information

Article

Federated Blockchain Learning at the Edge

James Calo [1,*] and Benny Lo [2]

1 Department of Computing, The Hamyln Centre, Imperial College London, London SW7 2AZ, UK
2 Department of Surgery and Cancer, The Hamyln Centre, Imperial College London, London SW7 2AZ, UK; benny.lo@imperial.ac.uk
* Correspondence: jam414@ic.ac.uk

Abstract: Machine learning, particularly using neural networks, is now widely adopted in practice even with the IoT paradigm; however, training neural networks at the edge, on IoT devices, remains elusive, mainly due to computational requirements. Furthermore, effective training requires large quantities of data and privacy concerns restrict accessible data. Therefore, in this paper, we propose a method leveraging a blockchain and federated learning to train neural networks at the edge effectively bypassing these issues and providing additional benefits such as distributing training across multiple devices. Federated learning trains networks without storing any data and aggregates multiple networks, trained on unique data, forming a global network via a centralized server. By leveraging the decentralized nature of a blockchain, this centralized server is replaced by a P2P network, removing the need for a trusted centralized server and enabling the learning process to be distributed across participating devices. Our results show that networks trained in such a manner have negligible differences in accuracy compared to traditionally trained networks on IoT devices and are less prone to overfitting. We conclude that not only is this a viable alternative to traditional paradigms but is an improvement that contains a wealth of benefits in an ecosystem such as a hospital.

Keywords: IoT; machine learning; neural networks; federated learning; blockchain; learning on the edge

check for updates

Citation: Calo, J.; Lo, B. Federated Blockchain Learning at the Edge. *Information* **2023**, *14*, 318. https://doi.org/10.3390/info14060318

Academic Editors: Spyros Panagiotakis and Evangelos K. Markakis

Received: 7 April 2023
Revised: 18 May 2023
Accepted: 23 May 2023
Published: 30 May 2023

1. Introduction

The number of and use cases for IoT devices are increasing exponentially [1]; this is due in part to the increase in their ability to handle complex tasks that were once only possible on full-sized computers. One of the core use cases is within the medical field, referred to as the IoMT, with the market share of these devices on the rise. In 2017, the IoMT market was worth USD 28 billion and is projected to be USD 135 billion by 2025 [2]. Therefore, a solution that leverages this infrastructure while utilizing technologies proven to be effective is essential.

IoT architecture contains two main boundaries, cloud computing and edge computing, which differ primarily by the location where the computations take place and additionally differ in computing power. Additionally, there are three primary layers: cloud, fog and mist [3], as depicted in Figure 1. These layers are analogous to their namesakes, clouds are large and furthest away from the ground, fog is lighter and hovers between the ground and the clouds and mist is a thin layer of water molecules suspended just above the ground. The distance from the ground can be thought of as the distance from the users and the size of the water molecules are analogous to the computing resources at each layer.

Cloud computing is still the most prevalent [4,5] , allowing complex computations to be run on dedicated machines with the results streamed to IoT devices on demand, significantly reducing the need for IoT devices to perform complex calculations and instead act as a go-between for the client and server. This is effective but requires a connection to work; any disruption, due to congestion or network cutout, will cause a delay or even complete loss of results. This is unacceptable in many contexts such as during surgery or within a self-driving vehicle.

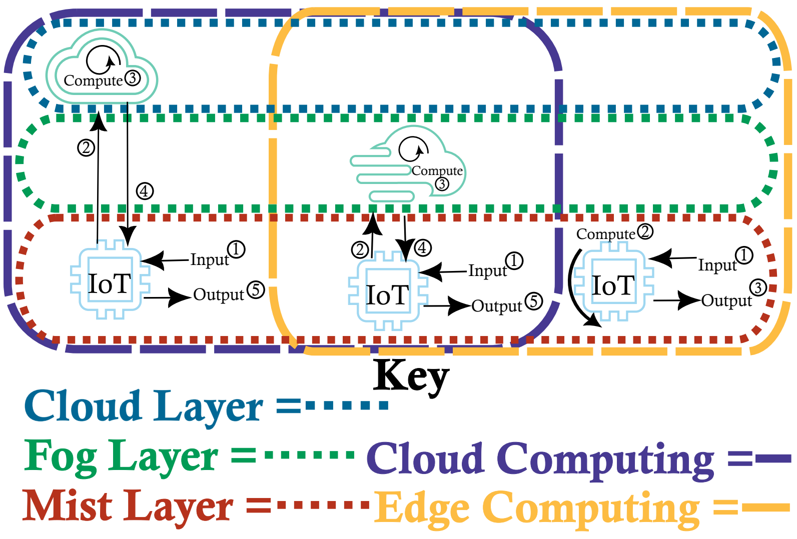


Figure 1. IoT Architecture Archetypes: Cloud and Fog both require an external server for handling the computations requested by the IoT device. The primary difference is the server's location, which is local for fog and external for cloud, and communication protocol. Mist, on the other hand, requires no external communication and all computations are performed on the device itself.

