units each shows a significant accuracy increase when using our framework compared to PD-SGD, achieving up to 17 percentage points more accuracy. The source code is available at https://github.com/ds-kiel/matchcurv.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Human-centered computing** → *Ubiquitous and mobile computing*.

KEYWORDS

Deep Learning, Federated Learning, Decentralized Federated Learning, MATCHA, FedCurv

1 INTRODUCTION

The proliferation of connected devices equipped with sensors has led to the generation of vast amounts of data, presenting ample opportunities for leveraging advanced data analysis techniques like machine learning (ML) [4]. However, the decentralized nature of this data presents challenges in its practical utilization, mainly due to stringent privacy regulations [12].

In 2016, McMahan et al. introduced Federated Learning (FL) [14] as a solution to data privacy concerns, showcasing its efficacy through next-word prediction on smartphones, see Figure 1. In FL, individual (edge) devices locally train statistical models and transmit only model updates to a central server. After aggregating these updates to create a consolidated model, the model is deployed back to the edge devices and can undergo further refinement. This ensures data privacy by preserving data at its source while enabling collaborative model training. Decentralized Federated Learning (DFL) extends FL by eliminating the need for a central server, relying instead on peer-to-peer communication among devices [7].

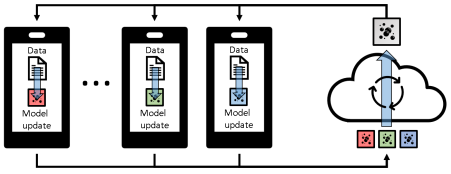


Figure 1: In Federated Learning, devices like smartphones train a model and send model updates to a central server. The server combines these updates into a new model and redistributes them back to the clients for further enhancement. This process continues until the desired model performance is reached.

The disparity, however, in computational and communication capabilities among edge devices, referred to as system heterogeneity, complicates the orchestration of the training process [9]. Additionally, the data collected by individual devices often exhibits non-identical characteristics. This is referred to as statistical heterogeneity and further complicates the training of a model capable of effectively capturing variations across different devices [8].

This paper introduces MatchCurv, a novel DFL framework designed to manage statistical and system heterogeneity while enhancing communication efficiency. MatchCurv consists of three key design elements:

- **Enhancing Communication Efficiency:** Devices share parameters more frequently with peers important for information distribution in the network rather than with less significant ones. This minimizes communication expenses while guaranteeing efficient information dissemination.
- **Tackling Statistical Heterogeneity:** We strategically apply penalties to models that diverge from the global model. This allows the trained model to adjust to diverse data distributions across devices, enhancing generalization and performance.
- **Addressing System Heterogeneity:** We introduce a deadline to interrupt model training in slower devices and accommodate partial solutions, effectively preventing them from impeding the overall training.

In a scenario of 10 devices with data distributed in a non-IID way, 25% of the devices being stragglers, and a reduced communication budget of 50% per device, MatchCurv achieves 89.27% accuracy on Fashion-MNIST [19] compared to traditional periodic decentralized stochastic gradient descent (PD-SGD) achieving 72.11% accuracy, highlighting the efficiency of our approach in DFL.

In MatchCurv, we combine principles from centralized FL and distributed ML, tailoring them to the requirements of DFL. This adaptation ensures compatibility and synergy between these methodologies, enhancing communication and effectively managing statistical and system heterogeneity within the decentralized environment. We also release the source code necessary to run and evaluate the framework.

The remainder of this paper is structured as follows: Section 2 discusses the background and related works. Section 3 explains the design and main algorithm. Then, in Section 4, we evaluate our framework concerning all the mentioned challenges. And finally, in Section 5, we conclude.

