Federated Learning using Peer-to-peer Network for Decentralized Orchestration of Model Weights

Monik Raj Behera
Blockchain Engineering
JPMorgan Chase & Co
Bengaluru, India
monik.r.behera@jpmorgan.com

Sudhir Upadhyay
Blockchain Engineering
JPMorgan Chase & Co
New York, United States of America
sudhir.x.upadhyay@jpmorgan.com

Robert Otter
Blockchain Engineering
JPMorgan Chase & Co
London, United Kingdom
robert.otter@jpmorgan.com

Suresh Shetty
Blockchain Engineering
JPMorgan Chase & Co
New York, United States of America
suresh.shetty@jpmorgan.com

***Abstract*—In recent times, Machine learning and Artificial intelligence have become one of the key emerging fields of computer science. Many researchers and businesses are benefited by machine learning models that are trained by data processing at scale. However, machine learning, and particularly Deep Learning requires large amounts of data, that in several instances are proprietary and confidential to many businesses. In order to respect individual organization's privacy in collaborative machine learning, federated learning could play a crucial role. Such implementations of privacy preserving federated learning find applicability in various ecosystems like finance, health care, legal, research and other fields that require preservation of privacy. However, many such implementations are driven by a centralized architecture in the network, where the aggregator node becomes the single point of failure, and is also expected with lots of computing resources at its disposal. In this paper, we propose an approach of implementing a decentralized, peer-to-peer federated learning framework, that leverages RAFT based aggregator selection. The proposal hinges on that fact that there is no one permanent aggregator, but instead a transient, time based elected leader, which will aggregate the models from all the peers in the network. The leader (aggregator) publishes the aggregated model on the network, for everyone to consume. Along with peer-to-peer network and RAFT based aggregator selection, the framework uses dynamic generation of cryptographic keys, to create a more secure mechanism for delivery of models within the network. The key rotation also ensures anonymity of the sender on the network too. Experiments conducted in the paper, verifies the usage of peer-to-peer network for creating a resilient federated learning network. Although the proposed solution uses an artificial neural network in it's reference implementation, the generic design of the framework can accommodate any federated learning model within the network.**

***Index Terms*—federated learning, machine learning, privacy, distributed, decentralized, blockchain, peer-to-peer, p2p**

I. Introduction

In the era of distributed computing, scale of data and computational resources are amazingly handled by distribution of workloads across multiple systems in horizontal direction. Distributed computing, specific to "distributed machine learning", opens up exciting opportunities. However, it also introduces new challenges, in the areas where data privacy and data security are important. Further, designing a resilient, highly available and robust ecosystem for machine learning is equally challenging. Federated learning [2] lays the foundation of implementing machine learning, on a distributed landscape [1], where heterogeneous machines can participate in collaborative manner. The main idea is to have an "aggregator node" and set of "participant nodes" in the federated network. Participant nodes send their local model gradients to the aggregator node, and aggregator node collectively composes all the received models together into a singular global model, which is again sent back to all participant nodes. In this setup, the participants benefit from common learning, without knowing about all the underlying data. While federated learning does take care of privacy, security and anonymity, aggregator nodes become the single point of failure in the federated network. This posses the risk of downtime in case of unwanted failures, halting critical process on the network.

In current available implementations, all of them have centralized design of an aggregator node and various participant nodes, and they form a star topology for communication. Implementations like Tensorflow Federated [5] and PySyft [6] rely upon centralized servers to do the final aggregation on local gradient outputs. There is an interesting aspect presented by implementation of Owkin [7], which talks about utilizing "Hyperledger" distributed ledger technology [8] for orchestration of non-sensitive metadata and exchange of algorithms to individual nodes. If the computing nodes are interested, they may publish the results of computations on DLT, for everyone's benefit.

To design a resilient and highly- available aggregator service through decentralization, we propose an approach of implementing federated learning using peer-to-peer [3] communication among the nodes in federated networks. The idea is to have all the nodes with equivalent capabilities on the network and, all of them should be able to communicate with each other. The network, at any given point of time, has only one node, that would be acting as a “leader” and rest of the nodes will act as “followers”. In case the leader crashes, the network is intelligent enough to conduct a new leader election, and select leader among the remaining peers. The said mechanism, follows a raft consensus [4] to conduct elections and assign leaders. Hence, leader would responsible for acting as transient aggregator for the network. Further, to distribute the work load among peers, and give a fair chance to every peer within the network to assume the role of a “leader”, a new leader is elected after a predefined time interval is elapsed.

