symmetry

MDPI

Article

Implementation Framework for a Blockchain-Based Federated Learning Model for Classification Problems

Zeba Mahmood * and Vacius Jusas

Department of Software, Kaunas University of Technology, 44251 Kaunas, Lithuania; vacius.jusas@ktu.It
* Correspondence: zeba.mahmood@ktu.edu

Abstract: This paper introduces a blockchain-based federated learning (FL) framework with incentives for participating nodes to enhance the accuracy of classification problems. Machine learning technology has been rapidly developed and changed from a global perspective for the past few years. The FL framework is based on the Ethereum blockchain and creates an autonomous ecosystem, where nodes compete to improve the accuracy of classification problems. With privacy being one of the biggest concerns, FL makes use of the blockchain-based approach to ensure privacy and security. Another important technology that underlies the FL framework is zero-knowledge proofs (ZKPs), which ensure that data uploaded to the network are accurate and private. Basically, ZKPs allow nodes to compete fairly by only submitting accurate models to the parameter server and get rewarded for that. We have conducted an analysis and found that ZKPs can help improve the accuracy of models submitted to the parameter server and facilitate the honest participation of all nodes in FL.

Keywords: decentralized ledger; federated learning (FL); artificial intelligence (AI); machine learning (ML); zero-knowledge proofs (ZKPs)

check for updates

Citation: Mahmood, Z.; Jusas, V. Implementation Framework for a Blockchain-Based Federated Learning Model for Classification Problems. *Symmetry* **2021**, *13*, 1116. https://doi.org/10.3390/sym13071116

Academic Editors: Kuo-Hui Yeh and Peng-Yeng Yin

Received: 23 March 2021
Accepted: 17 June 2021
Published: 23 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Different web services are increasingly being used to solve a myriad of problems in organizations all across the globe. Amidst all the use cases of the Internet, a debate about the roles of centralized and decentralized Internet architectures have arisen in recent years. While the centralized Internet serves organizations well, there are concerns about its future with regard to security and transparency. As a result, decentralized systems are slowly evolving to usher in a trustless economy with no centralized entities. Currently, artificial intelligence (AI) has evolved into complex networks combining deep learning (DL) methods where nodes are trained to handle complex problems that would otherwise remain unresolved with conventional technologies. DL has been associated with the successful implementation of systems such as pattern recognition, speech recognition [1], and protein analysis in cancer studies.

However, most DL models are still created along with the current architecture of the Internet staying highly centralized. Basically, the federated learning (FL) training process is based on one centralized server, which controls client management, global model maintenance, and gradient aggregation. During the training process in each round, the server transmits the current model to selected participating nodes. After obtaining the model, the nodes update all the local data and submit new updated gradients to the server [2]. These days, only a few pre-trained models such as VGG [3], ResNet [4], and GoogleNet [5] have been made available to the interested parties. The primary reason for such a few pre-trained DL models is the centralized architecture that these frameworks are built on. Consequently, this reduces the number of models present and incentives that users obtain from them. Being a decentralized ledger that provides transparency, immutable storage of data, and auditability, blockchain not only helps to create accuracies in DL models, but also ensures that honest nodes are incentivized in the form of tokens. Ideally, blockchain can enforce auditability, which is lacking in centralized frameworks and create

a completely trustless system where nodes compete honestly to enhance the precision of DL models. Kaggle [6] is considered a centralized framework that incentivizes its users to compete and improves the accuracies of both machine learning (ML) and DL models.

In this framework, a sponsor is required to submit a dataset and the corresponding ML or DL model to contestants. In turn, contestants compete to train and retrain the model in an attempt to enhance the accuracy of the model for which they are rewarded. To ensure that contestants do not overfit their models and cheat the system, Kaggle allows the contestants to only access test sets without retrieving labels. In this manner, both contestants and sponsors depend on Kaggle to act. Being centralized, the Kaggle framework is prone to hacks, and any contestant can attempt to compromise the server and obtain labels in advance. To eliminate the centralization challenges such as those of Kaggle, distributed computing has been provided. With distributed computing, the management of labels is no longer taken care of by a centralized server or node, but by all the nodes participating in the network. However, it still comes with an underlying problem of privacy. Currently, Resnet, VGG, RNN, Inception v3, and R-CNN are some promising pre-trained models [7]. FL, which integrates ML models, offers distributed learning. FL methods are used in three different types of systems, i.e., Horizontal federated learning, vertical federated learning, and federated transfer learning [8]. The FL architecture is usually based on the assumption that a node must train an ML model, with this training being partially committed in several other nodes. The key node, also known as the coordinator, gathers all qualified models and integrates them into a single model. The remaining nodes train the partial models on locally available data and send the trained models to the coordinator to be combined into a single final model [9]. These models are usually subject to continuous training via different external training sources. Some models such as Deep learning Networks Horizontal Federated Learning [10] are used, in which the model features are frequently altered but the model is constantly taught. FL centralizes power near the edge, where the edge determines whether to share training data with the cloud, updates DL models and communicates DL tasks across several networks, i.e., the Internet of things [11].

The FL framework was proposed by Google in 2016 [12]. For the first time, it introduces a comprehensive secure blockchain-based FL framework, and ML transforms areas as computer vision and speech recognition. It depends on the collection of data in privacy-invasive ways. FL comes as new sub-field of ML, which allows training models not to save and exchange data. It works as a training model for users to exchange weight updates to the server. In the nutshell, the algorithm sections, which are trained by users according to specific needs, are transferred to the users' computers. Instead of exchanging data, users send mostly specific designed compute model improvement to the server. This technique is known as more user-friendly, privacy-enabled and flexible. Mobile phone applications are one of the common examples where FL algorithms and training models works [13]. A huge amount of data are generated during the use of devices. This type of data is usually more privacy-protected, as all details are not exchangeable with the server. FL also allows for the training of a common model with all of the data, without losing computational resources or missing out on smart algorithms. Since more data are available, FL methods may also result in better models than traditional techniques. This study dwells on DL and blockchain technologies.

