Computer Networks 201 (2021) 108594

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Proof of Learning (PoLe): Empowering neural network training with consensus building on blockchains

Yuan Liu [a,*], Yixiao Lan [a], Boyang Li [b], Chunyan Miao [b], Zhihong Tian [c]

[a] Software College, Northeastern University, China
[b] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[c] Cyberspace Institute of Advanced Technology, Guangzhou University, China

ARTICLE INFO

Keywords:
Consensus mechanism
Proof of Learning
Secure mapping layer

ABSTRACT

The advent of neural network (NN) based deep learning, especially the recent development of the automatic design of networks, has brought unprecedented performance gains at heavy computational cost. On the other hand, in order to generate a new consensus block, Proof of Work (PoW) based blockchain systems routinely perform a huge amount of computation that does not achieve practical purposes but to solving a difficult cryptographic hash puzzle problem.In this study, we propose a new consensus mechanism, Proof of Learning (PoLe), which directs the computation spent for block consensus toward optimization of neural networks. In our design, the training and testing data are released to the entire blockchain network and the consensus nodes train NN models on the data, which serves as the proof of learning. As a core component of PoLe, we design a secure mapping layer (SML) to prevent consensus nodes from cheating, which can be straightforwardly implemented as a linear NN layer. When the consensus on the blockchain network is achieved, a new block is appended to the blockchain. We experimentally compare the PoLe protocol with Proof of Work (PoW) and show that PoLe can achieve a more stable block generation rate, which leads to more efficient transaction processing. Experimental evaluation also shows the PoLe can achieve a stable block generation rate without significantly sacrificing training performance.

1. Introduction

Pioneered by the Bitcoin system [1], blockchain has become the underlying technology of crytocurrency and smart contracts based applications. In this study, we refer to a blockchain network as a generalized distributed ledger or database governed by a consensus algorithm, where participants do not trust each other but are able to collectively maintain a consistent ledger [2]. In a Proof of Work (PoW) based blockchain network, miners compete to be the first to generate the next valid block and earn a reward for their work to solve the cryptographic puzzle. The consensus algorithm incorporating proper incentive distribution design is the soul of a blockchain network system, contributing to its attractive characteristics like decentralization, transparency, auditability, security and temper-resistance [3]. The most successful blockchain projects, e.g. Bitcoin and Ethereum, adopt the proof of work (PoW) consensus mechanism [4,5], where each consensus node distributively and individually solve an asymmetric cryptographic puzzle problem that is difficult to solve but easy to verify. PoW based blockchain network has attracted massive computational resources. According to the Cambridge Bitcoin Electricity Consumption Index (https://www.cbeci.org/), at the time of this writing (October 3, 2021), the annualized electricity consumption of Bitcoin mining (101.68 TWh) exceeds that of Austria (about 66 TWh). Most of the energy is spent solving the cryptographic puzzle problem. Beyond providing trustless consensus, it serves no useful purpose.

Recently, considerable performance gains in deep learning have been achieved by scaling up the network and training data [6–9] and from the automatic design of neural network (NN) architectures [10–12]. These trends have created an ever-growing demand for compute power. For example, the largest GPT-2 model has a total of 1542 million parameters [7]. The evolution of the transformer model from [10] required 32,623 TPU hours or 2,192,960 P100 GPU hours. Such computational requirements render model optimization unrealistic for academic researchers and even for small-sized enterprises, and hinder the wide adoption of artificial intelligence technologies.

Therefore, in this study, we aim to direct the computation and energy spent on blockchain consensus to the practical function of training NN models. Several pioneering studies have investigated in proposing new consensus schemes based on training machine learning

[*] Corresponding author.
E-mail address: liuyuan@swc.neu.edu.cn (Y. Liu).

https://doi.org/10.1016/j.comnet.2021.108594
Received 5 February 2021; Received in revised form 20 October 2021; Accepted 1 November 2021
Available online 9 November 2021

model [13–15] since 2019. Based on comparative analysis of these studies, we contend that a practical consensus mechanism for training NN models should satisfy the following three challenging properties: (1) **customized NN task** — the main computation power of a consensus node should be utilized in training customized neural network models; (2) **no pre-training** — the training task cannot be pre-prepared and no consensus node can start training the model in advance; (3) **trained model anti-theft** — the trained model should be identifiable and serve as work proof that no other consensus node can claim a model that is not trained by itself. Unfortunately, these existing consensus mechanisms for model training fail to satisfy all the above three properties. The proposed consensus mechanism in this study aims to fill this research gap.

Specifically, we bridge the compute resource of a blockchain network to train neural network tasks ensuring no pre-training and trained model anti-theft properties. Similar to PoW, the proposed Proof-of-Learning (PoLe) consensus mechanism contains two main constituents: an asymmetric puzzle and a tampering prevention technique that links adjacent blocks. For the asymmetric puzzle, we rely on the non-convex optimization of large neural networks, based on the realization that it is NP-hard to find a set of network weights that achieve a given training accuracy, but easy to verify if a given set of weights can achieve a given training or test accuracy. The no pre-training and trained model anti-theft properties are achieved by the design of the secure mapping layer (SML). The SML is a linear neural layer to connect encrypted training data with a conventional neural network task, which is generated based on the previous and current block hash. Thus, each training task can only be started with the previous block confirmed among the network, and no one can train a task in advance. Moreover, since the current block generated by each node is different (i.e. the coinbase transaction identifies the block rewards for the block generator), the SML of each trained model can uniquely identify the node who trained the model and no other node can steal a trained model. In other words, our SML associates the immutable block link and consensus node identity in the trained model to prove its work. Furthermore, in order to encourage miners to contribute valid blocks, we propose a reward distribution approach where ommer blocks (the valid blocks not regarded as the winning block) are also properly rewarded.

The main contributions of this work are summarized as follows.

- We propose a new blockchain consensus algorithm called Proof of Learning (PoLe), which channels the otherwise wasted compute power to the practical purpose of training neural network models.
- In order to maintain the no-pre-training and trained model anti-theft properties, we propose the design of a secure mapping layer SML that uses a linear NN to link adjacent blocks together and identify the miner identity. Empirical evidence in Experiment 2 and 3 shows that the encryption detects tampering behavior without significantly degrading predictive performance.
- We provide a mechanism design that encourages two types of participants, data nodes and consensus nodes, to collaborate, while the consensus nodes compete among themselves. The mechanism rewards the consensus nodes for generating a valid block or a ommer block which can contribute good generalization performance in terms of testing accuracy.