In order to throw light on key differences of the proposed solution and related works, the solution proposed here is dependent upon the fundamental idea of raft based consensus, to elect a “transient leader” within the network. The aggregation occurs only at the leader node. To compare with implementations involving centralized servers, we propose the concept of floating, transient leaders, being elected through raft consensus. If the current leader becomes unavailable or unresponsive for any reason, the network is capable of recovering, by electing a new leader among the peers, by conducting fair election. This ensures each node gets a fair chance of performing aggregation. In comparison to implementation based on DLT, we are using peer-to-peer communication approach, which is depending on RPC mechanism [9], to provide reliable, highly available and robust messaging layer. This essentially removes the computational overhead of maintaining the data consistency over the network too, like in case of DLT, but still makes the network decentralized in nature.

II. Peer-to-Peer Federated Learning

In the case of peer-to-peer network, and in our case, federated learning implemented on peer-to-peer communication, we have three roles. In the proposed implementation, any healthy peer could potentially assume any of the roles -

- Leader (Aggregator) node - *for aggregation role*
- Follower (Participant) node - *participants in the network*
- Candidate node - *potential peers, who contest in the election*

To facilitate the communication between multiple nodes and elected aggregator independently, the implementation leverages the secured peer-to-peer communication as detailed in the ‘experimental setup’ sub-section. We have implemented peer-to-peer communication between the nodes. The peer-to-peer communication between different participants leverages the existing remote procedure calls, using TCP [10] layer of networking. Since it’s a peer-to-peer communication, all the messages, sent by a specific node are sent as broadcast to the network. All the messages from a given node are broadcast to the network, just as in any other p2p network. However, based on the message type and if encrypted, only the intended recipient can read the message. If the messages are not encrypted, all the nodes can read and perform necessary action. Further, for a p2p network to be effective and efficient, all the nodes should be able to discoverable and reachable peers present in the network. This may require opening necessary enterprise firewalls in a safe and secure manner.

Since the leader role is transient for every node, either reassigned after a certain time or failure, the RSA [11] public and private keys associated in encryption need to be dynamic in nature. The pair of RSA keys are to be rotated for every message, initiated by the leader/participant node. After a successful leader election, the leader announces it’s RSA public key to the network that can be then used by the participants to encrypt any message that it intends to send to the leader.

A. Architecture

As depicted in Figure 1, all the peers are connected to each other through mesh topology, where any peer can communicate with any other given peer though remote procedure calls. The major components of federated learning in peer-to-peer setup include:

1) Leader election and resignation
2) Handling fault and re-election
3) Model aggregation by the leader
4) Model information broadcast by the leader
5) Model fetch, initiated by follower

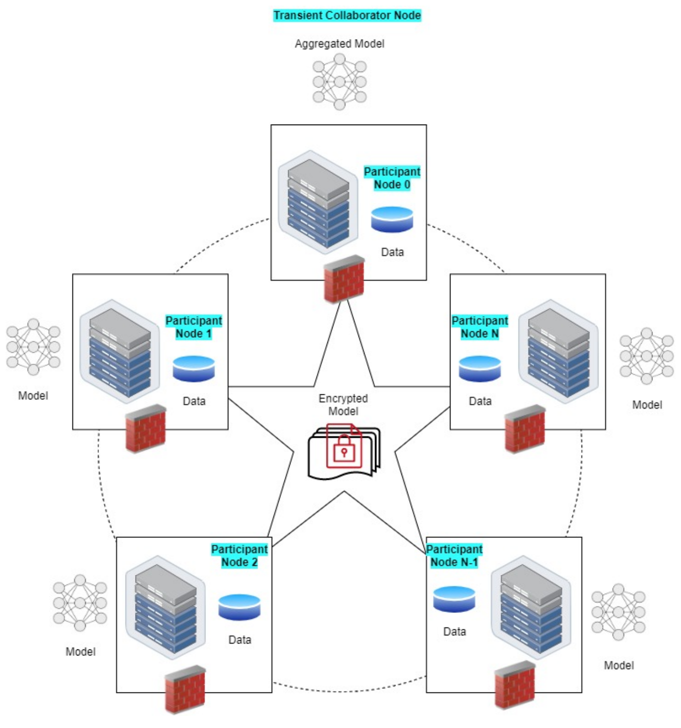

Fig. 1. Architecture diagram with peer-to-peer network, where one of the peers gets elected as transient leader, which aggregates local gradients of remaining peers