1.1. Distributed DL

For a DL algorithm to be considered accurate and reliable, vast amounts of training data are involved. Because of this problem, overly high computational power and resources are required [14]. However, many organizations are continuously retraining most of these complex DL models requiring large datasets to accomplish the same goals, which often waste computing power and resources. Unlike traditional centralized DL systems where datasets are submitted to a single server, FL distributes the computation process to the group of actors. An important aspect of DL is the activation that generates the results of a specific neuron. Activation is required to deduct important information regarding

non-linear features of the training data. In DL, the back-propagation process uses a concept known as stochastic gradient descent (SGD) to progressively compute the accuracy of the model while minimizing the overall model error at each subsequent training. Assuming that E_{total} is the model's overall error rate, V_{out} is the DL model output while V_{target} is the target value, then E_{total} can be calculated as follows:

$$E_{total} = \sum_{i=1}^{n} (V_{target(i)} - V_{out(i)})2. \quad (1)$$

When E_{total} is calculated, individual weights wj, i can now be updated as follows:

$$wj, i = wj, i - \vartheta * \partial(Etotal)/d(wj, i), \quad (2)$$

where ϑ is the learning rate, E_{total} is regarded as the comprehensive total with respect to weights which are denoted as i and j, respectively. In a typical DL model, training is repeated several times, until the goal is reached. In this method of learning, the training data are partitioned and stored in the distributed system in each of the nodes. The node trains the model on a local node and uploads the output to the server, which is then modified and redistributed to individual nodes. The model is retrained multiple times, until the number of errors gets smaller. The idea behind the system for distributed DL is outlined in Figure 1.

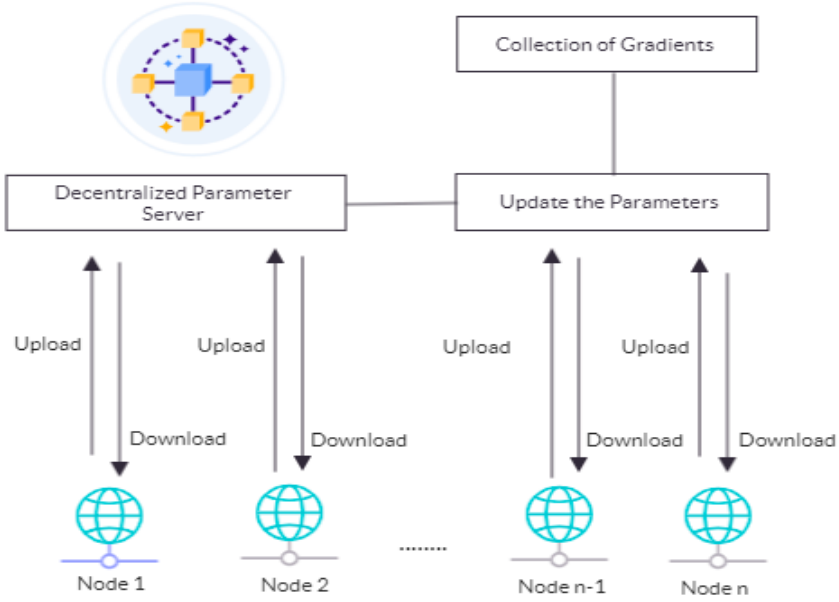

Figure 1. A distributed deep learning (DL) framework.

Our method relies on the parallelism technique, where multiple nodes contribute, through the SGD approach [15], to improving the accuracy of the model. The node stores its training model version while preserving the input of the model which forms a subset of all the datasets. As such, the nodes participating in the DL share similar model parameters by uploading and downloading parameters to and from the centralized server. The nodes are then needed to upload and update their training gradients via SGD. After upgrading a particular model, nodes will download the parameters from the server and repeat the training process, until the final trained model is obtained as shown in Figure 2.

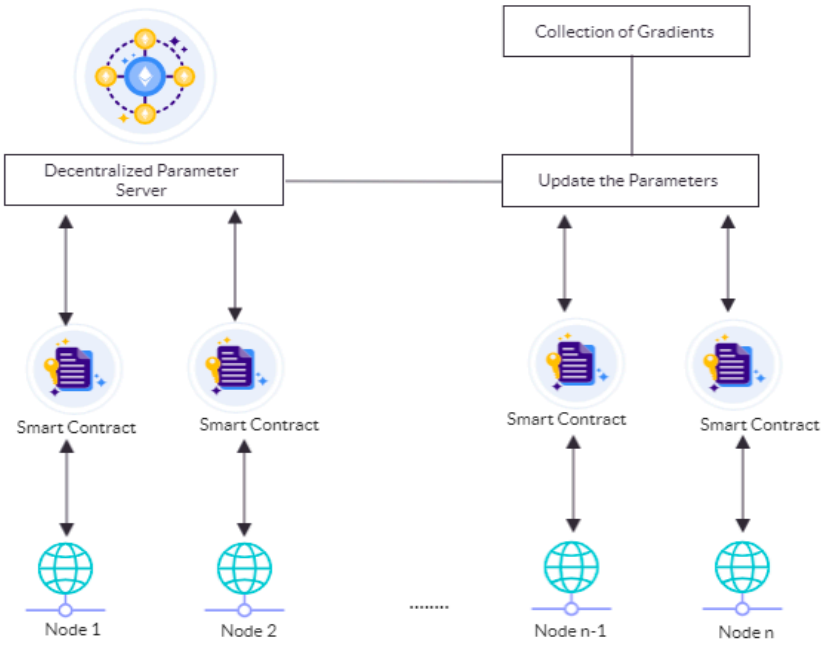

Figure 2. A blockchain-based federated learning (FL).