Fog computing [6,7] attempts to overcome these issues by exchanging the cloud server for a local server in the same location as the IoT devices; for example, a fog server may be housed in a hospital so that every internal IoMT device can connect directly to it as opposed to an external connection such as via the internet as is commonly the case in cloud computing. However, fog computing may still experience connection issues and requires space for dedicated hardware in close proximity. This is not ideal for situations where users either cannot be in close proximity or servers cannot be installed in the required location, such is the case with IoT sensors used in remote locations such as the Antarctic. Fog computing is a hybrid approach and is thus in the intersection of cloud computing and edge computing.

Therefore the most useful archetype is mist computing [8], where all computations are run on the IoT device itself or another (local) participant. Whilst this too requires a connection and is therefore prone to the same issues as fog computing, all participating devices are autonomous and can act individually as required. Unfortunately, this is also the most challenging archetype as the available resources are heavily limited and often require more specific solutions dependent upon the abilities of the individual device. However, whilst mist computing itself is completely within the edge computing boundary, it is frequently used in concert with the other layers. Therefore, when we refer to edge computing we refer to using only the mist layer, with each IoT device communicating only to each other without the requirement of dedicated (and more powerful) external servers.

Finally, there are additional paradigms that use software approaches to allow for additional benefits; dew computing, for example, caches and mirrors data on the cloud or fog in order to respond as if the data were local but requires syncing with the true servers. The most prevalent examples of dew computing are in cloud file storage where a local computer may contain cached copies of the files which are then synced with the cloud whenever changes are made or a refresh is requested.

Edge and fog computing, as a whole, have made great strides. Mobile Edge Computing (MEC), for example, allows for cloud computing capabilities with vastly reduced latency, high bandwidth and real-time access to network information [9]; therefore big data techniques can be applied with real-time feedback. However, this system does not actually leverage edge computing, instead using a fog computing architecture and as such must overcome issues such as reliance on users being in proximity to the fog servers, ensuring the connection does not interfere with the user's ability to access the internet (mobile data).

In order to overcome these issues, there are a small number of IoT systems that utilize true edge computing; however, these solutions either do not leverage machine learning at all or cannot be trained at the edge. For example, research by A.A. Abdellatif et al. developed an edge-based classifier for seizure detection and compressed sending of results [10,11]. Whilst this is a great step towards edge computing there is no machine learning being leveraged and so these types of systems are not able to use some of the recent advances in the field of machine learning for health. On the other hand, IoT devices using machine learning do exist and bringing machine learning, particularly neural networks and their collective family (convolutional, deep, etc.), onto the edge is a wide area of research [12]. TensorFlow has the TensorFlow Lite [13,14] framework which is designed to convert a trained neural network to run on edge devices; it is a very popular toolkit and is used in the book, TinyML [15] for running neural networks on embedded devices. However, whilst it can be used for on-device training, to our best knowledge, there are no systems that employ this.

Given the accelerated demand for faster training of more complex neural networks, there has been a focus on dedicated hardware such as GPUs and application-specific integrated circuits such as Google's Tensor Processing Unit (TPU) and field-programmable gate arrays (FPGAs), which are inherently cohesive with the matrix multiplication operations that underpin neural networks.

Whilst improvements to these specialized hardware components have accelerated our machine learning abilities, due to their high cost, large footprint and high energy draw, they are often missing from IoT devices, where smaller and less obtrusive devices are preferred. Even IoT devices that do contain specialized hardware, such as modern mobile phones which may have GPUs, require smaller, embedded, versions that are significantly less powerful than their full-sized counterparts. Furthermore, the less hardware required the lower the baseline battery draw is, a vital factor in IoT.

As a result, learning on a CPU on the edge is paramount! Removing dependency on specific hardware and providing all benefits via software allows existing devices to use this framework and keep the footprint of new devices small and focused on efficiency. Intel developed the Intel Xeon Scalable processor, which is geared towards learning and inference, focusing on three main features: the computation and memory capacity of the CPU, software optimizations for the CPU within machine learning tasks, specifically DNNs, and the use of distributed training algorithms for supervised deep learning workloads [16]. All three of these features translate to edge computing perfectly; for example, the memory capacity allows for better batching of input data and by not using a GPU there is no overhead between moving data from the CPU to GPU and back again. Furthermore, edge computing and IoT are inherently distributed and as such learning can be offloaded and computed at the source or destination, potentially increasing the efficiency of the whole system allowing each device to act autonomously and only interact with other devices when needed [7]. Furthermore, since training occurs upon observation of new data, which may not be consistent, inference, via the previous globally updated network, may be used simultaneously resulting in an online, continuous learning paradigm where networks are incrementally improved while running; this could be further improved by leveraging unsupervised techniques [17] or by using the inferring network as the supervisor.