A leader, when elected by raft consensus algorithm, assumes the role of the model aggregator, which is transient by design. In normal network operation, the elected node stays as leader for T period, as governed by RAFT design. A leader is elected only for a specific duration, and at the end of which another leader is elected from the available nodes. In order to ensure the leader is available for that duration, all the participant nodes expect a heartbeat every predefined frequency from the leader. A missed heartbeat is an indication of unavailability of the leader. In such a case, some peers from the network may promote themselves as candidate nodes for the next leader. After a fair election, one of the candidate nodes gets elected as leader and the rest of the nodes continue to remain followers till the next election. The life-cycle of a raft consensus and leader election associated with it, is explained in Figure 2

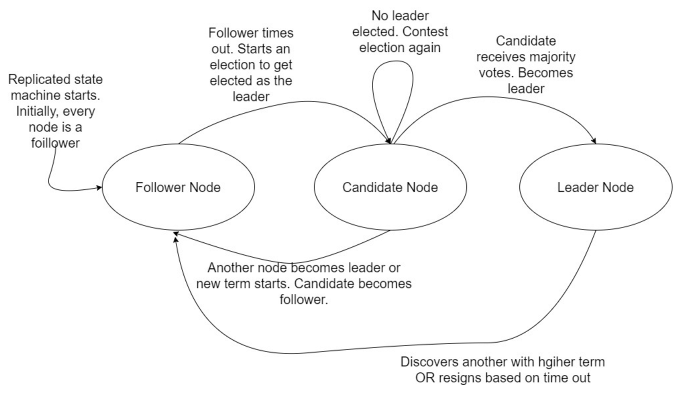
Replicated state machine starts. Initially, every node is a foillower
Follower times out. Starts an election to get elected as the leader
No leader elected. Contest election again
Candidate receives majority votes. Becomes leader
Follower Node
Candidate Node
Leader Node
Another node becomes leader or new term starts. Candidate becomes follower.
Discovers another with hgiher term OR resigns based on time out

Fig. 2. DFA state machine transitions depicted for a raft consensus algorithm. This explains the state transition between follower, candidate and leader roles for every node, present in the peer-to-peer network

After the network assigns leadership to the elected node, the node behaves as an aggregator node, starting with sending broadcast messages with it's newly generated RSA key pair's public key. This key would be used by any peer that intends to send an encrypted message to the leader. The message payload can contain information related to model exchange, updates, etc to the aggregator node. By design, all the messages will be sent as broadcast on the network. The unencrypted messages are available for all of the participants and each participant can read and act as necessary. However, the messages that are intended for certain participants would be encrypted and only nodes with valid private keys can decrypt the message. Sending local model updates by participant nodes to the leader node is a classic example of using encrypted messages for communication between participant and leader node. Leader node sends the model's information, at a given frequency to all the participant nodes.

Suppose a node doesn't have the latest model version, that node would initiate an encrypted message for the leader node on the network. To ensure only the leader can read the payload, those messages would be encrypted with the leader's published public key. Along with the encrypted cipher, for every new message, the node would generate a new pair of RSA keys, whose public key would be sent with the encrypted message. The generated privately key will remain with the participant node itself. As the message reaches the leader node, it can decipher it using it's own private key. Rest of the other nodes will receive the message too, but can't decipher the payload. After the leader has processed the message, it will generate and publish a response that will be encrypted by a public key sent by the participant. This ensures that only the participant that is intended to receive the message can decrypt, while the rest of nodes can not.

This mechanism is central to "dynamic message encryption-decryption". As is the nature of the network peer-to-peer all the nodes are aware of each other, there is a need to secure the data in-transit, from malicious actors. Added to that, in federated learning setup, privacy of data is of utmost concern.

Hence maintaining the privacy, anonymity and security of message is a fundamental requirement by design. One more benefit of "dynamic message encryption-decryption" is the implicit anonymity which is achieved in the federated network. Since the RSA key pairs of participant nodes are rotating for every message, there is no way a leader node can identify or tag an incoming message to a specific node, making it truly anonymous in nature, from a message flow perspective. In Figure 3, a standard "dynamic message encryption-decryption" life cycle is depicted as DFA state machines, which is actively used in peer-to-peer federated learning described in the paper.