1.2. Evaluation in Blockchain

As a P2P [16] system, blockchain represents a trustless ecosystem that does not need trustworthy intermediaries. Within cryptocurrencies which in fact are one of the primary applications of blockchain, this technology has eliminated the need for centralized authorities, such as governments and banks, helping parties to interact seamlessly. Rather than relying on traditional accounting systems that are heavily centralized, blockchain has replaced intermediaries with consensus algorithms that ensure that transactions are recorded on a public ledger. Whenever parties interact, transactions are appended on a block by miners who compute consensus algorithms to verify that the transaction is valid. Every newly created block is added to all the blocks that are created before making up a chain that starts from the genesis block to the current block. Once a transaction is confirmed by the network, the information becomes available for all interested parties. Thus, on the website blockchain.com, the amount of bitcoins sent and received, addresses of both the receiver and the sender, the size of the fee paid to the network, the exact date and time when the transaction takes place, and much more details are contained. The key problem with this framework is scalability, as it can only process 19 transactions per second at the maximum. When more people try to send funds over the network in a given moment, transaction fees grow and so does the time need for transactions to get verified. Since then, many new frameworks trying to solve this problem have emerged. Among them are TRON [17], EOS [18], Zilliqa [19], Tezos [20], and Cosmos [21].

Blockchain can give incentives that permit community participation in FL. Besides incentives, blockchain supports traceability, transparency, and associated digital trust in FL models. This paper introduces a blockchain-based FL framework with incentives for nodes participation to enhance the accuracy of classification problems. The framework is based on Ethereum and creates an autonomous ecosystem where nodes compete to improve the accuracy of classification problems and are rewarded. Most blockchain implementations for FL have an underlying problem: privacy. Our framework uses ZKPs to make sure that data uploaded to the parameter server are accurate and private. The introduced framework allows nodes to compete fairly by only submitting accurate models to the parameter server for which they are rewarded.

Our contributions are shown as following:

- Introducing ZKPs as a means of creating a completely trustless FL environment where users' data are protected in a transparent algorithm.
- Creating a prototype of blockchain-based FL to assess the efficiency of a ZKP-based blockchain in terms of training accuracy, throughput, and time.

The rest of the paper is organized as follows. First, ZKPs and how they can ensure privacy in an FL environment are introduced. Later, we discuss the FL model and present our initial findings regarding training accuracy, throughput, and time.

2. Methodology

Zero-Knowledge Proofs

DL models are inefficient because of the scarcity of the training data. Blockchain offers an alternative solution to this problem, as it can create incentives within an FL framework for users to act truthfully. This solution has the potential to motivate many users to participate in the training and re-training of DL models. Besides incentives, blockchain supports the transparency, traceability, and auditability of transactions. This can create the necessary digital trust that is currently lacking in traditional DL systems. However, the blockchain solution is likely to create privacy challenges, since the ledger is public and data can be accessed by any interested party. If blockchain is incorporated in FL, then all the nodes in the blockchain will be in a position to access labels from test datasets since such datasets are posted on the public ledger.

Extensive research on ascertaining the realization of FL is underway by most analysts specialized in exploiting decentralized technologies. However, two issues have received attention in the majority of the researches, i.e., privacy challenges arising from a malicious parameter server and handling of personal information. In a typical distributed computing system, nodes upload their DL models to a centralized parameter server that later acts as a central link for all other modes. A malicious parameter server can act dishonestly and affect the accuracy of DL models that are submitted by peers within the network. For instance, a malicious server can deliberately drop gradients from some parties or incorrectly update the model's labels [22]. Therefore, a blockchain-based FL system should guarantee that gradients stay confidential while ensuring certainty with regard to the auditability of the model correctness. The second challenge refers to ensuring that users' data remain private. If parties attempt to collaborate, there is an underlying threat that some malicious users can obtain personal data from the parameter server. In some applications such as healthcare systems, privacy rules have made users reluctant to share their information due to malicious activities including inference attacks against HCUPnet.

Our model leverages ZKPs as the solution to the underlying problem of privacy in the use of public blockchains. A ZKP is a cryptographic system involving a prover (any participant in a cryptographic protocol) to authenticate and confirm with another entity called a verifier that it knows the validity of a given problem without revealing any data about the identity itself. Apart from ascertaining the validity of such a problem [23], ZKP must come with the following characteristics:

- Complete: If a given assertion is correct, then a verifier should be persuaded in it.
- Sound: If a given assertion is not correct, then no prover (whether malicious or not) should be proven.
- Zero-knowledge: If a given assertion is correct, then no verifier should infer any other new knowledge.

Consider a Turing machine. Let A, B, and C be some abstract Turing machines. A zero-knowledge proof with (A,B) for language L is regarded zero-knowledge, if for any time (T) with verifier (ŭ) there exists a simulator and $P(x)$ represents a prover transaction, and can be written as:

$$\forall x \in L,\ z \in \{0,1\}^x, View[P(x) \leftrightarrow u(x,z)]S(x,z), \tag{3}$$

where the record of all the transactions (interactions) between ŭ (x, z) and $P(x)$ shows the interaction of variables with transaction; the prover (P) can be created as an unlimited Turing machine, and $S(x, z)$ stands as an interaction value. This equation means that the whole interaction between the prover and the verifier is completely random and undistinguishable. Ethereum blockchain can be modeled as a prover, since it is a Turing-like computing framework. In the model, Z (a secondary string) provides a-priori information. Regarding privacy, P must return a decision on whether the DL model is correct or not without disclosing any data.

In a blockchain-based FL environment, zero-knowledge proofs allow provers to confirm that gradients obtained so far from their DL models are indeed accurate without conveying any other data about their identities. In other words, this data are concealed from the rest of the nodes. This solution provided by ZKPs can help address privacy challenges that usually arise with public blockchains in FL environments. A number of ZKPs have been suggested in other frameworks such as zk-SNARKs, zk-Starks [24], ZeroCash, and Bulletproof [25]. Our model adopts the zk-SNARKs that was first implemented by another privacy-oriented blockchain project ZCash. zk-SNARKs can be implemented in Ethereum smart contracts as a key component of enforcing privacy. As generic protocols, zk-SNARKs can verify any DL model within Ethereum smart contracts and generate Turing-complete Ethereum virtual machines (EVMs) that allow provers and verifiers to create any logic relevant to DL algorithms [26]. To make our model implemented in Ethereum, we create three smart contracts G, P, and V to incorporate zk-SNARKs and design them as follows:

- G (the key generator) accepts the hidden parameters m as well as DL model (DLmodel). G produces two keys with two arguments (a prover key provkey and a verification key verifkey). These keys are public entities that are created only once for each DLmodel.
- P (a prover): As inputs, it accepts a provkey, a public input m, and a private witness (w). In turn, it computes a mathematical proof in the form: proofzkp = P (provkey, m, w), for which the prover knows a source which can testify the DL model's correctness.
- V (verifier) computes V (verifkey, m, and proofzkp), which returns true if proofzkp is valid and inaccurate if it is not. V only returns true, if the proofzkp is aware of the existence of a specific witness w that satisfies the DL model (x, w).