Therefore, a privacy-ensuring, low-powered and generalizable method is required for training neural networks running at the edge. Federated learning allows multiple networks (of the same architecture) to train simultaneously on non-iid data without ever storing them and aggregate their knowledge into a global network, thereby learning at the edge. However, vanilla federated learning requires a centralized server for the aggregation; when privacy is involved, trust can be an issue. Therefore, we propose a hybrid approach utilizing the decentralization of a blockchain to create a framework that runs efficiently on multiple IoT devices across a P2P network, removing the need for a trusted centralized server, thereby improving privacy and adding robustness to data integrity. Furthermore, this enables the learning process to be distributed across participating devices (which federated learning does not implicitly do) leveraging the ubiquity of IoT devices to offset their lack of power by increasing the number of devices that are able to run in parallel.

1.1. Contributions

In this paper, we utilized our previous federated blockchain learning framework [18] to propose a pure mist solution to train a consortium of neural networks on a single Android phone to perform image classification on the CIFAR-10 dataset [19] Therefore, the main contributions of this work are as follows:

1. We developed a novel IoT federated learning framework, using Tensorflow Lite and our previously developed blockchain framework, to perform training at the edge (LotE) that is fully decentralized, leveraging our blockchain framework, ensuring that the data are private and secured against malicious attacks and requiring no trust between participants. This system requires no intermediary servers, which results in a mist-only architecture.
2. Using this framework, we build a configurable system for the training of neural networks on IoT devices and tested it on the CIFAR-10 dataset using a physical Pixel 4 Android smartphone running Android 13 with a Qualcomm Snapdragon 855 Octa-core CPU (1 $\times$ 2.84 GHz Kryo 485 Gold Prime, 3 $\times$ 2.42 GHz Kryo 485 Gold and 4 $\times$ 1.78 GHz Kryo 485 Silver) in order to obtain practical, and not simulated, results [19].
3. This system utilizes TensorFlow Lite as our machine learning framework (as opposed to our own framework's implementation) since standard TensorFlow is so widely used, is compatible with our existing federated blockchain learning framework and TensorFlow Lite converts TensorFlow models for use on IoT devices. Therefore, any existing system using Tensorflow will be able to receive the full benefit of our system with next to no overhead.

1.2. Related Works

Federated learning using blockchains has had an exponential surge in interest in recent years, especially for designing privacy enhancing and trustworthy AI, resulting in systems such as that by Yang et al. [20], which proposed a federated blockchain method using the PBFT consensus protocol. It can resist 33% of malicious users with a drastically lower energy cost than PoW but requires "Authorized" edge servers. Additionally, they use Multi-Krum as the federated aggregation scheme in order to achieve Byzantine-resilience. However, to the best of our knowledge, this proposed scheme has not been implemented on real hardware and utilizes cluster computing on the fog layer, requiring edge servers with significantly more power than standard IoT devices have to validate the blocks in the chain.

Islam et al. [21], on the other hand, utilizes drones as a method to ensure connection between devices running on the edge (pure mist computing) and utilizes differential privacy alongside federated blockchain learning to further enhance privacy. However, each entity in the system must register before being allowed to participate, which can cause issues if the system is required to grow and shrink dynamically.

1.3. Organization

The rest of this paper is organized as follows: Section 2 introduces federated learning and the blockchain as well as how we utilize these two different theories into a cohesive system. Section 3 analyzes the computational complexity and latency. Section 4 presents the results of our system running on a physical Android smartphone against the CIFAR-10 dataset in order to obtain results that demonstrate the real-world ability of our system. Finally, Section 5 discusses our findings and the direction we believe our future work will take.

2. Materials and Methods

2.1. Federated Learning

In the standard case, neural networks aim to approximate a function by minimizing the prediction loss with respect to the network's parameters:

$$\min_{\omega \in \mathbb{R}^d} \mathcal{T}(\omega) = \ell(\boldsymbol{x}, \boldsymbol{y}, \omega) \tag{1}$$

where ℓ is a chosen loss function, $\boldsymbol{x}, \boldsymbol{y}$ are the training (input and desired output) vectors and ω is the network's parameters.

However, with federated learning there are many networks training (potentially simultaneously) to form a global network, which may not have any training data of its own, and is an aggregation of these participating networks. Therefore, the *FedAvg* algorithm [22] (Equation (2)) is used to obtain the global weights for the global network, which may then be used as the initial network for the next round of federated learning.

$$\mathcal{F}(\omega) \triangleq \frac{1}{|\chi|} \sum_{i=1}^{N} |\chi_i| \cdot \mathcal{T}_i(\omega_i) \tag{2}$$

where for N participating networks, $|\chi_i|$ is the number of examples seen by network i (for $i \in \{1 \ldots N\}$), $\mathcal{T}_i(\omega_i)$ is the trained weights of network i as defined in Equation (1) and $|\chi| = \sum_{i=1}^{N} |\chi_i|$ is the total amount of data seen by all participants. Note that only the number of examples seen by each network is sent to the blockchain, no data are ever stored.