Description of the states are as follows -

- S1 : Participant node encrypts the message with aggregator node's public key.
- S2 : Aggregator node decrypts the message using private key of itself
- S3 : Aggregator node encrypts the message using participant node's public key
- S4 : Participant node decrypts the message using the private key generated in the initial state

$$\begin{tikzcd} \arrow[r] & S1 \arrow[r] & S2 \arrow[r] & S3 \arrow[r] & \boxed{S4} \end{tikzcd}$$

Fig. 3. State diagram *(Life-cycle)* of "dynamic message encryption-decryption", as described in the paper. Here $S1$ is the initial state, when a arbitrary participant node generates a new RSA key pair for communication. $S4$ is the final state, when participant node deciphers the response message from leader node using the generated private key. $S2$ and $S3$ are intermediate states as described above

B. Experimental Setup

In order to validate the proposed solution and it's hypothesis in achieving the acceptable accuracy score in the federated network, we did set up an experimental environment. In the given set-up, we used MNIST [12] data. We created a network with 5 nodes. Each node has RHEL 8.0 OS installed with Python 3.8 installed, along with all the supporting libraries like socket, Tensorflow 2.0, requests, etc. The services are designed to run as a self-sufficient daemon in the node, which can run the machine learning computations on data. Each participant node has 4GB RAM with 2 core processor. The relevant ports and firewalls are setup among all the nodes for communication. Since this is an experimental setup, all the nodes are situated in a single subnet, to ease out the network complexity.

We have added artificial neural network machine learning model in the federated framework. The neural network [14] being used has one input layer, with 128 input neurons, one hidden layer of 2096 neurons and one output layer with 10 neurons. Figure 4 depicts the model graphically.

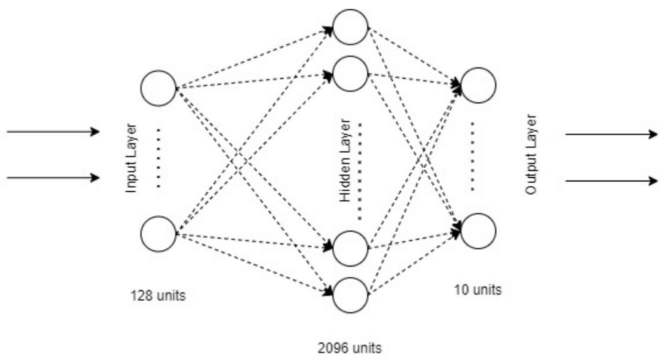

Fig. 4. Architecture diagram of neural network model for hand-written digit recognition using MNIST data

In the execution of setup, we studied the relation between loss and learning rate of adam's optimizer [13], by changing the value of learning rate, as depicted in Figure 5. We also studied accuracy and it's relation with learning rate, as shown in Figure 6. Table I represents loss and accuracy of all models in the federated network described above. Table concludes that *using federated computation, we achieved better accuracy over time*.

TABLE I
LOSS AND ACCURACY OF LOCAL AND GLOBAL MODEL OVER NETWORK

Node	*Loss*	*Accuracy (in %)*
Global Model - Initial	**2.4663**	**91.03**
Participant 1	1.8932	94.74
Participant 2	1.656	94.89
Participant 3	1.823	94.23
Participant 4	1.8688	94.41
Participant 5	1.6311	94.58
Global Model - Aggregated	**0.72**	**96.04**

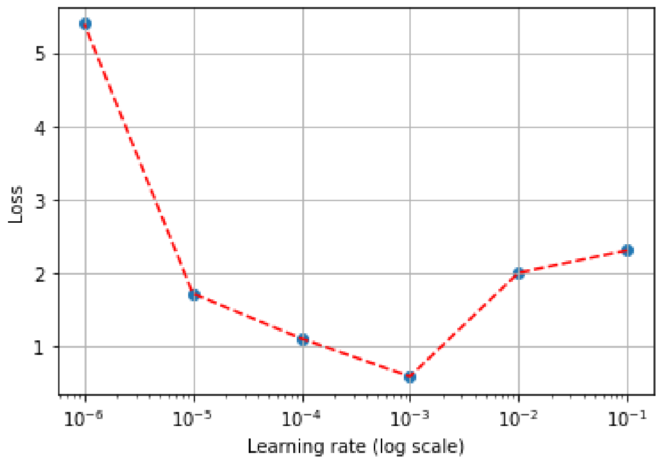

Fig. 5. Loss vs learning rate of the neural network implemented for MNIST data based federated learning, which uses Adam optimizer