Algorithm 1 was used to analyze the zk-SNARKs in the Ethereum blockchain and is specified below.

Algorithm 1 zk-SNARKs in Ethereum.

```
G ← G (m, DLmodel)
provkey ← G (m, DLmodel)
verifkey ← (m, DLmodel)
Prover ← (provkey, m, w)
Proofzkp = P(provkey, m, w)
Verifier ← (verifkey, m, proofzkp)
If w correct then
verifier (verifkey, m, proofzkp) = True
else
verifier (verifkey, m, proofzkp) = False
end if
```

3. System Architecture

In this section, we introduce the unique model underlying our system. In short, our framework implies that an FL framework uses incentive mechanisms provided by different blockchains to reward model provers and model verifiers in a transparent, secure, and auditable manner. FL ensures that data privacy and auditability are achieved using zero-knowledge proofs.

3.1. Overview of the System

In this system, each classification problem is managed by a separate blockchain with its own associated token. An add-only ledger is created to store all logs and records emanating from a specific DL model for a particular classification problem. For any problem, all the inputs and outputs that the DL model requires to solve are provided. For instance, a fashion-MNIST problem has its input *inMNIST* defined as a one-channel ($1 \times 28 \times 28$) pixels with values ranging from 0 to 1, while its output *outMNIST* is a 10-class label of which the values range from 1 to 10:

$$\left\{ inMNIST \in R^{\,1*28*28} \,\middle|\, 0 < inMNIST < 1 \right\}, \tag{4}$$

$$\{outMNIST \in Z | 0 < 1 < outMNIST < 10\}. \tag{5}$$

In this system, each problem is regarded as a set of test dataset tuples where each tuple $\{P$; NZKP; $Yn\}$ has $P = \sum_{i=1}^{m} Input$, NZKP is the non-zero knowledge proof function, while Yn is the output defined by $Yn = f(xn)$ on the true test label for a vector xn. Each party in the blockchain-based FL is associated with a public key (pub_{key}) and a private key (priv_{key}). Each actor has a role to play in the FL ecosystem as these roles include:

- DL model formulators: they define test datasets in the blockchain-based FL. These nodes are responsible for formulating problems to be solved.
- Model solvers: they compete to enhance/improve the accuracy of an FL model based on blockchain.
- Model validators: they use ZKPs to prove that a given model submitted by a model solver is accurate.

3.2. Building Blocks of the System

Our framework has two main parts: the underlying blockchain and the FL.

3.2.1. Enhancement of Blockchain Using ZKPs and Ethereum

Each blockchain consists of two parts:

- Problem blocks;
- Validation blocks.

Each problem block contains data about the block number, problem definition, test datasets, and a hash of the parent block. Each validator block contains the block number, parent block's hash, ZKP hash, and the block's hash. The blockchain starts with a problem block specifying how the model is improved, followed by a validator block which verifies the accuracy of the model. When a given problem is submitted to the FL environment, it is broadcasted to the entire set of nodes in a particular cooperating group within a given time threshold. During this period, any node within the cooperating group can add test datasets to the problem. When the time threshold is exceeded, the problem block is committed to the blockchain together with the block number, problem specification, test datasets, and hash of the parent block. All the data are contained in the new block. This triggers competition time where model provers compete to submit their enhancements/accuracies to the blockchain via ZKPs. Each user who is designated as a validator confirms the DL model and submits a proof as a vote to the system. For each ZKP, an associated unique verification hash is computed and tracked in the blockchain. At the end of the competition time, validator blocks are committed to the blockchain which at this time contains the highest number of unique ZKPs. This triggers the next competition cycle, as shown in Figure 3.