2.2. Decentralization with Blockchain

In order to replace vanilla federated learning's requirement for a centralized server we propose using a blockchain to tweak the paradigm to use a decentralized distributed ledger. This changes federated learning's architecture from the cloud/fog to the edge, where every device is independent and autonomous. The system will work even with only one node, and, in the event that all nodes go down, the system can recover fully since each node contains a copy of the accepted blockchain. This may not have been possible with a central server. The following details our design regarding the fundamental components of the blockchain.

2.2.1. Block

Our block format closely mirrors Bitcoin's format but with two major changes: the target formula and the federated components. The target is used to decide when the block has been mined via proof of work (PoW), which we chose over alternative, more green (both environmentally and chronologically) consensus mechanisms, such as proof of stake (PoS) [23–25]. Unfortunately, the downside of PoW is the energy cost; a miner who performs PoW must continually guess, in a deterministic manner, a hash that is less than the target, since the target is in big endian hexadecimal format we refer to it as a hash with more leading zeros than the target.

The criticism stems from the high computational cost of computing the hash which provides no computational benefit, other than the required effect of making it infeasible for a malicious node to pervert the system. However, as the blockchain is used to calculate the global update, the mining process provides some benefit, albeit tangentially. Furthermore, in an IoT system, this cost is somewhat negated as there are likely to be many interconnected devices and as such the problem can be split across them, much like mining pools.

Moreover, since the blockchain is primarily used as a trust mechanism for federated learning, the mining target difficulty can remain lower, reducing the computational cost; this additionally increases the rate of block addition to the chain which in turn reduces the time between each global network update. Therefore, lower-powered devices will have enough computing resources to generate hashes competitively, whilst still providing the same protection. As a result, we decided on a block rate of approximately one every 1.5 min. This rate is long enough for multiple local updates from different sources to be included in a block prior to the block being added to the chain, without being so long that the global update is outdated or a local device that misses out on the federated update grows stale.

PoS, on the other hand, would not be as suitable since it relies heavily on transactions, has no mining step, would give too much power to larger contributors, whose networks are likely to be less diverse. It also promotes coin hoarding, which negates the side benefit of blockchain rewards; this is what incentivizes institutions to utilize their spare computing power.

2.2.2. Mining

Mining of local updates via PoW requires storing the target in the block; however, the target size has the same number of bits as the hash. Therefore, much like bitcoin, we encode the target in 4 bytes:

$$\textbf{\textit{Target}} \overset{\textbf{\textit{hex}}}{=} 0\text{x}\overbrace{\phi_1\phi_2}^{\Phi}\overbrace{\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6}^{\Theta} \triangleq \Theta * 2^{8*(\Phi-4)} \tag{3}$$

where the first byte ($\phi_1\phi_2$) is an exponential scale and the lower three bytes contribute the linear scale. As with Bitcoin, we scale the exponential by 8, as there are 8 bits in a byte, simplifying the bit manipulation calculations. However, unlike Bitcoin, we scale the exponent by 4 (whereas Bitcoin scales by 3) in order to generate target values at the lower end of the spectrum since we use a lower block mining rate than Bitcoin.

2.2.3. Cryptography

We opted to use Keccak-256 for our cryptographic hashes instead of SHA256 and RIPEMD160, which Bitcoin uses. Keccak-256 is stronger compared to both and is used by Ethereum, which is a distributed state machine as opposed to a distributed ledger. Since we plan on adopting some of Ethereum's changes to the plain distributed ledger, such as smart contracts, keccak-256 is a more natural fit. However, we continue to use double hashing as Bitcoin does.

2.2.4. Networking

Using a peer-to-peer (P2P) network for our communication layer provides an essential benefit; whilst a single node will still be functional, the benefits of federated learning would be severely reduced. P2P networks are ideally suited to handle two vital situations: first, each node must be able to request a copy of the blockchain, including a list of addresses of other nodes which will receive the address of the connecting node and, secondly, the ability to broadcast information to all nodes accessing the chain, even those that may not be directly connected to this node.

By using a pair of UDP sockets (Algorithm 1), we can parallelize the communication and allow the communication to be distributed amongst different (local) devices. A hospital, for example, may have many IoMT devices but none with networking capabilities, just Bluetooth; they could therefore connect all IoMT devices to a single, network-capable IoT device, which would handle the networking and correct forwarding, much like Network Address Translation (NAT) with regards to WiFi routers. Consequently, any IoT device may participate, with the only requirement being a connection to a networking node, potentially via Bluetooth or even hardwired to a communication module, at some point downstream. Moreover, if a collection of IoT devices train as one unit, only one device needs to connect to the outbound UDP connection with all devices gaining the benefits.