2 BACKGROUND & RELATED WORK

Federated Learning is a solution to train models on unreliable edge devices such as smartphones and IoT devices, which are crucial for privacy-preserving applications. DFL improves upon centralized FL (CFL) by empowering individual devices to initiate training and make decisions about the training process [7]. Each device collects updates from neighbors, aggregates them locally, and shares responsibilities traditionally held by a central server. DFL offers a decentralized approach suitable for scenarios where employing a central entity is challenging, potentially improving scalability. Nevertheless, several other challenges persist in Federated Learning.

Communication Efficiency: DFL requires devices to connect with all other peers to share the model updates. Selective communication with a subset of neighbors is crucial to alleviate this burden. However, this approach may impede information dissemination, emphasizing the significance of strategic neighbor selection. While many works focus on improving communication efficiency in FL, such as FedAvg [14], NetMax [21], and Matcha[18], they only focus on improving communication ignoring statistical or system heterogeneity.

Statistical Heterogeneity: The inherent heterogeneity of data collected by individual devices presents a challenge in achieving models capable of effectively capturing variations across different devices. Works such as FedProx [10], FedCurv [16], and others [20] tackle this issue but either assume that all devices participate in each round, need to share local data, or ignore system heterogeneity.

System Heterogeneity: System heterogeneity, arising from differences in hardware and network connectivity, can cause devices to lag behind during training. To address this, FedProx [10] and FedHP [11] assign multiple local updates based on device capabilities. Adapting this approach to DFL, we introduce a time-based deadline to halt training on slower devices, effectively managing system heterogeneity without requiring a central entity to assign tasks.

While many works tackle these challenges in CFL, in MatchCurv, we leverage multiple strengths of selected works to provide a comprehensive solution to the challenges encountered in DFL. We address both statistical and system heterogeneity, all while improving communication efficiency.

3 DESIGN

This section explains how we incorporate different works to tackle the aforementioned challenges.

Communication Efficiency In DFL, devices establish an ad-hoc network represented as a graph. When selecting a subset of neighbors, choosing a subgraph with higher algebraic connectivity is advantageous. Improved algebraic connectivity decreases characteristic path length, promoting faster information propagation and mitigating the effects of communicating with only a subset of peers [13, 15].

The Matcha framework [18] divides the graph into matchings (subgraphs), with each matching's probability reflecting its contribution to algebraic connectivity λ_2. Following model training, matchings are randomly selected based on probabilities, and devices exchange model updates along corresponding edges. The probabilities P for the m matchings are calculated by solving the following:

$$\begin{aligned} &\max_{P} g(P) := \lambda_2 \left(\sum_{j=1}^{m} p_j L_j \right) \\ &\text{subject to} \sum_{j=1}^{m} p_j \leq m C_b \wedge 0 \leq p_j \leq 1 \end{aligned} \tag{1}$$

Here, p_j represents the probability assigned to the j^{th} matching and L_j is the Laplacian matrix of the j^{th} matching. Introducing a communication budget ($C_b \in [0, 1]$) determines the size of the neighbor subset chosen for communication. For instance, setting $C_b = 0.1$ achieves a 10× reduction in communications compared to $C_b = 1$.

In MatchCurv, we leverage the device topology to enhance communication efficiency. While protocols like NetMax [21] improve efficiency by prioritizing high-speed links, this approach may not optimize information dissemination. Instead of relying on local heuristics like link speed, we utilize MATCHA to prioritize links based on algebraic connectivity. The consensus step facilitates acquiring the training topology, ensuring all devices share the same network view.

Statistical Heterogeneity In FL optimization, the objective is to minimize the following:

$$F(w) = \sum_{k \in K} q_k F_k(w) \tag{2}$$

In a set of devices K, F_k is the local objective function of device k. Here, q_k determines the relative impact of each device [14].

FedCurv strategically penalizes parameter deviations w by utilizing the Fisher Information Matrix (FIM) [5, 16]. The optimization problem in FedCurv is:

$$\begin{aligned} \min_{w} \hat{F}_{t,k}(w) = F_k(w) \\ + \lambda \sum_{j \in K \setminus k} (w - w_{t-1,j})^T \mathrm{diag}(I_{t-1,j})(w - w_{t-1,j}) \end{aligned} \tag{3}$$

Here, $w_{t-1,j}$ and $I_{t-1,j}$ denote the model parameters and FIM for device j in the $(t-1)^{th}$ training round, respectively. Meanwhile, F_k is the optimization problem from Equation 2 corresponding to device k.

With this, FedCurv effectively addresses statistical heterogeneity, but its reliance on a central entity limits its applicability to

decentralized settings. In DFL, device access to parameters and FIMs is restricted based on the training topology. Thus, we focus on gathering parameters and FIMs solely from neighboring devices, extending FedCurv to the DFL context and showing its effectiveness in managing statistical heterogeneity within a decentralized environment, even in the presence of systems heterogeneity and reduced communication.

Therefore, in MatchCurv, the local update step uses Equation 3 from FedCurv to address statistical heterogeneity, surpassing alternatives like FedProx.

System Heterogeneity Traditional FL usually ignores devices that lag behind in completing local updates, often dropping or sampling devices based on their capabilities, which can introduce bias. Thus, the local update step in our framework adheres to a time-based deadline, which is also not reliant on a central entity. This way, slow devices train until the deadline and still contribute with their partial solution instead of being dropped or causing delays due to others waiting for them.

The iterative training and parameter-sharing process continues for a predetermined number of rounds or until we achieve the desired model performance. We integrate these mechanisms into our framework, including the ability to deploy on Raspberry Pis (RPis). See Algorithm 1 for an outline of MatchCurv's implementation in six distinct steps.

Algorithm 1: MatchCurv Algorithm

```
   /* Initiating Training                                   */
1  Broadcast invitation and hyperparameters.
2  Wait for responses from interested devices.
3  N := {v|v ∈ UIDs of Responding Devices}
   /* Topology Consensus Step                               */
4  Construct Graph G(V, E) from N
5  Ḡ(V, E) := ∅
6  while Ḡ ≠ G do
7      G := Ḡ ⋃ G
8      Send G to neighbors.
9      Ḡ := merge received graphs
   /* Matcha Phase                                          */
10 M := ∅
11 while |E| > 0 do
12     Add maximal matching m from edges E to M.
13     Remove matching from E.
14 Solve for matching probabilities P using Equation (1).
   /* Start Time Consensus Step                             */
15 t := proposed starting time
16 t̄ := −1
17 while t̄ ≠ t do
18     t := max(t̄, t)
19     Send proposed time t to neighbors.
20     t̄ := maximum of received values
21 Wait until t.
22 foreach round ∈ [1, r] do
       /* Local Model Updates                               */
23     Update model using Equation (3) until timeout.
24     Compute Fisher Information Matrix (FIM).
25     Test the model performance.
       /* Sharing model updates                             */
26     Select M̄ ⊆ M randomly based on P
27     Send model updates and FIM to neighbors in M̄.
28     Wait for replies until timeout.
29     Merge received models and Save FIMs.
30 Save training results.
```

4 EVALUATION

In our evaluation, we compare MatchCurv to Periodic Decentralized Stochastic Gradient Descent (PD-SGD) with a simple fully connected neural network on the MNIST [3] dataset. At the end of the evaluation, we also train a network with two hidden layers of 128 units each and compare our implementation on Fashion-MNIST [19] to show our framework's robustness. Initially, to establish a baseline, we train the model for 1000 epochs using traditional machine learning methods, achieving a baseline test accuracy of 92%. We then train the model using the depicted topology in Figure 2 employing DFL. Each training round comprises ten epochs, with 50 training rounds in total. Like an ablation study, we evaluate statistical heterogeneity, system heterogeneity, and communication efficiency separately before combining all challenges and evaluating MatchCurv in a realistic scenario. We run all experiments in a simple network consisting of ten Raspberry Pis (Raspberry Pi 3 Model B) and verify the results in a simulation. The software stack is kept relatively simple, relying on Python, the pip package NetworkX [1] for communication, and the packages NumPy [6], SciPy [17], and TensorFlow [2] for implementing the algorithm.

4.1 Performance under Statistical Heterogeneity

To assess MatchCurv's performance under Statistical Heterogeneity, we divide the MNIST dataset among ten devices. The distribution scenarios include:

- IID (Independent and Identically Distributed): Each device receives 10% of all samples, including all digits.
- non-IID-5: Each device receives 20% of all samples, restricted to 5 digits specific to the device. The distribution ensures no overlap between devices while the samples across all devices still correspond to the initial dataset.
- non-IID-2: Each device receives 50% of all samples, restricted to 2 digits specific to the device.

In non-IID scenarios, conventional SGD leads to overfitting individual models to local data, causing divergence and poor generalization of the global data distribution. This results in a notable decline in model performance, as indicated in Table 1. In contrast, FedCurv mitigates these issues by incorporating a penalty into the objective function, preventing model divergence. Figure 3a shows that FedCurv significantly boosts accuracy, matching IID performance

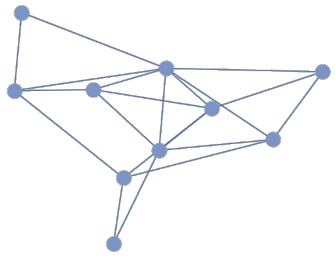


Figure 2: Our framework implementation ensures that devices form a fully connected network. However, to mimic real-world scenarios, we enforce the depicted preset topology.

Configuration	IID	non-IID-5	non-IID-2
Test Accuracy [%]	91.69	87.93	77.62

Table 1: Showcase of the performance of the model across different data distribution scenarios. The IID setting achieves accuracy close to the ML baseline, while accuracy notably decreases in non-IID settings.

% of Stragglers	0	25	50
Test Accuracy [%]	77.62	60.11	51.65

Table 2: Stragglers with non-IID-2 data negatively affect the model's performance, reducing accuracy to 60.11% and 51.65% with 25% and 50% stragglers, respectively.

with 91.64% accuracy in the non-IID-5 setting. In non-IID-2, accuracy surges by over eight percentage points from 77.62% to 85.72%. Moreover, FedCurv expedites accuracy attainment; in non-IID-2, we reach 77.62% accuracy by the 13th round, compared to 50 rounds without FedCurv. This approach significantly improves model accuracy compared to plain SGD, effectively addressing challenges associated with statistical heterogeneity.

4.2 Performance under System Heterogeneity

We randomly designate a percentage of devices as stragglers to simulate variations in computational capabilities. Stragglers, unable to complete their assigned work in time, are ignored during training, potentially affecting model accuracy. While IID data typically shows minimal performance decline due to stragglers, non-IID scenarios exhibit decreased performance, as shown in Table 2. However, due to the devices having non-IID data, it is crucial to include slow devices to reach generalization.

Figure 3b shows that when dealing with 25% of stragglers, integrating FedCurv alone increases accuracy from 60.11% to 68.39%. Using an interrupt strategy, where we interrupt devices during training with a fixed training time, yields a test accuracy of 77.97%.

Communication Budget	100%	50%	25%
Test Accuracy [%]	77.62	56.31	43.98

Table 3: Showcase of the effect of reduced communication and randomly assigned probabilities.

This shows how valuable partial work is, especially in non-IID scenarios. Instead of waiting for them to fully complete their training, which would slow down the overall system and increase training time, we can include their partial work to achieve faster generalization. By integrating both approaches — accepting partial work from stragglers and using FedCurv — we attain an accuracy of 85.59% regardless of stragglers, surpassing the baseline performance of 77.62% without any stragglers under non-IID-2 data. Similarly, Figure 3c shows that with 50% of stragglers, their combination achieves the highest accuracy of 85.64%. This indicates that increasing the number of stragglers does not affect the accuracy much as long as we use at least FedCurv or the interrupt strategy.

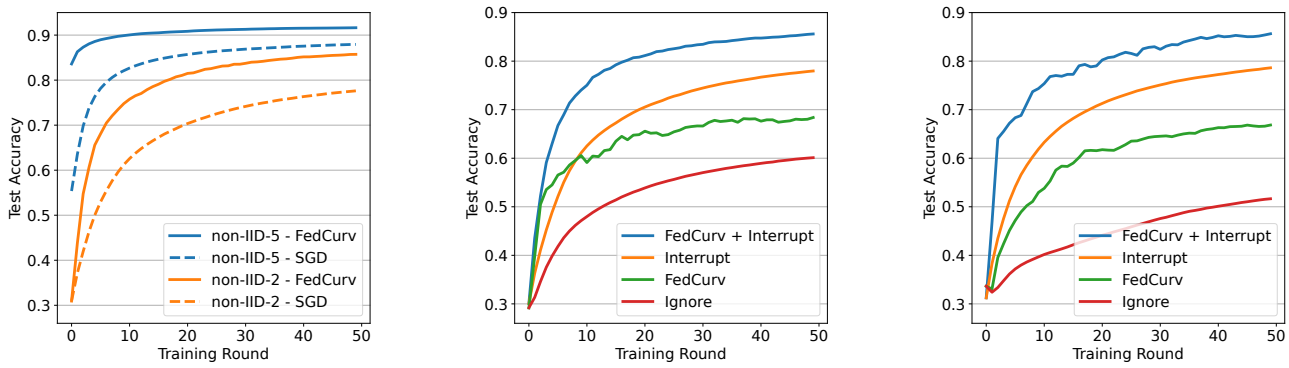
4.3 Performance under a Communication budget

As it is common to perform Federated Learning with mobile devices, such as smartphones, keeping communication to a minimum is vital, as it is very costly. To alleviate the communication burden on devices, we introduce a communication budget (Cb), inspired by MATCHA. Conducting experiments in a non-IID-2 setting, we randomly assign these probabilities instead of using Equation 1 to compute them. The model performance significantly drops under a reduced Cb, as shown in Table 3, because devices get fewer updates from their peers.

Reduced communication results in overfitting to local data. This manifests as significant fluctuations in the model's accuracy. Figure 4a shows that with a 50% Cb, employing FedCurv improves the accuracy from 56.32% to 71.81%. Implementing MATCHA, using algebraic connectivity for strategic probability assignment, gets 75.48%, underscoring the significance of strategic neighbor selection for exchanging model parameters. However, combining MATCHA with FedCurv results in a substantial increase of accuracy to 84.23% with reduced fluctuation. With only 50% Cb, this combination surpasses the baseline accuracy of 77.62% with 100% Cb. Similarly, with only 25% Cb, the combined MATCHA + FedCurv configuration achieves an accuracy of 75.43% from the initial 43.98%, which is an even greater jump.

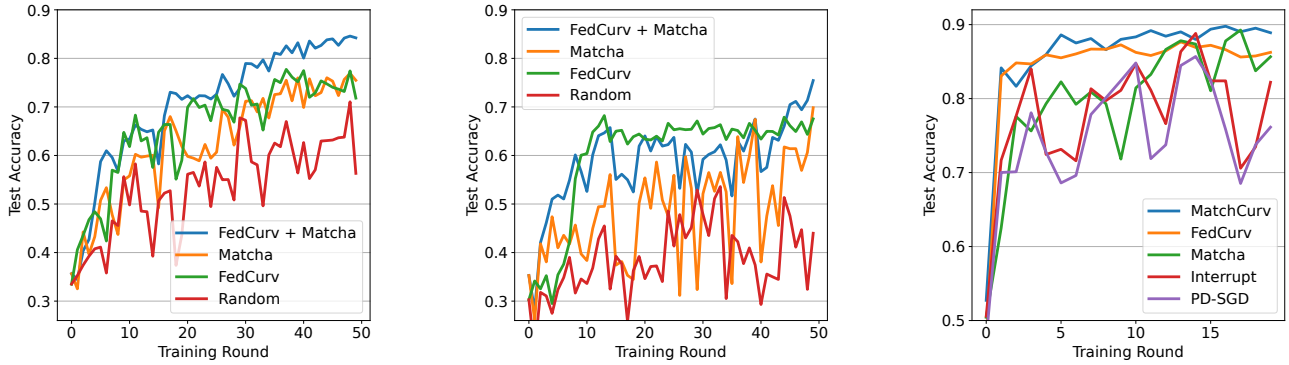
4.4 MatchCurv vs. PD-SGD

In Table 4, we summarize the previous results and see that combining FedCurv and the interrupt strategy deals the best with stragglers while combining FedCurv and MATCHA deals the best with a reduced communication budget. To thoroughly evaluate MatchCurv, which combines FedCurv, MATCHA, and the interrupt strategy and its resilience in real-world scenarios, we designate 25% of devices as stragglers and allocate a 50% Cb simultaneously. The data distribution follows the non-IID-5 pattern, and the experiment consists of



(a) non-IID-2 vs. non-IID-5 (b) 25% Stragglers (c) 50% Stragglers

Figure 3: (a) Comparison of FedCurv and SGD in non-IID scenarios. (b) Comparison of FedCurv, interrupt strategy, and their combination to deal with 25% stragglers in the non-IID-2 scenario. (c) Same scenario as (b) but with 50% stragglers.



(a) 50% Communication budget (b) 25% Communication budget (c) MatchCurv vs PD-SGD

Figure 4: (a) Comparison of MATCHA, FedCurv, and its combination to deal with a communication budget of 50% in the non-IID-2 scenario. (b) Same scenario as (a) but with a communication budget of 25%. (c) Comparison of PD-SGD, interrupt strategy, MATCHA, FedCurv, and MatchCurv in the non-IID-5 scenario with 25% stragglers and a communication budget of 50%.

100 training rounds. These conditions provide an ideal environment for assessing the robustness of our framework.

Our proposed strategies enhance model performance compared to PD-SGD, as shown in Table 5. Interrupting stragglers achieves 83.52% accuracy, compared to PD-SGD's 77.95% accuracy. MATCHA attains 85.72% accuracy, and FedCurv reaches 86.03% accuracy. Notably, when combining all strategies with MatchCurv, we achieve the best performance of 89.89% accuracy, nearing our initial baseline of 91.69% from conventional ML. Furthermore, the model's performance using MatchCurv shows reduced fluctuations and a consistent upward trajectory, as shown in Figure 4c.

To assess the framework's robustness, we conduct identical experiments using a fully connected neural network with two hidden layers of 128 units each, this time with the Fashion-MNIST dataset [19] for image classification. Results demonstrate a consistent trend, with MatchCurv improving the performance from 72.11% to 89.27% compared to PD-SGD.

Therefore, MatchCurv effectively reduces communication demands, manages statistical and systems heterogeneity, and ultimately improves model performance in DFL.

5 CONCLUSION

We create a robust Decentralized Federated Learning framework prioritizing communication efficiency and addressing statistical and system heterogeneity. By merging MATCHA and FedCurv, we optimize information sharing and statistical heterogeneity handling. Additionally, we incorporate the concept of accepting partial work from slower devices to manage system heterogeneity. Our framework, MatchCurv, closely matches traditional Machine Learning and outperforms PD-SGD in various experiments, enhancing

Strategy	Comm. Budget [%]	Stragglers [%]	Accuracy [%]
Ignore	100	25	60.11
FedCurv	100	25	68.39
Interrupt	100	25	77.97
FedCurv + Interrupt	100	25	**85.59**
Ignore	100	50	51.65
FedCurv	100	50	66.83
Interrupt	100	50	78.62
FedCurv + Interrupt	100	50	**85.64**
Random	50	0	56.32
FedCurv	50	0	71.81
MATCHA	50	0	75.48
FedCurv + MATCHA	50	0	**84.22**
Random	25	0	43.98
FedCurv	25	0	67.53
MATCHA	25	0	69.83
FedCurv + MATCHA	25	0	**75.43**

Table 4: The final accuracies of all strategies in the evaluated scenarios.

Config	Test Accuracy [%]
PD-SGD	77.95
Interrupt	83.52
MATCHA	85.72
FedCurv	86.03
MatchCurv	**89.89**

Table 5: Model performance of PD-SGD, MatchCurv, and individual strategies to address the challenges.

communication efficiency and achieving up to 17 p.p. more accuracy on Fashion-MNIST while reducing training time. Moving forward, we plan to enhance security and efficiency through parameter compression, encryption, and differential privacy measures. We also intend to tackle device dropouts and explore asynchronous training methods to further boost the framework's robustness and scalability.

ACKNOWLEDGMENTS

This project has received funding from the Federal Ministry for Economic Affairs and Climate Action under the Marispace-X project grant no. 68GX21002E.

REFERENCES

[1] 2008. Exploring Network Structure, Dynamics, and Function Using NetworkX - SciPy Proceedings. https://proceedings.scipy.org/articles/TCWV9851.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[3] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[4] Zakaria Abou El Houda, Abdelhakim Hafid, and Lyes Khoukhi. 2019. Co-IoT: A Collaborative DDoS Mitigation Scheme in IoT Environment Based on Blockchain Using SDN. In *2019 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9013542

[5] Kazuya Fujita, Kensuke Okada, and Kentaro Katahira. 2022. The Fisher information matrix: A tutorial for calculation for decision making models. https://doi.org/10.31234/osf.io/hdwut

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[7] Lie He, An Bian, and Martin Jaggi. 2018. Cola: Decentralized linear learning. *Advances in Neural Information Processing Systems* 31 (2018).

[8] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. 2020. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*. PMLR, 4387–4398.

[9] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60. https://doi.org/10.1109/msp.2020.2975749

[10] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[11] Yunming Liao, Yang Xu, Hongli Xu, Lun Wang, and Chen Qian. 2023. Adaptive configuration for heterogeneous participants in decentralized federated learning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[12] Misbah Liaqat, Victor Chang, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Toseef, Umar Shoaib, and Rana Liaqat Ali. 2017. Federated cloud resource management: review and discussion. *Journal of Network and Computer Applications* 77 (2017), 87–105.

[13] William Liu, Harsha Sirisena, Krzysztof Pawlikowski, and Allan McInnes. 2009. Utility of algebraic connectivity metric in topology design of survivable networks. In *2009 7th International Workshop on Design of Reliable Communication Networks*. IEEE, 131–138.

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

[15] Milena Oehlers and Benjamin Fabian. 2021. Graph Metrics for Internet Robustness – A Survey. (2021). https://doi.org/10.48550/ARXIV.2103.05554 arXiv:2103.05554 [cs.NI]

[16] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. 2019. Overcoming Forgetting in Federated Learning on Non-IID Data. arXiv:1910.07796 [cs.LG]

[17] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[18] Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. 2019. MATCHA: Speeding up decentralized SGD via matching decomposition sampling. In *2019 Sixth Indian Control Conference (ICC)*. IEEE, 299–300.

[19] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747 [cs.LG]

[20] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

[21] Pan Zhou, Qian Lin, Dumitrel Loghin, Beng Chin Ooi, Yuncheng Wu, and Hongfang Yu. 2021. Communication-efficient decentralized machine learning over heterogeneous networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 384–395.