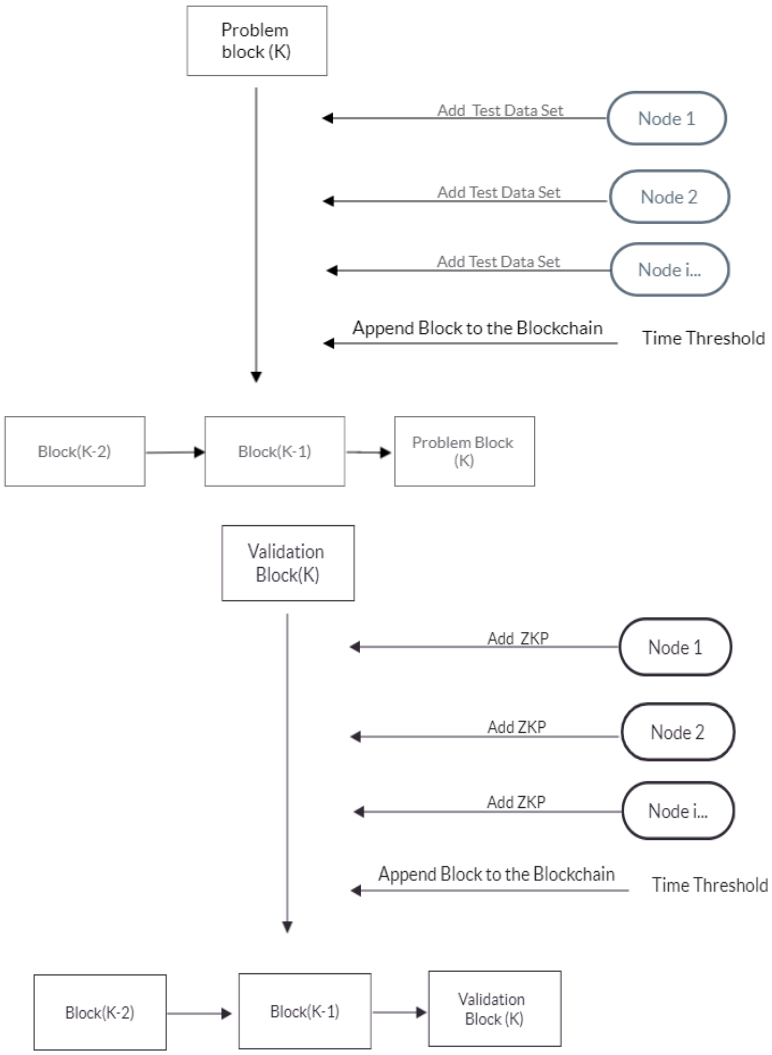

Figure 3. Blockchain validation and problem blocks.

Once a validator block gets committed, a reward is calculated in the form of tokens. An incentive mechanism is vital in FL for two key reasons. First, it motivates actors (problem formulators) who want to use an FL model but do not have sufficient training data to obtain optimal results on their own. Second, it ensures that actors behave honestly while improving FL models. In the essence of the framework, there exists an EVM which allows running numerous smart contracts with different types of business logic and confirming each contract on the blockchain [27]. In our system, each transaction is considered as a single hashed instruction created by an actor, so long as that actor has access to it and can contribute to the ledger. This system is implemented with two main smart contracts as following:

(1) Trading contract;
(2) Processing contract.

The trading contract allows nodes to repetitively trade their stochastic gradients on the blockchain. Once exchanged, the stochastic gradients are hashed with the right ZKPs by each node and added to the contract. This ensures both privacy and traceability. The trading contract is shown in Algorithm 2 as bellows.

Algorithm 2 Trading Contract.

```
receiveProblem ( );
CheckTimeOutput (t1);
UpdateTime ( );
verifyProblem ( );
CheckTimeOutput (t2);
UpdateTime ( );
commitProblem (t3);
UpdateTime ( );
commitProblem ( );
CheckTimeOutput (t3);
UpdateTime ( );
downloadCommitProblem ( );
CheckTimeOutput (t4);
UpdateTime ( );
trainModel ( );
CheckTimeOutput (t5);
UpdateTime ( );
Return ( );
CheckTimeOutput (t6);
Update time ( );
```

Algorithm 2 shows that the trading contract is created using six features (receiveProblem ( ), verifyProblem ( ), commitProblem ( ), downloadCommitProblem ( ), trainModel ( ), and Return ( )). In each of the functions, the script undertakes a time checkout process which determines the period of time. Each operation should take a certain amount of time. When a problem is received on the ledger, for example, the time checkout process ensures that actors in a cooperating unit can add their test datasets. When the time lapses, the problem is verified. Besides the trading contract, a processing contract is required to undertake the processes of parameter updating. During processing, designated nodes add up stochastic gradients and send the outputs that are confirmed via ZKPs. Once the ZKPs are calculated, the parameter server gets updated. The processing contract is created using three functions as shown in Algorithm 3.

Algorithm 3 Processing Contract.

```
UpdateTransaction ( );
CheckTimeOutput (t7);
UpdateTime ( );
verifyTransaction ( );
CheckTimeOutput (t8);
UpdateTime ( );
appendTransaction ( );
CheckTimeOutput (t9);
UpdateTime ( )
```

Algorithm 3 shows the process of processing contract. If after the ith iteration of FL training local stochastic gradients are uploaded to the network, nodes compete via ZKPs to run the update process, verification, and appending of transactions to the Ethereum blockchain as specified by Update Transaction ( ), Verify Transaction ( ), and Append Transaction ( ).

3.2.2. Implementation Details

The implementation of our system is done on the base of the following three key components:

(1) Ethereum blockchain;

(2) FL training algorithm;
(3) Incentive mechanism.

We implemented an Ethereum-based blockchain where users interact with the system via ERC-20 tokens. Next, we developed an FL environment using R (version 3.6.1) and the following libraries:

- ggplot2: an R package for visualizations;
- Keras: an R package for DL [28];
- TensorFlow: a free R library for dataflow and differentiable programming across many DL tasks [29];
- tidyr: an R package for data preprocessing.

Our framework uses the MNIST dataset [30]. We extracted 70,000 samples in 10 categories for our DL model and split them into training data (28,000 samples) and test data (14,000 samples). After that, we split both training and test data into five samples (5600 units for training data and 2800 for test data). The resolutions of images in the MNIST dataset were low (28×28 pixels in grayscale), which is important for proper segmentation. Next, we conducted multiple pieces of training on the training data while calculating the accuracy levels and uploading them to the parameter server. Our objective is to train a DL model that classifies different images of clothing in the fashion MNIST dataset. Once a validator block gets committed, a reward is computed in the form of ERC20 tokens. The identity of each node participating on the Ethereum blockchain is represented by its wallet address, which is created and managed by MetaMask, a plugin for accessing Ethereum blockchain on browsers such as Mozilla and Google Chrome. The position of the model validators is recorded and confirmed on the blockchain. Whenever a DL model verifier submits a vote, a position is automatically recorded on the ledger and used to calculate ERC20 tokens which are sent as a reward.

4. Findings

We discussed the DL model, the system's performance and capability to address privacy concerns in a distributed environment, and the viability of blockchain in FL.

4.1. DL Model

First, we built a DL model (neural network) using the fashion MNIST dataset in R. Since most DL learning models have chains of multiple layers, our DL model includes three layers that come with parameters to be learned during the training process. The first layer converts image format conversion from a two-dimensional array (28×28 pixels) to a one-dimensional array (28×28 pixels). The second layer has 128 neutrons or nodes, while the third layer is a 10-neutron layer which returns an array of 10 probability scores, with all summing up to 1. Each neutron in the DL network has a score that indicates whether the probability of the current image belongs to one of the 10 classification cases. We compiled the model using the following indicators:

- Loss function: the loss function measures the DL model's accuracy during training as a percentage.
- Optimizer: it provides insights on how DL is updated with regard to training data and the loss function.
- Metrics: metrics are used to check the training and testing stages.