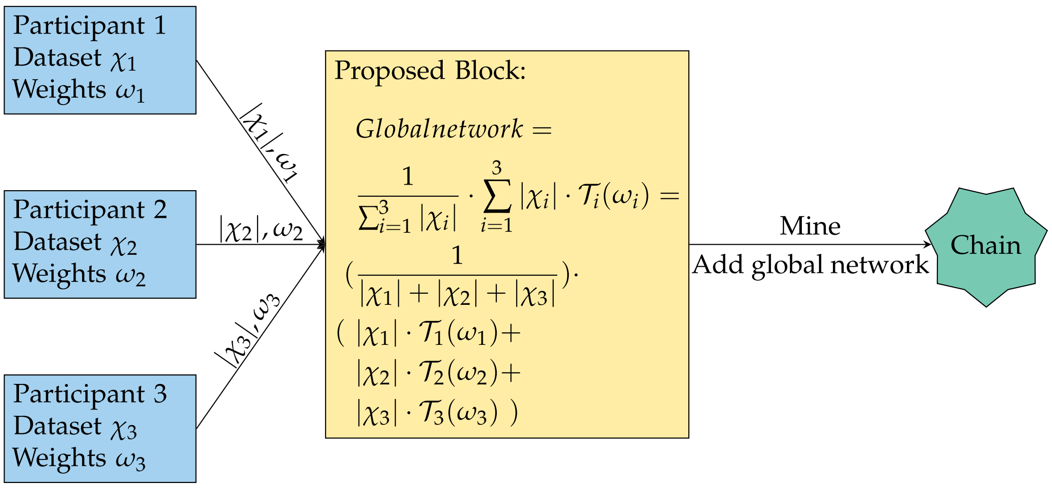
2.3. IoT Federated Learning

All devices contain an instance of a, potentially pre-trained, TensorFlow lite model and train on incoming data, private to each participant. After a number of epochs, each participant may submit their network's weights and the number of different examples seen to the blockchain (Figure 2). The chain then contains the globally aggregated model that can then be used as the new initial network for the next round of local training.

Algorithm 1 UDP Pair Communication.

```
procedure INBOUND
    loop
        $msg \leftarrow incomingMsg$
        if $ValidateIsNewestChain(msg)$ then
            $chain \leftarrow msg$
        else if $msg \notin addresses$ then
            $addresses.append(msg)$
            $Outbound(Inbound.Address, msg)$
        end if
    end loop
end procedure

procedure OUTBOUND($msg, addr = NULL$)
    if $msg == JoinNetwork$ then
        $Broadcast(Join + Inbound.Address)$
    else if $addr == NULL$ then
        $Broadcast(msg)$
    else
        $SendDirectMsg(msg, addr)$
    end if
end procedure
```



Participant 1
Dataset χ_1
Weights ω_1
Participant 2
Dataset χ_2
Weights ω_2
Participant 3
Dataset χ_3
Weights ω_3
$|\chi_1|, \omega_1$
$|\chi_2|, \omega_2$
$|\chi_3|, \omega_3$
Proposed Block:
$Globalnetwork = \frac{1}{\sum_{i=1}^{3}|\chi_i|} \cdot \sum_{i=1}^{3} |\chi_i| \cdot \mathcal{T}_i(\omega_i) = (\frac{1}{|\chi_1| + |\chi_2| + |\chi_3|}) \cdot (|\chi_1| \cdot \mathcal{T}_1(\omega_1) + |\chi_2| \cdot \mathcal{T}_2(\omega_2) + |\chi_3| \cdot \mathcal{T}_3(\omega_3))$
Mine
Add global network
Chain

Figure 2. Example of three clients contributing to the blockchain. Each participant trains a local network for a set amount of epochs over their observed dataset χ_i resulting in the network's weight set ω_i. The number of examples seen $|\chi_i|$ and the weight set are passed to the proposed block to be aggregated into the global model using Equation (2). Once the proposed block is mined, it is added to the chain with the new global network in the block; this may be used by anyone with access to the chain and used for the next iteration of local training. Note $(x \cup y \cup z) \subseteq all\_possible\_data$.

3. System Complexity

In this section, we analyze the complexity and latency of the proposed system, including actual timings taken from the system running in debug mode on a physical Pixel 4 Android smartphone running Android 13 with a Qualcomm Snapdragon 855 Octa-core CPU (1 × 2.84 GHz Kryo 485 Gold Prime, 3 × 2.42 GHz Kryo 485 Gold and 4 × 1.78 GHz Kryo 485 Silver) to give real, practical values to fairly evaluate the system.