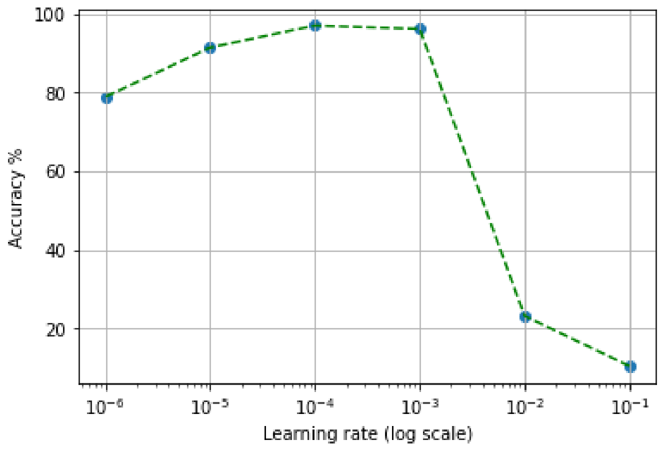

Fig. 6. Accuracy (in %) vs learning rate of the neural network implemented for MNIST data based federated learning, which uses Adam optimizer

Along with federated learning experimentation, to verify the resiliency of the system, the leader node was knowingly shut down, in order to simulate node failure. As expected, after 20 seconds, a new node becomes the leader in the network. Since the leader's heartbeat was missed, the network was intelligent enough to initiate election, as a process of self healing from leader failure in the network. The network was also run for 7 days, which is essentially 7 leader time-outs, considering the pre-defined "leader resignation timeout" is 24 hours. Hence leaders resigned and went back to become followers, initiating elections in the network. This fundamentally shares the responsibility and work-load among the peers, on a time sharing basis. The key idea here is to run two set of functions, one being in active and other being in inactive (temporarily sleeping) mode. Whenever a node is acting as a leader, both the set of functions, denoting roles of "participant" and "aggregator" are run in active-active mode, in two separate, independent threads. Whereas, in case of nodes acting as participants, the set of functions for "participant" and "aggregator" are run in active-inactive mode, in two separate threads. There is some added latency, which needs to be accounted for in the election process and synchronizing heartbeats. Table II shows the observation during the raft election, initiated during network start, resignation and shutdown of leaders. In the table, "F" means follower node state, "C" means candidate node state and "L" means leader node state.

C. Results

Based on the experiments carried out earlier, the following inferences have been drawn from the results-

1) In case of MNIST data, the average accuracy of all the individual nodes combined together is 94.57%, whereas after using federated computations, within the framework proposed, the accuracy is increased by 1.47%, making the federated accuracy as 96.04%.
2) The proposed federated learning framework is capable of performing the federated computations on variety of ma-

TABLE II
STATE OF NODES IN EXPERIMENTAL SETUP DURING RAFT ELECTION

Time	*Node 0*	*Node 1*	*Node 2*	*Node 3*	*Node 4*
t_0	Election started				
t_1	**F**	**F**	**F**	**F**	**C**
t_2	**F**	**F**	**C**	**F**	**C**
t_3	Voting started				
t_4	Voting completed and leader elected				
t_5	**F**	**F**	**F**	**F**	**L**
...	... *(m-1)* time units elapsed				
t_m	Leader resigns/becomes inactive				
t_{m+1}	**F**	**F**	**F**	**F**	**F**
t_{m+2}	Election started				
t_{m+3}	**F**	**F**	**C**	**F**	**F**
t_{m+4}	**F**	**C**	**C**	**F**	**F**
t_{m+5}	Voting started				
t_{m+6}	Voting completed and leader elected				
t_{m+7}	**F**	**L**	**F**	**F**	**F**
...	...				

chine learning models, using peer-to-peer networking, which implicitly provides decentralization, resilience, security and anonymity, with keeping the performance of models intact.

3) It is to be noted here that true anonymity is proportional to stringent access to the network and nodes. Hence special care needs to be taken in maintaining the control in access of these said systems.
4) The peer-to-peer network, implemented for federated learning works similarly, it works in case of federated learning with centralized servers. Additionally it makes the solution, a truly decentralized one, with implicit resiliency.
5) Peer-to-peer federated learning assumes that every node, present in the federated network is capable of being an aggregator node. Hence it mandates all the nodes to be suitable for carrying out high power computations. This requires the nodes in the network to be homogenous in nature, unlike the federated learning in centralized ecosystem, where participating nodes can be comparatively less resourceful than aggregator node.
6) The federated network is capable of running any given machine learning model, as long as the global aggregation functions are implemented by every node on the network.
7) The network has no single point of failure, because of mesh topology and floating role of "aggregator". With increase in challenges of maintaining the networking infrastructure and compute resources, within the network, this makes the solution proposed more enterprise friendly, where data privacy is of utmost importance.

III. CONCLUSION

In the paper, we discussed details of implementing a peer-to-peer federated learning, and how that would benefit the enterprises on a decentralized network of information exchange. The primary benefit of using a peer-to-peer federated network is to make federated learning decentralized, resilient by removing single point of failure and giving every node on the network, a fair share of aggregation role. The experiments conducted in-lieu of the paper validates our hypothesis of using federated learning within a peer-to-peer network, and conducting rotation of "leader" role among the peers. From the results and observations, we infer that network is self-healing in case of unwanted failures.

Peer-to-peer federated learning has great potential in ecosystems, where participants want to share the learning from models and their individual data, without compromising the privacy of their own data. For architectures demanding decentralization of resources, and yet maintaining resilient and privacy respecting federated network, peer-to-peer federated learning brings a lot of exciting potential. Needless to say, peer-to-peer framework can also be the foundation for creating smart, self healing networks, specially in case of critical business applications.

The proposed solution puts forward challenges, where there is a theoretical opportunity to distribute the aggregation among sub-set of participant nodes, to further optimize the network. With peer-to-peer communication, maintenance of networking between each and every node is a challenge from security point of view. The proposed solution can be further improved by leveraging enhancements in the current consensus algorithms around authorization and security of the overall network. These enhancements coupled with proposed solution can bring exciting opportunities in the areas of distributed and privacy preserving machine learning domain.

ACKNOWLEDGMENT

We would like to thank our colleagues Brett J Sanford, Sean J Moran, Tulasi D Mova, Jacob Mendel, Constantin Gonciulea, Thomas Eapen, Vinay Somashekar, Antonios Georgiadis, Razi Abbas, Debidutta P Samantaray, Chang Y Jiao, Fran Silavong and Sanket Kamthe for reviews and design discussions. We would also like to thank JPMorgan Chase & Co for supporting the effort.

REFERENCES

[1] Bonawitz, Keith, et al. "Towards federated learning at scale: System design." arXiv preprint arXiv:1902.01046 (2019).
[2] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.
[3] Fox, Geoffrey. "Peer-to-peer networks." Computing in Science & Engineering 3.3 (2001): 75-77.
[4] Ongaro, Diego, and John Ousterhout. "In search of an understandable consensus algorithm." 2014 USENIX Annual Technical Conference (USENIXATC 14). 2014.
[5] 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, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, 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. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[6] Ryffel, Theo, et al. "A generic framework for privacy preserving deep learning." arXiv preprint arXiv:1811.04017 (2018).

[7] Galtier, Mathieu N., and Camille Marini. ”Substra: a framework for privacy-preserving, traceable and collaborative Machine Learning.” arXiv preprint arXiv:1910.11567 (2019).

[8] Maull, Roger, et al. ”Distributed ledger technology: Applications and implications.” Strategic Change 26.5 (2017): 481-489.

[9] Nelson, Bruce Jay. ”REMOTE PROCEDURE CALL.” (1982): 4863-4863.

[10] Jorgensen, Jacob W. ”Transmission control protocol/internet protocol (TCP/IP) packet-centric wireless point to multi-point (PTMP) transmission system architecture.” U.S. Patent No. 6,862,622. 1 Mar. 2005.

[11] Bleichenbacher, Daniel. ”Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS 1.” Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1998.

[12] Deng, Li. ”The mnist database of handwritten digit images for machine learning research [best of the web].” IEEE Signal Processing Magazine 29.6 (2012): 141-142.

[13] Bock, Sebastian, Josef Goppold, and Martin Weiß. ”An improvement of the convergence proof of the ADAM-Optimizer.” arXiv preprint arXiv:1804.10587 (2018).

[14] VOHRADSKY, JIŘÍ. ”Neural network model of gene expression.” the FASEB journal 15.3 (2001): 846-854.