The model is then trained and tested on the test data set. Its accuracy is then updated on the parameter server. Epoch levels adjustments are made with further tuning and enhancements to the DL model to generate more accurate results for the subsequent datasets. As the DL model progresses the training process, both accuracy and loss metrics are displayed. This illustrates the validation accuracy curve for our DL model. The accuracy improves with the number of epochs. Figure 4 illustrates the accuracy loss curve of the proposed DL model.

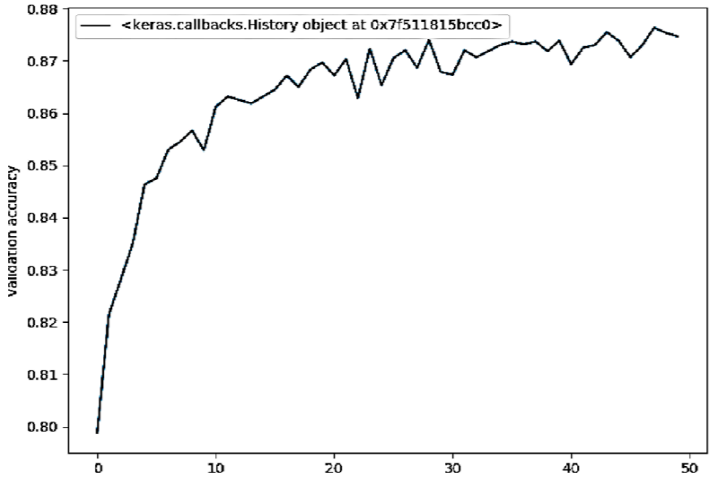

Figure 4. The accuracy loss curve in the DL model.

As shown in Figure 4, our DL model's accuracy levels oscillate from 0.80 (80%) to approximately 0.88 (88%). Each node participating in the training process is identified by two keys: public key and associated private key. When the training process is concluded, a node that has just trained the DL model encrypts its computation results, which include the accuracy of the DL model by using the parameter server's public. When the parameter server obtains the information, it uses its private key to retrieve the contents of the data from the node. To verify that a particular node enhances the DL model's accuracy, the parameter server uses a federated averaging algorithm [31]. Under this framework, local nodes are allowed to conduct more than one batch of updates on their training data instead of the gradients. The idea behind the federated averaging algorithm is to ensure that weights emanating from the same node do not alter the model's accuracy. Algorithm 4 specifies the federated averaging algorithm concept.

Algorithm 4 Federated Average.

```
ServerUpdate ( );
Initalize Wo;
for count ← 0 to do
    Wait for the update from N nodes to calculate the current average accuracy;
Current Accuracy Score = Client Update ( );
    ωt = Summation of Weighted updates;
Client Update ( );
α← split data into ƙ mini-batches;
Π ← |α|; // Update the weight;
ω init ← ω;
for batch (n) ϵ α do
    Compute the new weights;
Return the weights to server;
```

4.2. Ethereum Blockchain

When a node submits its current weights, the parameter server determines whether the submitted score is higher than the current one on the ledger. If it is higher, it broadcasts the information to all validators who in turn use ZKPs to verify that the submitted score is indeed true. If the validators verify that the weight is correct, a block is committed and a reward in the form of Ether (ETH) is computed. ETH is the native currency (coin) that supports various transactions on the Ethereum blockchain. Since the identity of each node participating on the Ethereum blockchain is known via the wallet address, its position is recorded and confirmed on the blockchain and rewarded with ETH. Whenever a DL model

verifier submits a vote (enforced using smart contract), his/her position is automatically recorded on the blockchain and used to compute the tokens which will be received. We assessed the feasibility of FL model training in our framework in a distributed environment based on two metrics: throughput and how it addresses privacy issues. We used a Laptop PC having Core i7 2.2 GHz with Radeon RX Vega and 4 GB RAM. We based our final score for each metric after averaging five trails. We evaluated our systems with respect to throughput and training accuracy. In terms of throughput, we found that cipher size remains largely constant for multiple federated averages, as shown in Figure 5.

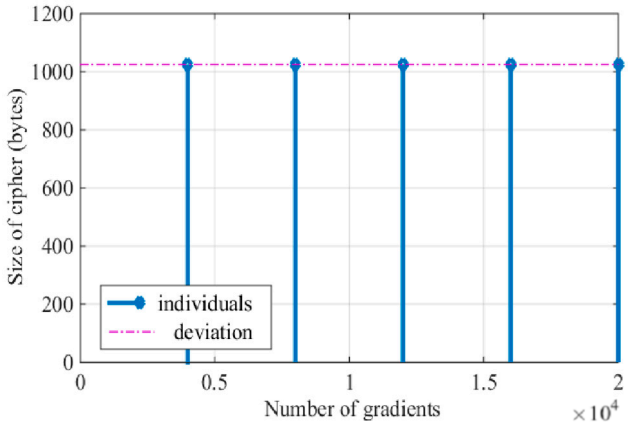

Figure 5. Size of cipher as the number of gradients increases.

As evident in Figure 5, cipher size remains largely constant (ranging from 0 to about 1000 bytes), as the number of gradients increase. On the other hand, the system's throughput progressively decreases, as the number of gradients increases, as shown in Figure 6.

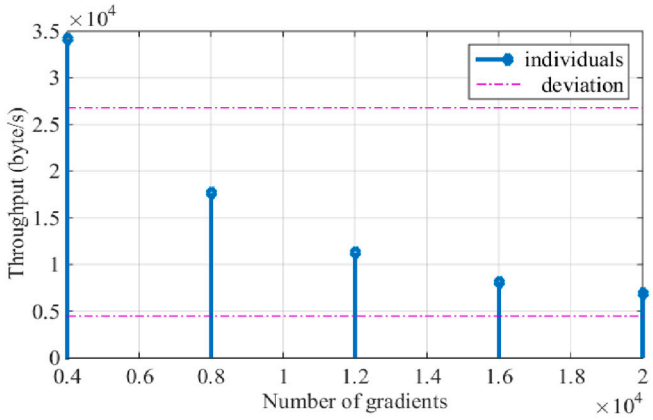

Figure 6. System's throughput versus the number of gradients.

The decline in the system's throughput can be attributed to Ethereum's performance challenges, which result from increased applications running on Ethereum blockchain [32]. At present, Ethereum can process an average of 7 transactions per second (tps) to 19 tps, which is quite low in terms of handling the increasing number of applications. If the number of users trying to conduct transactions exceeds this number in a given moment, those who offer a free higher network have better chances to get their transactions confirmed. Thus, transaction fees may get very high, when the network gets congested. This scalability problem is acknowledged, and some solutions have already been proposed to address it. One of such solutions that Ethereum developers implement is Sharding [33]. This technology implies dividing networks into many small pieces or shards. In order for a transaction to be verified, only the confirmation of a few shards, instead of all nodes, will be required, which would significantly reduce the system's overload and help Ethereum re-

solve the problem of scalability. This technology is expected on the 4th stage of Ethereum's development, also known as serenity.

With regard to training accuracy, we found that as the number of nodes participating in the training process increase, the model's accuracy also increases. We then trained the model five times by increasing the number of users at each training (i.e., the first training had one user and used the first test sample, the second training had two users and used the second test sample, and so on). With the help of different test datasets, ensured randomness is achieved. By sharing gradients on the Ethereum blockchain, each node is in a position to obtain updated averages that are contributed by other nodes. Because each node uses the previous score stored on the ledger as the baseline, which contribute to the overall accuracy of the model, only federated averages that increase the previous ones are committed and append to the block.

To train the model on the server, decentralization is the main source used. In this method, training each client and model every time can cause time delay. FL minimizes this problem, as each model has its local data to use according to the required degree of predication. Clients will request to retrain the model using local data which are available on the server with new weights. Then, the server will collect the new model, which is given by the clients and aggregate them into one model. This process is more adaptive and precise in terms of the privacy and security of the client's data as compared to the model proposed by Eugene et al. [34]. The CIFAR-10 dataset is used to check the security of the proposed system, which consists of 60,000 32 $\times$ 32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images in this dataset, which is further divided into five training batches and one test batch, with each having 10,000 images. The test batch contains exactly 1000 randomly selected images from each class. The remaining images will be trained later in random order. The batches contain the remaining images in random order, which may contain more images from one class than another. The model is trained using the client-side data on the server, and then, it sends the output to the server with new weights and gradients.

By enhancing our framework with Ethereum, we inherit not only all the decentralization and security features, but also limitations of the network. Since Ethereum is a public blockchain, privacy challenges are likely to arise. To address this challenge, we used zero-knowledge proofs. With ZKPs, any data including weighted averages and training datasets can be uploaded to the parameter server and shared privately. At the same time, validators can verify that a certain DL model is accurate without revealing the private details of the prover. With ZKPs, our framework allows peers to submit their models without using true test labels. Consider a sybil attack on the blockchain where a malicious node can create multiple wallet addresses (accounts) and create multiple proofs to fool the network into believing that the transaction is valid. It can be conducted on a PoW-based system, if an attacker owns various identities with sufficient computational power to have substantial influence in the system. For an attack to be effective, the amount of computing power owned should be significant, but it would still cost the same if it is centralized in one entity or split among several ones. Similarly, in PoS schemes, the attacker can have access to the same voting power, if one identity owns all the attacker's stake or if it is divided into several identities [35]. ZKPs have been proven to be a good solution to this problem, since the verifier must compute a verification hash which can only return true if the proof is correct. If the proof is incorrect, the verification hash will be false, which in this case means that one node uses multiple accounts. This is enforced in a processing contract already mentioned. Again, the cost of gas (fee charged on Ethereum users to validate transactions) is too high, which prohibits such behaviors on the network.

5. Conclusions

In recent times, attempts have been made to introduce blockchain in AI, in particular in DL. Our solution has several advantages compared to the state-of-the-art FL models. In this paper, we have introduced an FL framework that uses Ethereum blockchain. We have also

introduced ZKPs, particularly the zk-SNARKs, and demonstrated that they can be used to achieve the required privacy that is currently lacking in public blockchains. Figure 5 shows an evaluation of the system's throughput, which reveals improved accuracies for the DL models despite the Ethereum's scaling challenges. Our model provides a practical chance for further research on the role of blockchain in AI and ML with the following advantages:

- The integration of the blockchain ensures the transparency and integrity of the computation: once data is logged, no one can modify them due to the PoW mechanism and the consensus protocol.
- ZKPs keep sensitive data from being disclosed to non-authorized parties.
- ZKPs integration with trading contract ensures both privacy and traceability.

Further research is required to establish which blockchain framework (NEM, Hyperledger Fabric, or Corda) is best suited for DL applications.

Author Contributions: Conceptualization, Z.M.; methodology, Z.M.; software, Z.M.; validation, Z.M., V.J.; formal analysis, Z.M.; investigation, Z.M.; resources, Z.M.; data curation, Z.M.; writing—original draft preparation, Z.M.; writing—review and editing, Z.M., V.J.; visualization, Z.M.; supervision, V.J.; project administration, V.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Livezey, J.A.; Chang, E.F.; Bouchard, K.E. Deep learning as a tool for neural data analysis: Speech classification and cross-frequency coupling in human sensorimotor cortex. *PLoS Comput. Biol.* **2018**, *15*, e1007091. [CrossRef] [PubMed]
2. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S. 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.
3. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
4. Wu, Z.; Shen, C.; van den hengel, A. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognit.* **2019**, *90*, 119–133. [CrossRef]
5. Al-Qizwini, M.; Barjasteh, I.; Al-Qassab, H.; Radha, H. Deep learning algorithm for autonomous driving using googlenet. In Proceedings of the IEEE Intelligent Vehicles Symposium, Redondo Beach, CA, USA, 11–14 June 2017.
6. Roelofs, R.; Shankar, V.; Recht, B.; Fridovich-Keil, S.; Hardt, M.; Miller, J.; Schmid, L. A meta-analysis of overfitting in machine Learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 9175–9185.
7. Kandarpa, V.S.; Sunda r Kandarpa, B.; Lokeshwar, B.; Anil Baruah Kumar, P.; Sankara, S. Evaluating training time of inception-v3 and resnet-50,101 models using tensorflow across CPU and GPU. In Proceedings of the 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 29–31 March 2018.
8. Süzen, A.A.; Şimşek, M.A. A novel approach to machine learning application to protection privacy data in Healthcare: Federated Learning. *Namık Kemal Tip Dergisi* **2020**, *8*, 22–30.
9. Ilias, C.; Georgios, S. Machine learning for all: A more robust federated learning framework. In Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019), Prague, Czech Republic, 23–25 February 2019.
10. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
11. Elnagar, S.; Thomas, M.A. Federated deep learning: A conceptual model and applied framework for industry 4.0. In Proceedings of the AMCIS, Virtual, 15–17 August 2020.
12. Niknam, S.; Dhillon, H.S.; Reed, J.H. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Commun. Mag.* **2020**, *58*, 46–51. [CrossRef]
13. Liu, F.; Wu, X.; Ge, S.; Fan, W.; Zou, Y. Federated Learning for Vision-and-Language Grounding Problems. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11572–11579.
14. Jeffrey, D.; Greg, S.C.; Rajat, M.; Kai, C.; Matthieu, D.; Quoc, V.L.; Mark, Z.M.; Marc', R.A.; Andrew, S.; Paul, T.; et al. Large scale distributed deep Networks. In Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012), Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1223–1231.
15. Gao, H.; Xu, A.; Huang, H. On the convergence of communication-efficient local SGD for federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 18–19 May 2021.

16. Bahri, L.; Girdzijauskas, S. Blockchain technology: Practical P2P computing. In Proceedings of the IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W), Umea, Sweden, 16–20 June 2019.
17. Li, H.; Li, Z.; Tian, N. Resource bottleneck analysis of the blockchain based on tron's tps. In Proceedings of the International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Kunming, China, 20–22 July 2019; Springer: Berlin/Heidelberg, Germany, 2019; Volume 1075.
18. Rahman, M.U. Scalable role-based access control using the EOS blockchain. *Cornell Univ. Cryptogr. Secur.* **2020**, *2007*, 02163.
19. Aydinli, K. Design and Implementation of a Scalable IoT-based Blockchain. Master Thesis, CSG@ IfI, University of Zürich, Zürich, Switzerland, 2019.
20. Allombert, V.; Bourgoin, M.; Tesson, J. Introduction to the tezos blockchain. In Proceedings of the 2019 International Conference on High Performance Computing & Simulation (HPCS), Dublin, Ireland, 15–19 July 2019.
21. Bernardo, B.; Cauderlier, R.; Pesin, B.; Tesson, J. Albert, an intermediate smart-contract language for the Tezos blockchain. In Proceedings of the International Conference on Financial Cryptography and Data Security, Kota Kinabalu, Malaysia, 11–14 February 2020.
22. Zhu, L.; Han, S. Deep Leakage from Gradients. *Mach. Learn.* **2019**, *2*, 17–31.
23. Wang, H.; Zhixin, S. Research on zero-knowledge proof protocol. *IJCSI Int. J. Comput. Sci. Issues* **2013**, *10*, 194.
24. Panait, A.; Olimid, R.F. On using zk-SNARKs and zk-STARKs in blockchain-based identity management. In *Innovative Security Solutions for Information Technology and Communications*; SecITC; Springer: Cham, Germany, 2020.
25. Andrey, J. *A Concrete Instantiation of Bulletproof Zero-Knowledge Proof*; IACR Cryptol: Lyon, France, 2019.
26. Li, X.; Zheng, Y.; Xia, K.; Sun, T.; Beyler, J. *Phantom: An Efficient Privacy Protocol Using zk-SNARKs Based on Smart Contracts*; IACR Cryptol: Lyon, France, 2020; p. 156.
27. Ahrent, W.; Bubel, R.; Ellul, J.; Pace, R.; Pardo, R.; Rebiscoul, V.; Schneider, G. Verification of smart contract business logic. In Proceedings of the International Conference on Fundamentals of Software Engineering, Tehran, Iran, 1–3 May 2019.
28. Arnold, T.B. kerasR: R Interface to the keras deep learning library. *J. Open Source Softw.* **2017**, *2*, 296. [CrossRef]
29. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Dean, J.; Devin, M.; Ghemanwat, S.; Irving, G.; Isard, M.; Kudlur, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), Savannah, GA, USA, 2–4 November 2016.
30. MNIST-Dataset, "Github". Available online: https://github.com/yeshijinsel/MNISTdataset (accessed on 4 December 2020).
31. Im, W.L.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.C.; Yang, Q.; Miao, C. Federated learning in mobile edge Networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2031–2063.
32. Oliva, G.A.; Hassan, A.E.; Jiang, Z.M. An exploratory study of smart contracts in the Ethereum blockchain platform. *Empir Software Eng* **2020**, *25*, 1864–1904. [CrossRef]
33. Yu, G.; Wang, X.; Yu, K.; Ni, W.; Zhang, J.A.; Liu, R.P. Survey: Sharding in blockchains. *IEEE Access* **2020**, *8*, 14155–14181. [CrossRef]
34. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to backdoor federated learning. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Virtual, 26–28 August 2020.
35. Deirmentzoglou, E.; Papakyriakopoulos, G.; Patsakis, C. A survey on long-range attacks for proof of stake protocol. *IEEE Access* **2019**, *7*, 28712–28725. [CrossRef]