The first step of the system is the local training; using ω to denote the number of CPU cycles to execute one step of training (i.e., applying backpropagation to one input) then the complexity of an epoch of local training is defined as

$$\Theta_{local\_training} = \max_{\mathcal{D}_k \in \mathcal{D}} \left(\frac{\beta_{\mathcal{D}_k} \cdot \omega}{\Omega_{\mathcal{D}_k}} \right) \tag{4}$$

where $\beta_{\mathcal{D}_k}$ is the batch size of the input data used in each epoch, $\mathcal{D}_k$ is a participating device (from the set of all devices $\mathcal{D}$) and $\Omega_{\mathcal{D}_k}$ is the CPU frequency of the participating device $\mathcal{D}_k$. In our experiments (averaging over 144 epochs) the latency of $\Theta_{local\_training} = 23.68$ s.

Each participant then sends their model to the blockchain for validation, which consists of the PoW consensus algorithm followed by the aggregation of all models into a global model (using Equation (2)). Therefore the complexity of the global aggregation is as follows:

$$\Theta_{global\_aggregation} = \frac{\rho + \omega_{agg}}{\Omega_{\mathcal{V}}} \tag{5}$$

where ρ is the number of CPU cycles to execute the PoW algorithm and $\omega_{agg} = (N+1) \cdot \Xi + (N-1) \cdot \sigma$ is the number of CPU cycles to execute the aggregation of N models (where Ξ denotes scalar matrix multiplication and σ denotes matrix addition) and $\Omega_{\mathcal{V}}$ is the CPU frequency of the validator. In our experiments, consisting of four models, the latency of $\Theta_{global\_aggregation} = 106.83$ milliseconds which is a negligible cost owing to the large target for the PoW algorithm. However, in general, $\rho >> \omega_{agg}$ unless there are a large number of participating devices.

Running the entire system on our single device (sequentially, with no concurrency or optimizations applied) yielded a complete run time of 19.79 min consisting of 4 models training for 50 epochs each with global aggregation occurring every 25 epochs (i.e., twice in the complete run).

4. Results

To test our system, we used a simple convolutional network comprising two convolutional and max pooling layers, a final convolutional layer and two dense layers, to classify the CIFAR-10 dataset. Ordinarily, the model would be first pre-trained using TensorFlow before being converted to the TensorFlow lite format. However, we wished to train almost entirely on-device and so only ran for one epoch (against 100% of the CIFAR-10 training data) so as to have a starting point over a network with completely randomized weights; although we also pre-trained another instance of the network with 100 epochs to compare the effects of pre-training. The results of non-federated learning on these two networks are shown in Table 1 with the loss and accuracy after training further on the device (against only 25% of CIFAR-10's training data) for 50, 100 and 150 epochs. These results imply that the pre-training caused the model to overfit and is likely only useful if the data the device observes will differ from the pre-training data.

Table 1. Loss and accuracy of a neural network against CIFAR-10 test data trained and evaluated on an Android phone. Each network was pre-trained on a laptop for the specific number of epochs on 100% of the CIFAR-10 training dataset and then trained further on the Android phone for the specified number of epochs against 25% of the CIFAR-10 training data.

Pre-Trained Epochs	On-Device Trained Epochs	Final Loss	Final Accuracy
1	50	1.43	48.04%
	100	1.65	49.43%
	150	3.08	47.97%
100	50	1.73	48.02%
	100	3.43	46.94%
	150	4.42	46.67%

Next, we trained multiple instances of the single epoch pre-trained network via federated learning. Globally updating all participating networks after either 25 or 50 epochs and training for 50, 100 and 150 epochs, we tested the federated setting for 2, 4 and 8 participating networks. All participants were trained on an even split of 25% of CIFAR-10's training data such that no 2 networks saw the same example (an example of the case with 8 participating networks is shown in Figure 3. The results in Table 2 show that the accuracy is very similar to the non-federated context with the main difference within each configuration being loss increasing with more training despite the accuracy remaining virtually constant.

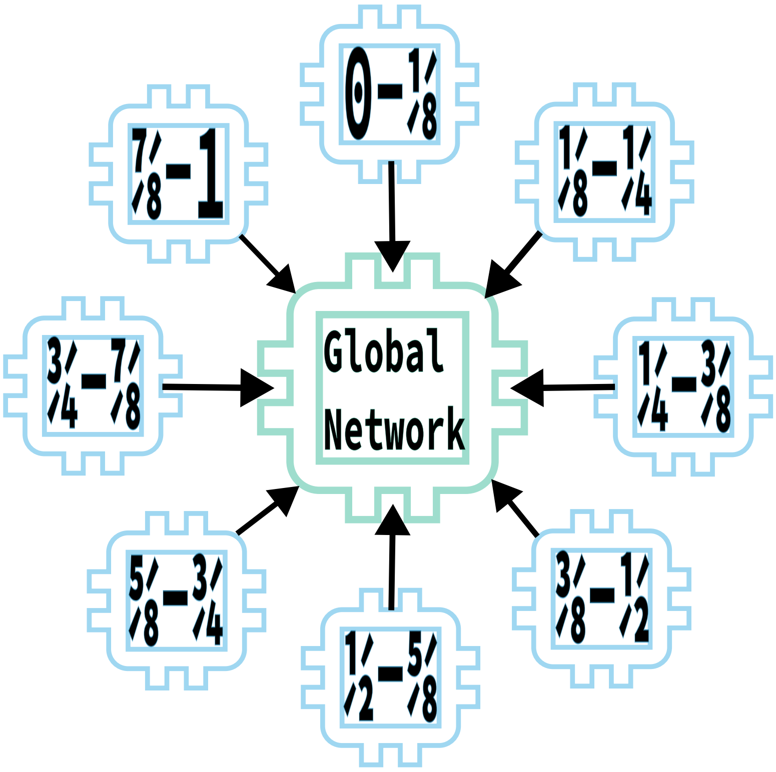


Figure 3. Federated Training of 8 models with an even split of the dataset. Given a dataset, χ, each local network, $i \in \{1 \ldots N\}$, trains on an even split of the dataset proportional to the number of networks such that for $\chi = \{\chi_1, \chi_2 \cdots \chi_k\}$, network i is trained on $\{\chi_{(i-1)k} \cdots \chi_{ik}\}$. These networks are then aggregated into a global network that performs as if it had been trained on χ despite not actually receiving any data, thereby preserving privacy.

Table 2. Loss and accuracy of a neural network against CIFAR-10 test data trained via federation and evaluated on an Android phone. Each participating network was trained on an even split of 25% of the CIFAR-10 training dataset with no participant seeing the same data.

Epochs per Global Update	Number of Participating Networks	Total Epochs	Final Loss	Final Accuracy
25	2	50	1.43	48.04%
		100	1.66	49.44%
		150	3.07	48.20%
50	2	50	1.43	48.03%
		100	1.66	49.19%
		150	3.05	47.93%

Table 2. *Cont.*

Epochs per Global Update	Number of Participating Networks	Total Epochs	Final Loss	Final Accuracy
25	4	50	1.43	48.04%
		100	1.64	49.62%
		150	3.06	48.12%
50	4	50	1.43	48.04%
		100	1.64	49.65%
		150	3.08	48.10%
25	8	50	1.43	48.04%
		100	1.65	49.46%
		150	3.08	48.28%
50	8	50	1.43	48.18%
		100	1.65	49.33%
		150	3.03	47.97%

5. Discussion

A new IoT-based, federated, decentralized and distributed learning approach is proposed with the aim of allowing learning on the edge by training multiple networks on multiple IoT devices and aggregating them into a global network that has seen no individual examples but is generalized for the specific task. We have produced a fully customizable Android application that allows on-device training of an arbitrary neural network via TensorFlow lite, allowing for existing networks to become federated and usable on an Android device in an extremely secure and privacy-enhancing manner.

However, there are a few components that we would like to address in future work: Smart contracts would be invaluable for automating tasks and sharing processing capabilities. Additionally, allowing the previous network iteration to supervise the next in a bootstrapping manner, forming a truly autonomous system. Finally, we wish to add homomorphic encryption to our system to further enhance the distributed manner of the system and to provide a novel, additional layer of privacy enhancement for both the machine learning models and the associated data.

Author Contributions: All authors contributed equally to this article. All authors have read and agreed to the published version of the manuscript.

Funding: Research was funded by the Engineering and Physical Sciences Research Council (EP/R513052/1).

Data Availability Statement: The data supporting this study's results can be found at https://github.com/SagaraBattousai/falcie/tree/0.1.0-AndroidResults (accessed on 3 April 2023) which is a git tag of the state of the project at the point it was used to generate our results.

Acknowledgments: We also acknowledge the invaluable inputs from Frank Po Wen Lo during this study.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
IoT	Internet of things
IoMT	Internet of medical things
iid	Independent and identical distribution
P2P	Peer-to-peer
PoW	Proof of Work

References

1. Ishaq, M.; Afzal, M.H.; Tahir, S.; Ullah, K. A Compact Study of Recent Trends of Challenges and Opportunities in Integrating Internet of Things (IoT) and Cloud Computing. In Proceedings of the 2021 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube), Quetta, Pakistan, 26–27 October 2021; pp. 1–4. [CrossRef]
2. Ghubaish, A.; Salman, T.; Zolanvari, M.; Unal, D.; Al-Ali, A.; Jain, R. Recent Advances in the Internet-of-Medical-Things (IoMT) Systems Security. *IEEE Internet Things J.* **2021**, *8*, 8707–8718. [CrossRef]
3. López Escobar, J.J.; Díaz Redondo, R.P.; Gil-Castiñeira, F. In-depth analysis and open challenges of Mist Computing. *J. Cloud Comput.* **2022**, *11*, 81. [CrossRef]
4. Chung, E.; Fowers, J.; Ovtcharov, K.; Papamichael, M.; Caulfield, A.; Massengill, T.; Liu, M.; Ghandi, M.; Lo, D.; Reinhardt, S.; et al. Serving DNNs in Real Time at Datacenter Scale with Project Brainwave. *IEEE Micro.* **2018**, *38*, 8–20. [CrossRef]
5. Fowers, J.; Ovtcharov, K.; Papamichael, M.; Massengill, T.; Liu, M.; Lo, D.; Alkalay, S.; Haselman, M.; Adams, L.; Ghandi, M.; et al. A Configurable Cloud-Scale DNN Processor for Real-Time AI. In Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, CA, USA, 1–6 June 2018.
6. Afroj, A.; Sahar, Q.; Naiyar, I.; Khalid, R. Fog, Edge and Pervasive Computing in Intelligent Internet of Things Driven Applications in Healthcare: Challenges, Limitations and Future Use. In *Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications*; IEEE: Piscataway, NY, USA, 2021; pp. 1–26. [CrossRef]
7. Giang, N.K.; Lea, R.; Blackstock, M.; Leung, V.C.M. Fog at the Edge: Experiences Building an Edge Computing Platform. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 9–16. [CrossRef]
8. Saeed, A.; Salim, F.D.; Ozcelebi, T.; Lukkien, J. Federated Self-Supervised Learning of Multisensor Representations for Embedded Intelligence. *IEEE Internet Things J.* **2021**, *8*, 1030–1040. [CrossRef]
9. Li, H.; Shou, G.; Hu, Y.; Guo, Z. Mobile Edge Computing: Progress and Challenges. In Proceedings of the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Oxford, UK, 29 March–1 April 2016; pp. 83–84. [CrossRef]
10. Abdellatif, A.A.; Mohamed, A.; Chiasserini, C. Automated class-based compression for real-time epileptic seizure detection. In Proceedings of the 2018 Wireless Telecommunications Symposium (WTS), Phoenix, AZ, USA, 17–20 April 2018; pp. 1–6. [CrossRef]
11. Emam, A.; Abdellatif, A.A.; Mohamed, A.; Harras, K.A. EdgeHealth: An Energy-Efficient Edge-based Remote mHealth Monitoring System. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–7. [CrossRef]
12. Chen, J.; Ran, X. Deep Learning with Edge Computing: A Review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
13. TensorFlow For Mobile & IoT Overview. Available online: https://www.tensorflow.org/lite (accessed on 29 December 2020).
14. Labrèche, G.; Evans, D.; Marszk, D.; Mladenov, T.; Shiradhonkar, V.; Soto, T.; Zelenevskiy, V. OPS-SAT Spacecraft Autonomy with TensorFlow Lite, Unsupervised Learning, and Online Machine Learning. In Proceedings of the 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 5–12 March 2022, pp. 1–17. [CrossRef]
15. Warden, P.; Situnayake, D. *TinyML*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2019; ISBN 978-1-492-05204-3
16. Rodriguez, A.; Li, W.; Dai, J.; Zhang, F.; Gong, J.; Yu, C. Intel Processors for Deep Learning Training. 2017. Available online: https://software.intel.com/content/www/us/en/develop/articles/intel-processors-for-deep-learning-training.html (accessed on 7 January 2021).
17. Hong, W.; Meng, J.; Yuan, J. Distributed Composite Quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, 2–7 February 2018; AAAI Press: Palo Alto, CA, USA, 2018; Volume 32. [CrossRef]
18. Calo, J.; Lo, B. IoT Federated Blockchain Learning at the Edge. *arXiv* **2023**, arXiv:2304.03006 .
19. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 25 February 2023).
20. Yang, Z.; Shi, Y.; Zhou, Y.; Wang, Z.; Yang, K. Trustworthy federated learning via blockchain. *IEEE Internet Things J.* **2022**, *10*, 92–109. [CrossRef]
21. Islam, A.; Amin, A.A.; Shin, S.Y. FBI: A Federated Learning-Based Blockchain-Embedded Data Accumulation Scheme Using Drones for Internet of Things. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 972–976. [CrossRef]
22. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, 20–22 April 2017; Volume 54.
23. Nair, P.R.; Dorai, D.R. Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain. In Proceedings of the 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 4–6 February 2021; pp. 279–283. [CrossRef]

24. Chicaiza, S.A.Y.; Chafla, C.N.S.; Álvarez, L.F.E.; Matute, P.F.I.; Rodriguez, R.D. Analysis of information security in the PoW (Proof of Work) and PoS (Proof of Stake) blockchain protocols as an alternative for handling confidential information in the public finance Ecuadorian sector. In Proceedings of the 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), Chaves, Portugal, 23–26 June 2021; pp. 1–5. [CrossRef]
25. Masseport, S.; Darties, B.; Giroudeau, R.; Lartigau, J. Proof of Experience: Empowering Proof of Work protocol with miner previous work. In Proceedings of the 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 28–30 September 2020; pp. 57–58. [CrossRef]