The rest of this article is organized as follows. Section 2 overviews the blockchain consensus mechanisms in the literature and comparatively analyzes our contributions. In Section 3, the proposed PoLe is described including the participating nodes, consensus protocol, data structure, the secure mapping layers and rewards distribution. The analysis and discussion on the incentive and security issues are presented in Section 4. Section 5 experimentally evaluates the efficiency and effectiveness of the blockchain system with PoLe. Finally, we conclude the study in Section 6.

2. Related work

Blockchain is a distributed platform for the internet of value [16,17] and has gained tremendous momentum in the past decade. Blockchain enables distributed parties who do not fully trust each other to maintain a shared ledger consistently [18]. Data consistency among blockchain distributed nodes relies on the consensus they reach via the consensus algorithm. Bitcoin uses Proof of Work (PoW) consensus mechanism [1]. In Bitcoin's PoW mechanism, consensus nodes distributively and individually solve a cryptographic puzzle by finding a nonce such that the hash of the nonce together with a proposed new block (containing the hash of the previous block) is smaller than a difficulty target which is a hash value starting with a certain number of zeros [19]. Once a valid nonce is found, the node broadcasts the new block in the blockchain network and other nodes can verify and accept the first-valid one through updating their local chain and continue to mine the next block. It is worth noting that the consensus nodes do not necessarily have synchronized clocks. Temporary forkings can be effectively finalized by a certain scheme, for example, in Bitcoin, the longest-chain rule is applied where only the longest chain is regarded as the valid chain on which the next block will be built. Due to the fact that the later blocks on the chain contain the hash of the previous block, tampering with an existing block requires regenerating all subsequent blocks through solving the required nonces. The compute power required by this process is prohibitively high, which guarantees the data security and data consistency of the blockchain systems.

The downside of PoW is that the blockchain network requires a tremendous amount of computing power which consumes a lot of energy to solve the cryptographic puzzle problems. Motivated to reduce the computational demand of PoW, proof of stake (PoS) [20], delegated proof of stake (DPoS) [21] have been proposed as alternatives of PoW. PoS dynamically adjusts the cryptographic puzzle difficulty of each node according to the amount and time of tokens held by different nodes, so as to form a situation where the nodes with more token age consumed (TAC) are more likely to take the lead in calculating valid blocks [20]. Although PoS reduces the need for computing power, it places trust in participants with a large amount of tokens, whom may collude to become a centers and an arbitrator of the network. Centralization may erode trust in the fairness of the consensus. This concern may have played a role in the slow transition of the Ethereum platform which is attempting to transit from PoW to PoS [22]. Other alternative consensus mechanisms such as proof of reputation (PoR) [23], credit-based PoW [24], proof of authority (PoA) [25], BlockDAG [26], and PBFT based protocols [27] have been also proposed in the literature. To date, PoW remains the most popular and widely accepted choice, especially in permissionless blockchain architecture [28].

Building a blockchain network based on machine learning model training has been investigated in several research studies since 2019 [13–15]. Considering the characteristics of deep neural network training tasks serving as a mining puzzle, we summarize three desirable and challenging properties for this type of consensus mechanisms: customized neural network task, no pre-training, and trained model anti-theft. The comparative analysis of the existing studies is presented in Table 1. Specifically, [13] proposes a proof-of-useful-work in Coin.AI where the mining scheme requires training deep learning models. A hash-to-architecture mapping is designed to establish a subjective function between a hash value and a valid deep learning architecture setup, achieving no pre-training and trained model anti-theft properties. However, the model architecture cannot be customized. In [14], the concept of proof-of-learning is proposed, which is inspired by machine learning competitions to help improving customized target models for suppliers. The miners of blockchain consensus network serve as model trainers. The trained model is evaluated by randomly selected validators based on a verifiable random function (VRF). However, the trained model is exposed to the validator and anyone with this model

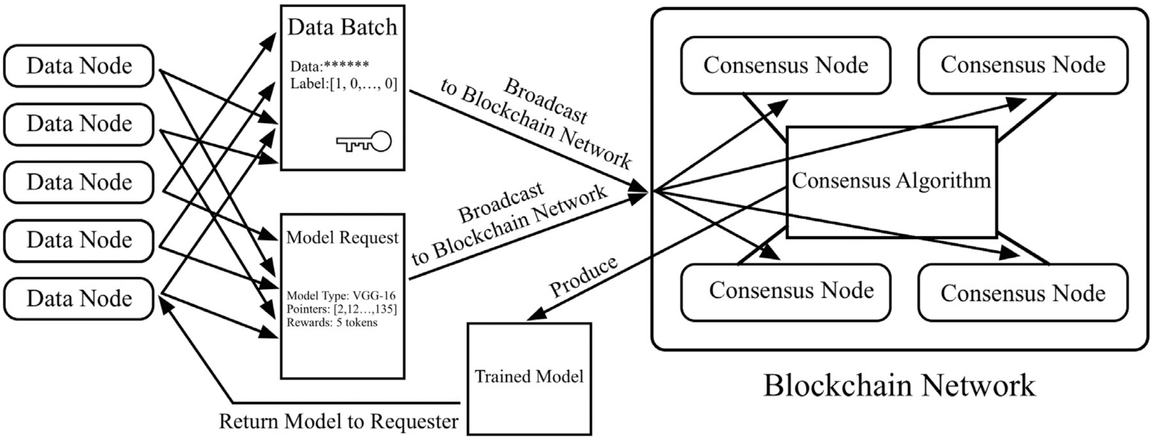


Fig. 1. Generalization of Data Nodes and Consensus Nodes in the Proposed System.

Table 1
The comparative analysis of deep learning based consensus mechanisms.

Studies	Properties		
	Customized NN Task	No Pre-training	Model Anti-theft
[13]	No	Yes	Yes
[14]	Yes	Yes	No
[15]	Yes	No	Yes
Our work	Yes	Yes	Yes

can pass the verification process. The most recent study is proof-of-useful-work proposed in [15] aiming to build better AI systems using the security of the blockchain. New blocks are generated based on a variant PoW whose difficulty is adopted based on trained model weights and performances, and the integrity of the trained model is ensured by auditing the sequential messages happening in the training process. However, a model can be pre-trained. Therefore, all these studies cannot fit all the three properties, and our study aims to fill this gap.

Furthermore, in contrast to traditional encryption schemes with "all-or-nothing" access, functional encryption (FE) schemes provide fine-grained access control, which allows the data to be accessed in terms of a given function and nothing else. For practical purposes, most of the studies on FE focus on the inner product based FE (IPFE) [29–32]. In the design of the proposed PoLe, we apply the general IPFE in [29] and IPFE with function-hiding (IPFE-FH) [30]. The security of the two encryption schemes are based on the Symmetric External Diffie–Hellman (SXDH) assumption which is sufficiently feasible in our protocol design.

3. The system model

The proposed system is a decentralized peer-to-peer network composed of two types of entities: data nodes and consensus nodes. Communication between nodes is secured via public–private asymmetric encryption schemes and happens in the form of blocks being broadcast to the entire network. Fig. 1 shows the generalization of the data nodes and the consensus nodes in the proposed system model. A data node is a user who commissions machine learning tasks via the blockchain. The consensus nodes are the suppliers of the computing power to the system; they compete to train a model that meets the requirements as specified by the data node. The winner node receives the reward offered by the data node. Besides distributing model training rewards, the blockchain also functions as a decentralized data store of encrypted data and ordinate transfer transactions. We describe the main components of the system as follows.

3.1. Data node

Data nodes are entities who commission machine learning tasks to the computing power suppliers, or consensus nodes. A task issued by a data node contains training dataset, a specification of the desired machine learning model, a minimum accuracy, and a reward. The training dataset is encrypted (see Section 3.4) and stored in blocks. A task request contains hash pointers to the training set, but not the test set. At the time of request, the task is broadcast to the entire network and added to the global task list. The global task list is an important content in the latest valid block (the previous one of the current block). The reward is immediately transferred from the data node's account to a virtual reservoir account, which will pay the winning consensus node with the best generalization performance. This account can only pay block rewards (block rewards and ommer rewards) in the first transactions of their blocks and all the other transactions transferring from this account are treated as invalid.

The model specification includes a full specification of network architecture such as the number and types of layers and their interconnections, but may leave out hyperparameters used for training, such as the learning rate or the weight decay coefficient. The model specification should be sufficiently detailed so that one can perform inference based on the specification from a complete set of model parameters. The specification also includes an accuracy metric, a minimum training accuracy that must be achieved, and a time limit for training.

The test set should remain off the blockchain in order to prevent malicious consensus nodes from using the test set for training. Therefore, the data node only broadcasts the test set after it starts to receive trained models. The data node can decide to wait for a number of solutions to arrive before releasing the test set. After the test set is released, no solutions from consensus nodes will be accepted. This is ensured through checking against the timestamp of the test set which is signed by the data node to prevent forgery.

Although nodes in a distributed system may not have perfectly synchronized clocks, the time discrepancy is far smaller than the time it takes to train a machine learning model [33]. The test data for each task is stored in the current block for which consensus nodes have competitively contributed their compute power in training the corresponding task. With the pointers of the test data in the block header, any node receiving the new valid block can validate the proposed winning solution. The task list in the valid block becomes the global task list by adding new received tasks and excluding the trained task.

3.2. Consensus nodes and the PoLe consensus

Consensus nodes, also called miners, are suppliers of compute power to the network. The miners compete to perform model training tasks issued by data nodes and are rewarded as a result. The behavior of the consensus nodes follow the Proof-of-Learning consensus protocol in Algorithm 1.

Algorithm 1 The PoLe Consensus Algorithm

```
Input:  task_list: the task list stored in previous block
        blk_chain: the blockchain
        PHS: the hash value of the previous block
        CHS: the hash value of the current block
Output: blk: the new generated block
 1: task ← PopMostValuable (task_list)
 2: train_data ← CollectData (task.data_pointers)
 3: SMLayer ← CreateSMLayer(PHS,CHS, task.model.input_length)
 4: sm_model ← InsertLayer(task.model, SMLayer)
 5: received_blks ← []
 6: while t < time_max && not received test_data do
 7:     train sm_model for one step
 8:     calculate train_accuracy
 9:     if train_accuracy ≥ task.required_accuracy then
10:         blk ← CreateBlock(sm_model)
11:         broadcast blk to other consensus nodes
12:         Append(received_blks, blk)
13:     end if
14:     if received a new solution blk then
15:         Append(received_blk, blk)
16:     end if
17:     t++
18: end while
19: if not received test_data then
20:     test_data=training_data
21: end if
22: sort received_blks in descending order of test_accuracy
23: for each blk in received_blks do
24:     if VerifyBlock(blk, PHS, test_data) then
25:         Append(blk_chain, blk)
26:         return blk
27:     end if
28: end for
```

When idle, a consensus node selects the task with highest value from the task list in the last block in its blockchain, which represents the best consensus task known by this node, and begins training the task. The value of a task is determined by the average reward in a unit of time. Suppose the previous block is consistent, then the highest value task should be the same for all the miners. Next, it performs several maintenance steps (Line 2–4) such as collecting data according to the task description and initializing the model parameters and creating the secure mapping layer (SML, see Section 3.4) which transforms the data in ciphertext to feature vectors for training the task model. Since the generation of SML is related with the current block hash (CHS) and the previous block hash (PHS), the malicious nodes are unable to start mining before the generation of the previous block.

After that, the miner optimizes the specified machine learning model using a method of its choice (Line 7). When the trained model meets the minimum training accuracy, the miner broadcasts a new block declaring its success and adds the new block at the end of local chain (Line 9–13). The miner then continues to train its task model if the training time limit has not been reached and the test_data has not been released (Line 6). If test data has been released before it completes the training task, the miner understands it loses the competition in this block height (block index) and terminates its training.

Before a miner completes its own training, it may receive other new blocks from other miners who claim to have completed the task properly. If the miner has not received the test data, it saves these blocks and continues training (Line 14–16).

Algorithm 2 VerifyBlock (blk, PHS, test_data)

```
Input:  blk: received new block
        PHS: the hash value of the previous block
        test_data: the testing dataset
Output: verified: True or False
 1: CHS=hash(blk);
 2: SMLayer1 ← CreateSMLayer(PHS, CHS,1)
 3: if SMLayer1 != blk.model.SMLayer(1) then
 4:     return False
 5: else
 6:     test_accuracy ← CalcAccuracy(blk.model, test_data)
 7:     if test_accuracy ≥ required_accuracy then
 8:         if blk.timestamp < test_data.timestamp then
 9:             return verified = True
10:         end if
11:     end if
12:     return verified = False
13: end if
```

It is possible when the maximum training time is up but the test_data is not released by model requester intentionally or unintentionally. In this case, the training_data is treated as the test_data for verification purpose (Line 19–21). In this special case, the miner who contributes the first block satisfying the required accuracy will become the block winner .

The miner will terminate its training session for any of the following three reasons: (1) the miner has found a model that achieves the minimum training accuracy; (2) the maximum time allowed for training has been reached, or (3) the miner has received the test data, indicating no further model solutions would be accepted. After consensus nodes receive a series of blocks, they first compare the test accuracy of these blocks and sort them in a descending order of the test accuracy (Line 22). Then, miners evaluate the validity of each block according to Algorithm 2 (Line 29). The computational time complexity of this algorithm is $O(1)$. A block firstly passes the verification process when Algorithm 2 returns true, then the block become the winning block (Line 24–27). The remaining blocks satisfying the requirement become ommer blocks.

When a new block is accepted as a winning block, the test data is then appended in the body of the block and the rewards of the task is transferred to the block owner. The task list in the new block is formed by subtracting the completed task from the original list in the previous block and adding the newly collected tasks. Consensus nodes will consider the transactions contained in the winning block to be valid and generate new blocks by attempting the next task from the task list of the winning block. The whole network will only admit the transaction in the winning block. The next winning blocks can get additional rewards by referring these ommer blocks, and the producer of the ommer block can also get rewards. The detail discussion of the rewards and incentives is presented in Section 3.5.

The winning block and ommer blocks are then appended to the miner's blockchain. Fig. 2 shows the whole process of collecting blocks, verifying blocks, comparing blocks and adding blocks to the blockchain.

Algorithm 2 shows how a received block is verified. Before a miner verifies a block, the first secure mapping layer is generated based on the PHS and CHS (Line 1 and 2) and the verification is denied if the calculated secure mapping layer is not the same as that in the new block. Otherwise, the miner verifies the test accuracy based on the released test data (Line 6). If the test accuracy is greater than the required accuracy then the block passes the verification (Line 7–11).

When the training data is completed before the specified time and the accuracy meets the task requirements, if more then one block is verified at the same time, the first verification-pass block will be considered as the winner, and the remaining blocks will be considered

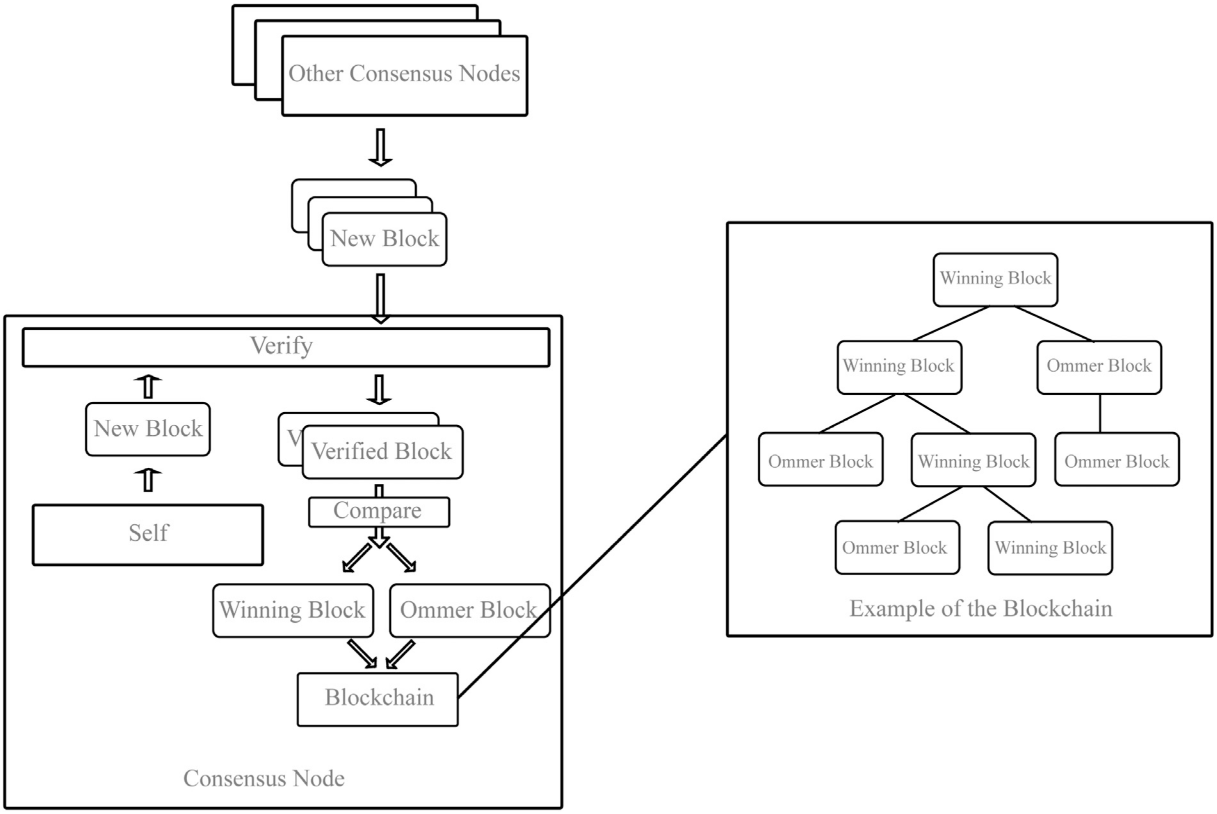


Fig. 2. Validating and Adding Blocks to the Blockchain.

as ommer blocks. When the training time exceeds the maximum completion time, the block with the highest accuracy is considered as the winner, and there is no ommer block in this case.

3.3. Data structure of blocks

The data structure of a block consists of a header and a body, which is similar to that in Bitcoin. The overall structure of the block is shown in Fig. 3. A block header contains block ID, winner's ID, selected task, previous block hash, trained model (TM), model's training accuracy, ommer block's hash and merkle tree root hash of body data. A block body stores the data organized in a binary branching merkle tree. The data in a block body includes the list of uncompleted tasks in the previous block, the list of newly collected tasks, the encrypted data uploaded by the user and transactions and test data set for solved task.

3.4. Secure data storage and secure mapping layers

To protect the privacy of training data and prevent training in advance or trained model being stolen, a secure mapping layer is designed in this section. In our design, a data node submits data to blockchain after encryption and the consensus nodes access the inner product between data $\boldsymbol{x}$ and an arbitrary function vector $\boldsymbol{z}$. We follow a general inner-product functional encryption (IPFE) [29] and IPFE with function-hiding (IPFE-FH) [30] simultaneously, where IPFE aims for verification and to prevent a model being trained in advance and IPFE-FH aims to protect the trained model from being theft by hiding functional keys.

Specifically, we consider a d-dimensional feature vector $\boldsymbol{x}$ to be encrypted and stored. The data functional encryption and decryption process can be described in four steps:

Step 1: Key generation. The master public keys, master private keys, and public parameters for IPFE and IPFE-HF are generated. These keys can be locally generated by data nodes or returned by a trust key generation service. Even if the keys are exposed, it is still secure for the encrypted data due to the nature of these encryption schemes.

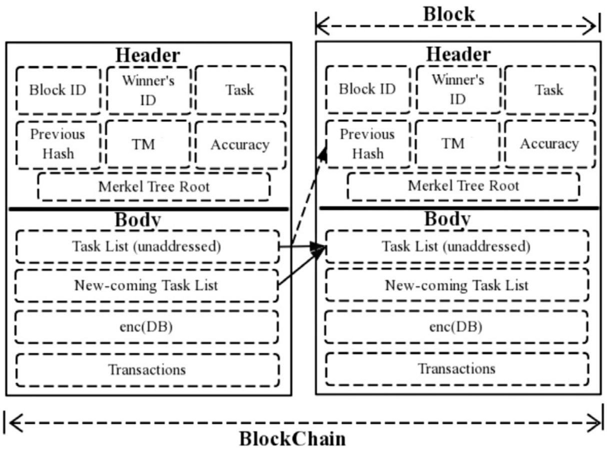


Fig. 3. The Designed Block Structure.

Step 2: Data encryption. Each data should be provided in two forms of ciphertexts for IPFE and IPFE-FH respectively. Data owners can update their data encryption keys as long as they enclose the corresponding keys in the task with the data involved.

Step 3: Query vector generation. In order to maintain the data integrity of the blockchain, at every consensus node, a set of query vectors $\{\boldsymbol{z}^{(i)}\}_{i=1}^{I}$ are generated from the hash of the previous block (PHS) and the current block (CHS). Algorithm 3 shows the detailed procedure. The computational time complexity of the algorithm is $O(D * I)$ where D is the dimension of the input data and I is the input dimension of the neural network. The function vectors are generated

Algorithm 3 Secure Mapping Layer Generation CreateSMLayer(PHS,CHS, n)

```
Input:  PHS: the hash value of the previous block
        CHS: the hash value of the current block
        D: dimension of encrypted data
        n: the number of one-layer neural networks
        k: the number of bits for each weight
Output: SMLayer: one-layer neural networks with weights ${z^{(i)}}_{i=1}^{n}$.
 1: Generate $I$-dimensional none-repeat-none-zero integer vector $R$
 2: R=R+1, R(1)=1
 3: for $i$ from 1 to I do
 4:     bPHS ← ToBinary(PHS+CHS)
 5:     primary_weight ← XOR(bPHS, bPHS $\ll R(i)$)
 6:     master_weight ← primary_weight
 7:     while # bits of master_weight $< kD$ do
 8:         master_weight ← Append (master_weight, primary_weight)
 9:     end while
10:     for $j$ from 1 to D do
11:         $z_j^{(i)}$ ← ToInteger( master_weight [j(k-1) : jk] )
12:     end for
13: end for
14: SMLayer=[];
15: for $i$ from 1 to I do
16:     SMLayer+=NN($z^{(i)}$)
17: end for
18: return SMLayer
```

through switching different number of bits in the XOR operation (Line 5 of Algorithm 3). Only the first function vector z^1 is determinate and any node can derive this vector based on the previous block and current block hashes. When a trained model is being verified by other consensus nodes, the first function vector can be generated again (see Algorithm 2). Models that do not use the correct function vector can thus be identified and discarded. All the other function vectors are kept secretly by consensus nodes, but the secret keys based on these function vectors are recorded in SML so as to output the inner production with the ciphertext of $\boldsymbol{x}$.

Step 4: Inner-product-based decryption. In decryption, a consensus node does not recover $\tilde{x}$. Instead, the consensus node finds the inner product $\langle \tilde{x}, z \rangle$ for the predetermined function vectors z. The inner production (IP) operation of z^1 with $\tilde{x}$ are based on IPFE, and the IP operations of other function vectors (z^2 to z^D) is based on IPFE-FH.

In practice, the consensus node employs I function vectors $\{z^{(i)}\}_{i=1}^{I}$ to derive their inner products with $\tilde{x}$ with dimension I, which become the input features to the neural network. This is implemented as a neural network layer that is positioned before the input layer of a neural network model and remain unchanged during the model training process.

Fig. 4 shows an example of a neural network model with a secure mapping layer with two middle layers with 256 and 128 nodes, respectively. Suppose there are n input nodes, then our secure mapping layers would generate n function vectors and each vector is significantly different for different miners, resulting in the original neural network (4-layer model in Fig. 4) being trained with different inner products of the training data. In other words, the trained model can only pass the verification with the respective miner's customized secure mapping layers.

3.5. Reward distribution

Once a node generates a valid block, it will receive the reward offered by the data node and a fixed reward issued by the blockchain system R_w. In addition, it can also get rewards by referring to previous ommer blocks. Those producers of ommer blocks can also get rewards when winning blocks refer to their ommer blocks. A winning block can only refer to the ommer block with the highest test accuracy at each block height. The rewards for the ommer generator is R_r as calculated below,

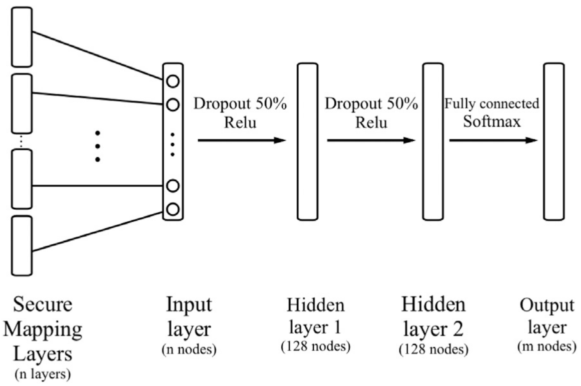


Fig. 4. An Example of A Neural Network with A Secure Mapping Layer.

$$R_r = R_w/((H_w - H_o) * O_{num}) \tag{1}$$

where H_w stands for the height of the winning block, H_o stands for the height of the ommer block and O_{num} is allowed maximal number of ommer blocks in height H_o. Here the total rewards for a block (the fixed reward and the referred ommer blocks) in a height are no greater than the block task rewards, which can be ensured by setting R_w as the half of the task reward.

4. Security analysis

We discuss how the proposed model performs against possible security attacks, including chain revision, model theft, pre-training, and consensus nodes collusion.

4.1. Chain revision

The first and most common attack in a blockchain system is chain revision where an attacker rewrites the content of the blockchain and publishes the modified chain. In PoW protocol, this is prevented by the computational difficulty of finding a nonce whose hash value is smaller than a target value. In PoLe, this is prevented by the difficulty of training deep learning models and the SML mechanism connecting the adjointing blocks with their hashes.

In PoLe, similar to PoW, a new block contains the hash of the previous block, which forces the attackers to recalculate all blocks after the modified block. Furthermore, the task to be trained for the new block is retrieved from the previous consensus block, which ensures the task to be publicly verifiable for all the distributed consensus nodes. This design increases the cost for attackers in manipulating the chain in our system and ensures data integrity.

4.2. Pre-training of task models

Since all training data are available on the blockchain ledger, a malicious consensus node may start training before other nodes in order to gain an unfair advantage. The SML is designed to prevent this behavior. As the SML weights are computed from the hash value of the previous block and the current block which contains the identity of the consensus node, no consensus nodes can start training before the previous block is confirmed by the distributed consensus. In addition, consensus nodes cannot cheat by training on test data because any model broadcast later than the release of test data will be rejected.

4.3. Model theft

Model theft refers to the possibility that a consensus node may take a neural network model trained and published by another node and claim ownership of that model by broadcasting it to the blockchain network.

First, the design of SML provides defense against the attempts to steal models trained by other nodes. Given the fact that each miner should always create a coinbase with itself as receiver, the generated block hash CHS thus contains the consensus node identity information. The SML further converts CHS and PHS to a set of functional query vectors with the first vector verifiable (with the content switch R(1)=1 in Line 2 of Algorithm 3), so that it is easy to verify whether the publisher of a network model is its creator or owner. SML maps the encrypted data input through inner production operations, and its output data for different nodes at different block heights is different. The conventional NN is then trained based on such output and the resulting trained model is identified by its unique SML. Thus, our SML attaches a trainer's identity via input data transformation to its model.

Second, the SML design prevents the following impersonation attack that trivially modifies the network weights. An attacker may manipulate the input layer as follows. Let the network creator's decryption function vectors be $Z = \{z^1, z^2, \dots, z^n\}$ and the thief's decryption be $\mathcal{Z} = \{z'^1, z^2, \dots, z^n\}$. Assuming the first layer of the network utilizes a linear layer with parameter W, the output of this layer is then $f(X) = WZX$, where X is a data vector in plaintext. The thief's goal is to adapt the stolen network's weight so that the function $f(X)$ remains unchanged. Actually, this attack is not effective in our system. Suppose z^2 to z^n are publicly known by the theft, the thief cannot derive the $\mathcal{Z}$ through left inverse of $\mathcal{Z}$ denoted as $\mathcal{Z}_L^{-1}$, and thus cannot set the first layer's weight as $WZ\mathcal{Z}_L^{-1}$. In this way, the input to the second layer of the model will be $f'(X) = WZ\mathcal{Z}_L^{-1}\mathcal{Z}X = WZX = f(X)$, which is unchanged. In our model, z^2 to z^n are secretly kept by the network creator and guessing these vectors is an NP-hard problem.

4.4. Collusion of consensus nodes

Blockchain systems are known to be vulnerable to the collusion of the majority consensus nodes, which is also called as the 51% attack. As an extension of blockchains, the PoLe consensus is not immune to such attack. When a set of nodes form a mining pool and coordinately generate the same new block, the SML for this collusion is the same, it may results in the mining advantages. However, a collusion formed by self-interested consensus nodes is difficult due to the following two reasons: (1) the effectiveness of NN model training is a convex optimization problem [34] and a NN model trained by collusive distributed nodes is less effective than training the model by a single node powered by all these nodes' compute resources. (2) The block rewards are paid by the model requester based on the test accuracy in a verifiable manner, and the model requester has no incentive to collude with the mining nodes through accepting an unqualified model.

4.5. Distributed consensus issue

The proposed PoLe based system is different from the conventional distributed NN training systems. For each block height, every consensus node independently trains a consensus task determined by the previous block. When the test data for a task model is published, all the consensus nodes cannot accept new models anymore. They would verify the legal block winner among their received models and accept the one with the highest performance accuracy and continue to mine the next block. Thus, the consensus process does not require synchronized clocks among the nodes but act in a decentralized manner. Through following the proposed PoLe, the nodes can verify the contents in blockchain ledger and sustain their chain update status consensus. Furthermore, the data integrity of the ledger is guaranteed by the cryptography hash function which is technically safer than the certification authorities in conventional distributed systems.

5. Experiments

The purpose of our experiments is to evaluate the efficiency and effectiveness of the proposed blockchain system with PoLe. We carry out three sets of experiments. The first experiment is designed to verify whether PoLe can reliably control the block generation time compared with PoW. If a blockchain system can generate blocks at regularly stable intervals, transactions carried out through the system will not experience unexpected delays, improving the smoothness of the user experience. The second experiment compares the accuracy of deep learning models trained from a PoLe-based blockchain system and traditional methods for the models. This experiment aims to evaluate the effectiveness of the model trained on encrypted data and examines the training accuracy of PoLe. The third experiment is to study the effect of the same data on the accuracy when using manipulated SML. This experiment is used to study whether it is possible to forge workload by stealing other trained models and replacing them with their own SML.

We utilize three datasets, MNIST, IRIS, and CIFAR-10. The MNIST dataset represents an optical digit recognition task from black-and-white images. We follow the original split of 60,000 images for training and 10,000 images for testing. The IRIS dataset consists of 150 data samples about a classification task with 3 classes, and we divide the data into training set taking 90% and testing set taking the other 10%. CIFAR-10 dataset is about an image classification task with 10 classes for 32 × 32 images, and the training data contains 50,000 images and the testing set contains 10,000 images.

5.1. Experiment 1: Variance in block generation time

When a blockchain-based distributed ledger is used to support financial transactions, an important performance indicator is the consistency in the number of transactions that the ledger can record per unit of time. This is directly reflected in the average time it takes to generate a block and the variance in that time. A blockchain with a high variance in block generation time may lead to poor user experience because users may need to wait a long time before their transactions are recorded by the chain. In this experiment, we evaluate the variance in block generation time on PoW and PoLe chains.

We build two blockchain systems based on PoW consensus and PoLe consensus respectively. For the PoLe blockchain, we employ 5 data providers and 3 consensus nodes. The number of consensus nodes in PoW is also 3. In PoLe blockchain, we set the training accuracy threshold to 0.8 considering the limited simulation compute power provided by the experimental laptop with CPU i7-4770K at basic frequency 2.2 GHz, and the training time is about 220 s. Each PoLe consensus node uses the same task in each round, that is to train MNIST data set on a conventional neural network (CNN), which is composed of four conventional layers, two maxpooling layers with pool size (2,2), two fully connected layers, and a 50% dropout layer. The two maxpooling layers are behind the second and fourth conventional layers respectively, and the dropout layer is located in front of the output layer. The number of channels of the CNN layers is 64, 64, 32 and 32 respectively, and the kernel size is (3,3), and there are 256 and 10 neurons in fully connected layers, respectively. For each task, the learning rate is set to 0.001 and the loss function is set to categorical crossentropy. After each training epoch, the node will test on the same source test set of 10,000 samples, and compare the test accuracy with the threshold. To set a proper difficulty for PoW, we do a set of experiments by varying the difficulty levels for 3000 s, and the average block generation time is listed in Table 2, in which PoW difficulty is represented by the number of zeros of the generated block hash. We choose PoW difficulty as 23 in this experiment and the expected block generation time is mostly close to 220 s. Both blockchain systems run to generate 50 blocks. We record the time it takes to generate every block.

Table 2
PoW difficulty setting.

PoW difficulty	Average time (s)	Var
2	0.00477	0.033
4	0.00333	0.15
6	0.00848	0.49
8	0.05494	2.07
10	0.17184	8.863
15	2.20998	321.94
21	67.088	403.657
22	69.149	5101.2
23	**293.72**	**130548.84**
24	321.251	133586.55
25	554.8987	206338.7

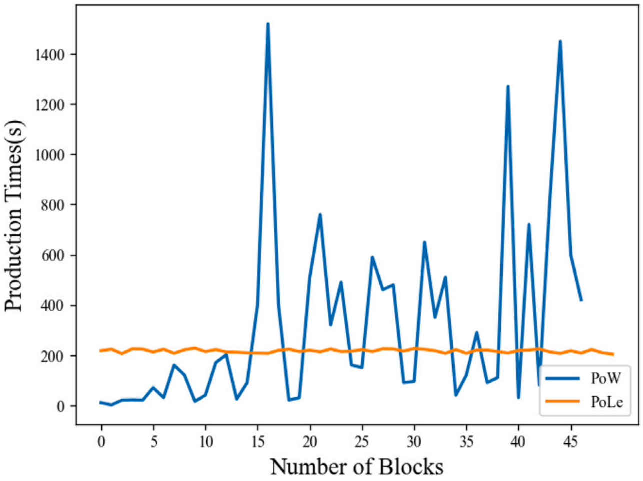


Fig. 5. Block Generation Time of PoW with Difficulty =23 and PoLe with Model Accuracy 0.8.

The block generation time of PoLe and PoW is shown in Fig. 5. We observe that PoLe has a much lower variance in block generation time than PoW, with the range between 200 and 250 at the variance of 140. In the experiment of PoLe, each consensus node only needs to train 1–3 epochs to exceed the threshold accuracy on the test set, and each epoch only takes about 1 min. In other words, the CNN training process has a consistent convergence speed, despite its apparent stochasticity. In contrast, due to the properties of a well-behaving hash function, the success of a nonce guess in PoW follows a high-variance distribution.

The reader may notice that we use the same machine learning problem repeatedly in our experiments. It is because that the miners cannot reuse a network trained for previous tasks as the solution to a new task, and the new task's data has to be decrypted by the secure mapping layer using the hash of the previous block. This feature prevents "gun jumping" or solution reuse.

5.2. *Experiment 2: Accuracy of model training*

We design SML to prevent cheating behaviors. However, the involvement of SML may lower the performance of the neural network because the data have been projected onto some random function directions. In this experiment, we measure the performance difference between secure networks with SML and unsecure networks without SML.

We create two neural network models, a multi-layer perceptron (MLP) and the VGG-16 network [35]. There are four fully connected layers in the MLP model, which contain 512, 256, 128 and 10 neurons respectively, and a 50% dropout layer after the third fully connected layer. For both networks, we create a secure version with SML inserted before the first layer and an unsecured model without SML. On MNIST dataset, the value of k in Algorithm 3 is set as 3. The output dimension of SML is set as 128 for MLP and 256 for VGG-16. The number of function vector s is set as 32 for MLP and 64 for VGG-16, respectively. To adapt MNIST to the input shape of VGG-16 model, we adjust the data size to 48 × 48. In addition, we test the MLP model on the simple IRIS dataset. The MNIST MLP model and IRIS MLP model are trained for 400 epochs, while the MNIST VGG-16 model is trained for 100 epochs. Finally, we train VGG-16 on CIFAR-10 for 120 epochs with SML output dimension as 2700.

Table 3
Accuracy and time cost of MLP and VGG-16 task training.

Model	Type	Accuracy (%)	Time cost (min)
MNIST MLP	Secure	99.90	14.55
	Original	99.52	11.26
MNIST VGG	Secure	99.70	78.56
	Original	99.93	64.51
CIFAR10 VGG	Secure	91.25	93.41
	Original	91.70	71.28
IRIS MLP	Secure	100	5.14
	Original	100	3.25

Table 4
Model accuracy using original SML and manipulated SML on MNIST.

Model	Accuracy(%)
Model + Origin SML	99.50
Model + Manipulated SML	13.48

The experiment was carried out on a server with Xeon Gold 6161 CPU at a basic frequency of 2.20 GHz. Figs. 6 shows the training accuracy and testing accuracy curves of MLP on MNIST. Similar results can be observed for models in different datasets with and without SML. Table 3 reports the final accuracy and the elapsed time. We observe that both the secure and unsecure models are trained to achieve similar performance, and the introduction of SML does not hinder model performance. We also observe an acceptable increase in the time needed for the secure models to complete the same number of epochs. The time needed to complete training increased by 29% for MLP and by 21% for VGG-16. This phenomenon is expected as SML creation brings additional computational complexity and could be mitigated with more efficient implementations.

Overall, we conclude that despite the input features being projected onto different directions, the SML does not harm predictive performance significantly by a maximal decrease value 0.23% in the evaluated four datasets in Table 3.

5.3. *Experiment 3: Effects of SML manipulation*

A consensus node may be encouraged to reuse a model by replacing the SML. For example, after a trained model has been broadcast to other nodes, another node may replace the SML layer and submit the solution as its own. Similarly, if a consensus node realizes the data node is re-issuing an old task, it may attempt to replace the SML in an old solution and resubmit it. Therefore, we empirically study the predictive accuracy of a trained model for the same data when the SML is manipulated.

We use the MNIST data set and measure the performance of the model trained in Experiment 2, as well as a model with a manipulated SML. Table 4 shows the accuracy of the two models. We can find that the accuracy of the model for the same data set is reduced by 86.02% when the SML is manipulated. This observation demonstrates that manipulating SML of a model will result in intolerable performance degradation, which allows the blockchain network to easily detect model-theft cheating.

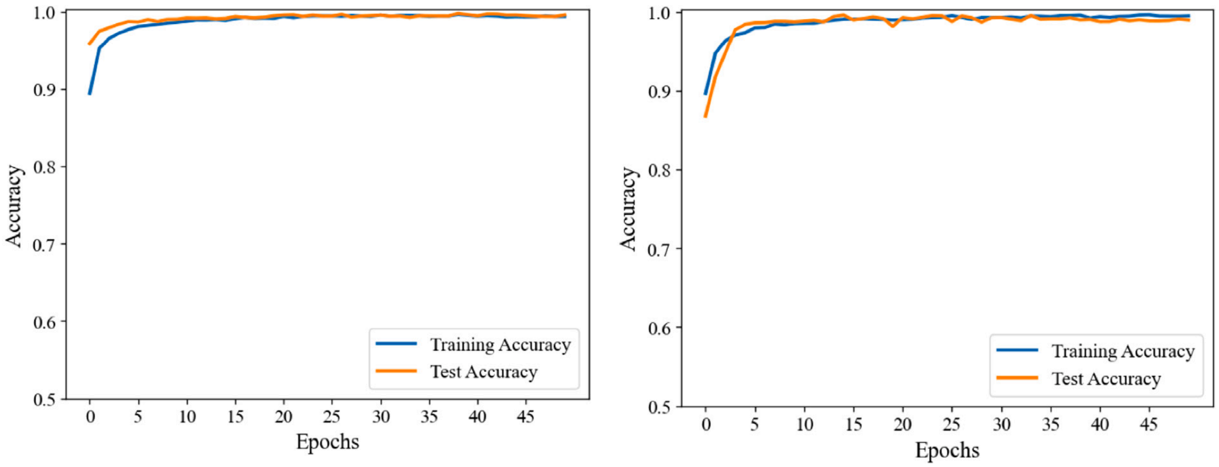


Fig. 6. Training and Test Accuracy Curves of Original and Secure MLP Models on MNIST.

6. Conclusions

Rapid progress in deep learning has created an unsatisfied demand for computation power. PoW-based Blockchain systems can effectively ensure data security at the cost of wasting huge computer resources. Starting from the realization that the training and testing of a machine learning model have asymmetric computational demands, we propose a new consensus algorithm called PoLe, which channels otherwise wasted compute on blockchain to the practical benefits of training machine learning models. We further design a secure mapping layer to enable the proposed system to achieve no pre-training and trained model anti-theft properties. Experimental results confirm that (1) PoLe is capable of producing a reliable stream of blocks, which support stable data storage in a decentralized manner, and (2) the security property of PoLe does not sacrifice model generalization performance. As neural network powered AI applications and blockchain networks continue to grow in the foreseeable future, we believe the proposed PoLe consensus mechanism will contribute to meeting their demands for computing resources and reducing environmental impact from high energy consumption.

In future work, we plan to evaluate the proposed PoLe in various deep learning model settings where the impacts of hyper-parameters of deep learning models will be empirically studied, guiding the potential improvement of PoLe in the next version.

CRediT authorship contribution statement

Yuan Liu: Conceptualization, Formal analysis, Methodology, Project administration, Roles/Writing – original draft, Funding acquisition. **Yixiao Lan:** Investigation, Data curation, Visualization. **Boyang Li:** Investigation, Resources, Roles/Writing – original draft, Validation. **Chunyan Miao:** Supervision, Validation. **Zhihong Tian:** Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China under Grant No. 62172085; Key-Area Research and Development Program of Guangdong Province No. 2020B0101090005; National Natural Science Foundation of China under Grant No. 62032013, and No. U20B2046; 111 Project (B16009); the Fundamental Research Funds for the Central Universities N182410001. This work is partially supported also by the Singapore National Research Foundation (NRF) under NRF Investigatorship (NRF-NRFI05-2019-0002) and NRF Fellowship (NRF-NRFF13-2021-0006).

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008, White Paper.
[2] Y. Liu, F.R. Yu, X. Li, H. Ji, V.C.M. Leung, Blockchain and machine learning for communications and networking systems, IEEE Commun. Surv. Tutor. 22 (2) (2020) 1392–1431.
[3] G. Nguyen, K. Kim, A survey about consensus algorithms used in blockchain, J. Inf. Process. Syst. 14 (1) (2018) 101–128.
[4] D.D.F. Maesa, P. Mori, Blockchain 3.0 applications survey, J. Parallel Distrib. Comput. 138 (2020) 99–114.
[5] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, Y. Yang, A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling, ACM Comput. Surv. 53 (1) (2020) 18:1–18:32.
[6] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, L. van der Maaten, Exploring the limits of weakly supervised pretraining, 2018, arXiv:1805.00932.
[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019, ArXiv Preprint. arXiv:1912.12860.
[8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: NAACL, 2019.
[9] O.O. Aremu, D. Hyland-Wood, P.R. McAree, A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data, Reliab. Eng. Syst. Saf. 195 (2020) 106706.
[10] D.R. So, C. Liang, Q.V. Le., The evolved transformer, in: Proceedings of the 36th International Conference on Machine Learning, 2019.
[11] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution forimage classifier architecture search, in: AAAI, 2019.
[12] X. Zhou, D. Dou, B. Li, Searching for stage-wise neural graphs in the limit, 2019, ArXiv Preprint. arXiv:1912.12860.
[13] A. Baldominos, Y. Saez, Coin.ai: A proof-of-useful-work scheme for blockchain-based distributed deep learning, Entropy 21 (8) (2019) 723.
[14] F. Bravo-Marquez, S. Reeves, M. Ugarte, Proof-of-learning: A blockchain consensus mechanism based on machine learning competitions, in: Proceedings of IEEE International Conference on Decentralized Applications and Infrastructures, 2019, pp. 119–124.
[15] A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics, M. Harvilla, A proof of useful work for artificial intelligence on the blockchain, CoRR (2020) arXiv:2001.09244.
[16] D. Tapscott, J. Euchner, Blockchain and the internet of value, Res. Technol. Manag. 62 (1) (2019) 12–18.
[17] X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, Future Gener. Comput. Syst. 107 (2020) 841–853.
[18] T.T.A. Dinh, R. Liu, M. Zhang, G. Chen, B.C. Ooi, J. Wang, Untangling blockchain: A data processing view of blockchain systems, IEEE Trans. Knowl. Data Eng. 30 (7) (2018) 1366–1385.
[19] M. Conti, E.S. Kumar, C. Lal, S. Ruj, A survey on security and privacy issues of bitcoin, IEEE Commun. Surv. Tutor. 20 (4) (2018) 3416–3452.

[20] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, self-published paper, 2012.

[21] D. Larimer, Delegated proof-of-stake white paper, 2014, http://8btc.com/doc-view-151.html.

[22] From PoW to PoS: Ethereum plans to reduce energy consumption, 2021, https://thetopcoins.com/blog/pow-pos-ethereum-plans-reduce-energy-consumption, Website.

[23] Q. Zhuang, Y. Liu, L. