Scalable and Portable Federated Learning Simulation Engine

Borja Arroyo Galende
ETSIT, Universidad Politécnica de Madrid
Madrid, Spain
borja.arroyog@upm.es

Juan Mata Naranjo
CINECA
Rome, Italy
j.matanaranjo@cineca.it

Silvia Uribe Mayoral
ETSISI, Universidad Politécnica de Madrid
Madrid, Spain
silviaalba.uribe@upm.es

ABSTRACT

Federated learning (FL) is one of the most promising approaches to ensure privacy in the application of data-driven techniques to sensitive information. However, the implementation of such approaches in a production environment is still an important challenge. In this paper, we present a scalable, portable, hardware-independent, model-agnostic FL Simulation Engine (FLSE) with the aim of easing the job of researchers who want to train FL models to be deployed in production environments. The FLSE offers a tool that can be used both standalone or embedded within a larger architecture, it can be deployed seamlessly and allows concurrent, scalable, and highly available V&V assessment support for FL models. The tool allows researchers to understand the behaviour, in terms of metric performance, of their proposed models in production scenarios, allowing a boost in trustworthiness towards ethical AI.

CCS CONCEPTS

• **Computing methodologies** → **Model verification and validation**; **Distributed algorithms**; **Learning paradigms**; • **Applied computing** → **Health care information systems**.

KEYWORDS

Federated Learning, Scalability, Portability, Machine Learning, Data Privacy, Cloud Computing, Simulation

ACM Reference Format:
Borja Arroyo Galende, Juan Mata Naranjo, and Silvia Uribe Mayoral. 2023. Scalable and Portable Federated Learning Simulation Engine. In *3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum (ESAAM 2023), October 17, 2023, Ludwigsburg, Germany*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3624486.3624488

1 INTRODUCTION

Machine learning (ML) and, in general, artificial intelligence (AI) have surpassed numerous barriers in recent decades [8], which have led it to become one of the most promising and fast-growing[1] scientific areas [5, 18, 19]. Some of these barriers have been overcome thanks to the increase in computational power [20], in conjunction with the collection and storage of huge amounts of data [14]. However, new challenges for the coming decades are related, among others, to fairness [15, 17], explainability [7, 13], and privacy [9], all of which are still very active areas of research [10].

In relation to privacy and from an AI point of view, one of the main limitations encountered occurs when the data necessary to model a problem are owned by multiple organisations and there are one or several regulations imposing data protection, e.g., in the clinical domain, sensitive information cannot be freely shared due to privacy restrictions [16]. This limitation considerably reduces the expressiveness of data-driven solutions and hinders cross-institution collaboration.

One of the most promising approaches to overcome this issue is Federated Learning (FL) [11], where raw data is kept isolated and secure, while allowing models to train at multiple data sites without human intervention or raw data transfer. FL achieves data isolation by exchanging model parameters rather than data. This exchange is performed, as commonly denoted in the literature, between server and client nodes. The server (central) node orchestrates most of the FL workflow, while the client (local) nodes are hosted alongside the isolated data. This method can still be prone to privacy violations; however, new techniques can effectively lead to actual regulatory-compliant, privacy-preserving, data-driven solutions [2].

Multiple sectors, such as healthcare, IoT, etc. [12, 21] have started to benefit from this growing technique [3]. However, training such models is not a trivial task in a production environment. This is due to the leap in knowledge compared to classical ML techniques together with a complex distributed setup, usually preventing data sites from adopting these approaches to learning.

The problem therein becomes: is there a way to leverage the adoption of ML techniques in highly constrained environments? This question triggered the design and implementation of a novel scalable, portable, hardware-independent, model-agnostic Federated Learning Simulation Engine (FLSE) with the aim of easing the job of researchers who want to train FL models to be deployed in a production environment.

2 CONTRIBUTIONS

In the context of FL, based on our experiences, there are three limitations that normally burden its applicability in production environments[2].

First, the lack of computing resources that is frequently related to FL application environments. As an example, clinical sites tend to offer more humble hardware compared to data centres, so avoiding unnecessary usage of these resources or, in other words, optimising the criteria for running computations seems to be a must to leverage

[1] See results of 2022 McKinsey Global Survey on AI

[2] In contrast to production environments, development environments usually allow researchers to access all data partitions across nodes.

ESAAM 2023, October 17, 2023, Ludwigsburg, Germany

ACM ISBN 979-8-4007-0835-0/23/10…$15.00
https://doi.org/10.1145/3624486.3624488

the potential of FL, especially on the local nodes. Furthermore, this situation can occur even if clinical sites opt for cloud providers, so lowering the costs as much as possible is mandatory.

Secondly, closely related to the previous issue, integrating traditional machine learning models into an FL framework requires additional domain knowledge. Moreover, some FL frameworks require several setup steps to run a minimal example. This can limit the adoption of such a technology, which in some scenarios prohibits performing data analysis with enough samples.

Finally, and most importantly, researchers in the field of FL often find it difficult to understand the behaviour, in terms of metric performance, of their proposed models in production scenarios. In that regard, it would be desirable to simulate an FL training job on a number of related use cases to obtain relevant metrics to understand model's performance, rather than directly addressing the real use case.

To solve the previous limitations, we propose a highly available concurrent simulation framework built on top of Flower [1] that allows both a standalone or a plug-in solution. Specifically, the latter supports Flip [6], which is the original design criterion, or other FL platforms. This tool simplifies the researcher's development workflow so that target evaluation use cases can be managed by the tool itself with little additional steps. When the AI practitioner evaluates a new model, it is automatically tested against all scenarios described in a use case. These relationships are described in Figure 1 using an example.

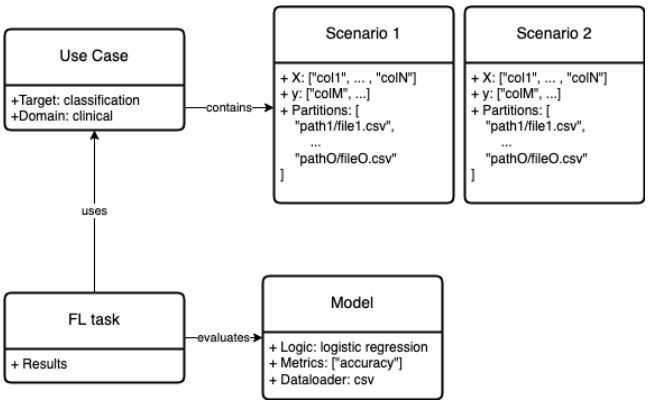

Figure 1: Example ER diagram of the FLSE.

3 METHODOLOGY

An outline of the methodology designed to implement the FLSE is presented in the current section. We begin by enumerating the functionalities offered by the tool, followed by the design of the main function required to run the simulation and finally all of the components which convey the principal properties of the tool, i.e. scalability and portability. It is important to note that the design criteria do not directly address any of the issues mentioned in Section 2. Contributions, as those are not software requirements; instead it is a mean to create a product which is able to solve all of the previous short-comings.

3.1 System Functionalities

The tool offers an automatic federated learning task orchestration system, supporting the management of target use cases. The functionalities provided by the tool are the following.

- Management of use cases, which, as shown in Figure 1, are composed of a collection of scenarios and some metadata on which to perform the simulation study. The user is able to add/modify use cases in order to suit her needs. Each scenario contains information on the paths to the datasets on which federation will be performed. The multiple datasets represent the number of client nodes involved in the simulation. In addition, each scenario has information on the names of independent and dependent variables used to run the entire simulation process. The end user can select the use cases according to the similarity with respect to the task on which the model needs to be assessed.
- Support[3] the Verification and Validation (V&V) process on FL models. V&V is a way to determine whether a model is, respectively, error-free (well implemented), due to simulation run with no failures; and well performing, via performance metrics. The FL model is provided to the system by the user, which is automatically evaluated against the selected use case (e.g. for binary classification problems in some clinical domain, the scenarios described in the use case must be strongly related). In particular, the V&V is conducted in the following manner:
  - *Verify* that the FL model is implemented correctly by performing error checks in model load, data load, local learning, global aggregation, etc. Tips are given to the user to improve implementation workflows.
  - In case of an error-free execution, *validate* that the FL model performs adequately in the use case by returning a set of user-defined metrics.

3.2 Design Criteria

The most important function is to evaluate the FL model on a specific target use case. This action triggers a sequence of steps that are automatically run by an executor entity. This entity sends the aggregation strategy to the server and the local learners to each separate client. A single local learning stage (Flower client training) is composed by fitting the local model over the datasets defined in the scenario and returning the model parameters to the server. The server then aggregates the model parameters of all clients based on the aggregation strategy, and a new iteration starts based on the results of the aggregation. The execution finishes once the number of federated epochs matches the value requested by the user (this parameter along with other modelling parameters are user-defined). Figure 2 describes how processes are run and when and how information exchanges occur during Flower execution.

The executor is implemented in Python in such a way that each component, i.e. servers and clients, run on independent processes. Processes communicate with each other through gRCP; in particular, communication occurs between client processes and the server process. The execution logs and outputs generated by each process

[3]The system remains as a support tool as there is no automatic step to assess metrics. This design seems less error-prone than fully automatic procedure.

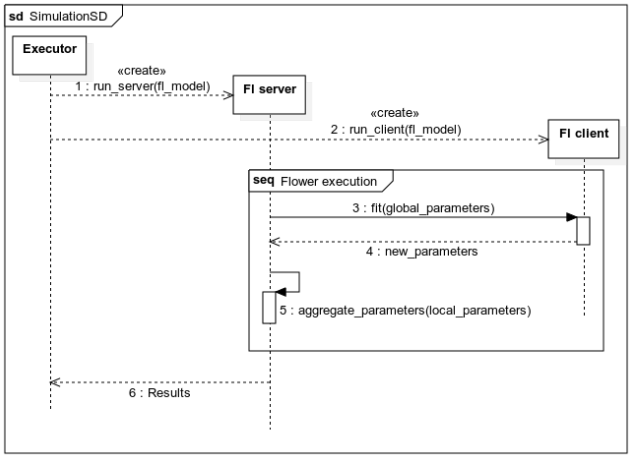

Figure 2: Sequence diagram illustrating the simulation flow occurring in the executor entity.

are collected and returned to the user along with the results. Frequent health checks are performed to ensure proper execution of each process. Given that the environments required by each of the entities, clients and server, are slightly different, their dependencies have been optimised in such a way that only the essential packages are installed. Thus, the forked processes get linked to a suitable Python environment.

The interface that a researcher must build to integrate with the engine, which inherits from MLflow [4] allowing for better model management, encapsulates both the local learner and the aggregation strategy of the FL workflow. This interface is depicted in the form of an example in Listing 1. It is designed with the intention of simplifying the implementation for the user while maintaining the necessary level of flexibility.

Listing 1: FL model Interface

```
import mlflow

class FLModel(mlflow.pyfunc.PythonModel):

    def create_strategy(self):
        import flwr
        return flwr.server.strategy.FedAvg()

    def create_model(self):
        import tensorflow as tf
        model = tf.keras.applications.MobileNetV2((32, 32, 3),
                classes=10, weights=None)
        model.compile()

        class PythonModelWrapper:
            def __init__(self, model):
                self.model = model
            def predict(self, context, x):
                return self.model.predict(x)
            def fit(self, x_train,y_train,
                    batch_size=32, local_epochs=1):
                self.model.fit(x_train, y_train,
                    epochs=local_epochs,
                    batch_size=batch_size)
            def get_weight(self):
                return self.model.get_parameters()
            def set_weight(self,parameters):
                return self.model.set_parameters(parameters)
            def evaluate(self,x_test,y_test):
                return self.model.evaluate(x_test, y_test)
        return PythonModelWrapper(model=model)
```

3.3 Component Description

To guarantee isolation and modularity, a containarized approach is leveraged. Therefore, the application consists of a stack of microservices designed with the principles of high availability, high concurrency, and scalability.

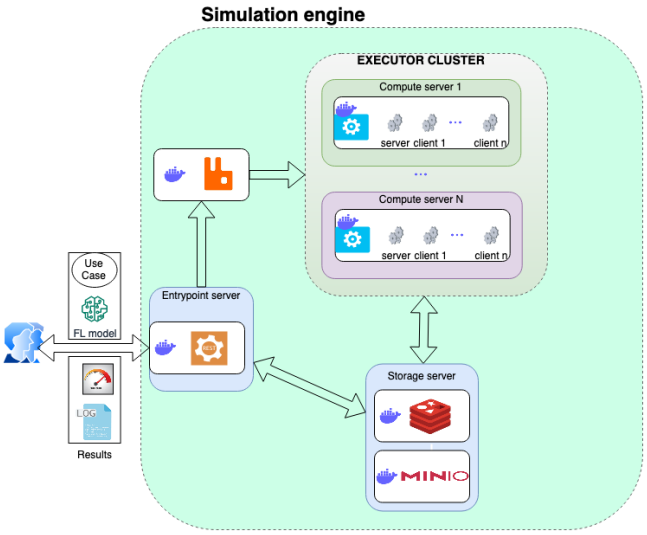

Figure 3: Architecture diagram showing the main components of the system; from left to right, a REST API receiving requests, a pub/sub broker, a persistent storage system and the executor cluster. Arrows indicate intercontainer information flows

A brief introduction to each microservice can be found in the next bullet list.

- RestAPI for external access to the system. Allows for use case management and FL task administration. Exposes endpoints, among others, for use case definition and FL model simulation.
- Pub/Sub Message Broker (Rabbitmq). Handles message exchanges between internal components.
- Persistent Object Storage System (Minio). Stores data files for use cases, FL models, simulation results, logs, and any artefact related to the simulation process.
- Results database (RedisDB). Caches the task ID, task status, and URL of artefact folders in Minio.
- Simulation executor. Runs the federated training by collecting the desired data and the FL model to run a Flower FL execution (clients and server) through forking processes.

All microservices except the latter run the official Docker images, including the RESTful API container that is based on FastAPI. The latest image uses a Conda customised setup, which supports the installation and management of different Python environments on

the same system. This criterion is established due to the fact that (1) local and global sides do not have the same dependencies, hence there should be different environments for each; (2) local side should support different libraries commonly used in traditional machine learning, thus there should be separate environments for each group of libraries; (3) Python versions and dependency versions are quite difficult to handle, but with Conda we can even define different environments for distinct dependency versions.

4 APPLICATION SCENARIOS AND DISCUSSIONS

The solution outlined throughout this article has been adopted in Genomed4All, a European funded project[4]. The scope of this project is to pool clinical and "-omics" data and exploit cross-border and multi-center data to advance research in personalised medicine following strict GDPR privacy policies.

In this specific case, the FLSE is an auxiliary step to facilitate the complete FL workflow. In particular, when the user requests to upload an FL model, it is first simulated with the engine. The engine assesses a proper and exception-free execution, providing an understanding of its performance over a set of related scenarios. Lastly, and based on these results, the user (or committee) can decide whether the FL model is transferred into a production ready stage or dropped. This idea is depicted in Figure 4.

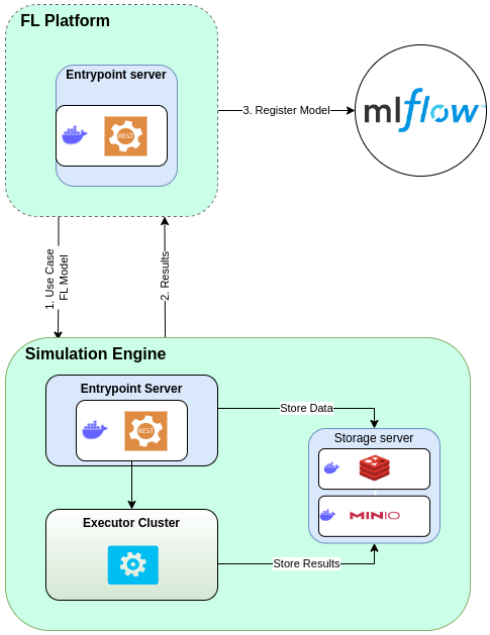

Figure 4: FLSE success flow in communication diagram.

The proposed solution has opened up the use of the FL platform in a critical moment of the project due to the following reasons:

(1) It has provided a way to clearly identify bugs in the code, either in the FL system or in the FL model code. Therefore, bugs can be isolated and solved quickly.
(2) It has avoided the unnecessary usage of computing resources within data providers' machines, which allows for cheaper infrastructure costs.
(3) It has allowed AI researchers to incrementally develop and test their FL models given the leap in complexity from traditional machine learning to FL.
(4) It has increased trustworthiness of the FL platform which leads to better exploiting the potential of FL in such healthcare European projects.

Depending on the requirements of the project, the proposed solution can be adapted to have a proper balance in terms of usability, complexity, and scalability. Thus, the levels of automation regarding FLOps that can be achieved are depicted below.

- Level 0: No automation. The user manually triggers the experiments, as is the case for Flower.
- Level 1: Automatic training and evaluation. The user specifies the config for a FL job run, which gets queued and runs automatically.
- Level 2: Automatic verification of models. Plus error free check of the models.
- Level 3: Automatic validation of models. Plus complying with acceptance criteria via metrics.
- Level 4: Automatic retraining of models. Plus automatic training on data arrival or user demand.
- Level 5: Automatic deployment. Plus continuous deployment approach.

Moving up the automation ladder implies a huge leap in complexity; therefore, there should exist evidence supporting a more mature operational level in order not to overcomplicate the software solution. In practice, due to project requirements, FLSE has been implemented to enable level 2 of automation.

5 CONCLUSIONS & FUTURE WORK

Novel approaches capable of solving GDPR compliance issues are currently leveraging the power of machine learning in complex environments. As such, FL offers a solution to train machine learning models over decentralised data, but it also comes at the cost of a leap in complexity. To simplify FL adoption, we present a simulation engine for federated learning, with the aim of tackling several issues and concerns that are normally related to this field.

The FLSE offers a tool that can be used both standalone or embedded within a larger architecture, it can be deployed anywhere and allows for concurrent, scalable, and highly available V&V assessment support for FL models. In addition, it effectively supports AI practitioners in the process of integrating FL driven model designs and to grasp model performance prior to production scenarios, allowing for a boost in trustworthiness towards ethical AI.

Lastly, for future work, there are several improvements that would benefit the overall functionalities of the system, including, among others, automatic handling of hyperparameter tuning, establishing the computational resources of the local nodes during simulations, and adding support for baselines. These upgrades could allow to reduce the need for human interaction.

[4]See Genomed4All web-page

ACKNOWLEDGMENTS

GenoMed4All has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101017549. The authors would like to thank all GenoMed4All members for their helpful contributions and insightful discussions.

REFERENCES

[1] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. 2022. Flower: A Friendly Federated Learning Research Framework. https://doi.org/10.48550/arXiv.2007.14390 arXiv:2007.14390 [cs, stat].

[2] Alberto Blanco-Justicia, David Sánchez, Josep Domingo-Ferrer, and Krishnamurty Muralidhar. 2022. A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning. *Comput. Surveys* 55, 8 (2022), 160:1–160:16. https://doi.org/10.1145/3547139

[3] Alissa Brauneck, Louisa Schmalhorst, Mohammad Mahdi Kazemi Majdabadi, Mohammad Bakhtiari, Uwe Völker, Jan Baumbach, Linda Baumbach, and Gabriele Buchholtz. 2023. Federated Machine Learning, Privacy-Enhancing Technologies, and Data Protection Laws in Medical Research: Scoping Review. *Journal of Medical Internet Research* 25, 1 (March 2023), e41588. https://doi.org/10.2196/41588 Company: Journal of Medical Internet Research Distributor: Journal of Medical Internet Research Institution: Journal of Medical Internet Research Label: Journal of Medical Internet Research Publisher: JMIR Publications Inc., Toronto, Canada.

[4] Andrew Chen, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Clemens Mewald, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Avesh Singh, Fen Xie, Matei Zaharia, Richard Zang, Juntai Zheng, and Corey Zumar. 2020. Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning (DEEM'20)*. Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3399579.3399867

[5] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. 2021. Advancing mathematics by guiding human intuition with AI. *Nature* 600, 7887 (Dec. 2021), 70–74. https://doi.org/10.1038/s41586-021-04086-x Number: 7887 Publisher: Nature Publishing Group.

[6] Borja Arroyo Galende, Silvia Uribe Mayoral, Francisco Moreno García, and Santiago Barrio Lottmann. 2023. FLIP: A New Approach for Easing the Use of Federated Learning. *Applied Sciences* 13, 6 (March 2023), 3446. https://doi.org/10.3390/app13063446

[7] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. 2019. XAI—Explainable artificial intelligence. *Science Robotics* 4, 37 (Dec. 2019), eaay7120. https://doi.org/10.1126/scirobotics.aay7120 Publisher: American Association for the Advancement of Science.

[8] Tim Hwang. 2018. Computational Power and the Social Impact of Artificial Intelligence. *SSRN Electronic Journal* (2018). https://doi.org/10.2139/ssrn.3147971

[9] Zhanglong Ji, Zachary C. Lipton, and Charles Elkan. 2014. Differential Privacy and Machine Learning: a Survey and Review. https://doi.org/10.48550/arXiv.1412.7584 arXiv:1412.7584 [cs].

[10] Anna Jobin, Marcello Ienca, and Effy Vayena. 2019. The global landscape of AI ethics guidelines. *Nature Machine Intelligence* 1, 9 (Sept. 2019), 389–399. https://doi.org/10.1038/s42256-019-0088-2 Number: 9 Publisher: Nature Publishing Group.

[11] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2017. Federated Learning: Strategies for Improving Communication Efficiency. https://doi.org/10.48550/arXiv.1610.05492 arXiv:1610.05492 [cs].

[12] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. 2020. A review of applications in federated learning. *Computers & Industrial Engineering* 149 (Nov. 2020), 106854. https://doi.org/10.1016/j.cie.2020.106854

[13] Ričards Marcinkevičs and Julia E. Vogt. 2023. Interpretability and Explainability: A Machine Learning Zoo Mini-tour. http://arxiv.org/abs/2012.01805 arXiv:2012.01805 [cs].

[14] Luca Mascetti, Maria Arsuaga Rios, Enrico Bocchi, Joao Calado Vicente, Belinda Chan Kwok Cheong, Diogo Castro, Julien Collet, Cristian Contescu, Hugo Gonzalez Labrador, Jan Iven, Massimo Lamanna, Giuseppe Lo Presti, Theofilos Mouratidis, Jakub T. Mościcki, Paul Musset, Remy Pelletier, Roberto Valverde Cameselle, and Daniel Van Der Ster. 2020. CERN Disk Storage Services: Report from last data taking, evolution and future outlook towards Exabyte-scale storage. *EPJ Web of Conferences* 245 (2020), 04038. https://doi.org/10.1051/epjconf/202024504038 Publisher: EDP Sciences.

[15] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *Comput. Surveys* 54, 6 (July 2021), 115:1–115:35. https://doi.org/10.1145/3457607

[16] Liam O'Neill, Franklin Dexter, and Nan Zhang. 2016. The Risks to Patient Privacy from Publishing Data from Clinical Anesthesia Studies. *Anesthesia & Analgesia* 122, 6 (June 2016), 2017–2027. https://doi.org/10.1213/ANE.0000000000001331

[17] Dana Pessach and Erez Shmueli. 2022. A Review on Fairness in Machine Learning. *Comput. Surveys* 55, 3 (Feb. 2022), 51:1–51:44. https://doi.org/10.1145/3494672

[18] Manav Raj and Robert Seamans. 2019. Primer on artificial intelligence and robotics. *Journal of Organization Design* 8, 1 (May 2019), 11. https://doi.org/10.1186/s41469-019-0050-0

[19] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J. Topol. 2022. AI in health and medicine. *Nature Medicine* 28, 1 (Jan. 2022), 31–38. https://doi.org/10.1038/s41591-021-01614-0 Number: 1 Publisher: Nature Publishing Group.

[20] John Shalf. 2020. The future of computing beyond Moore's Law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 378, 2166 (Jan. 2020), 20190061. https://doi.org/10.1098/rsta.2019.0061 Publisher: Royal Society.

[21] Jie Xu, Benjamin S. Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. 2021. Federated Learning for Healthcare Informatics. *Journal of Healthcare Informatics Research* 5, 1 (March 2021), 1–19. https://doi.org/10.1007/s41666-020-00082-4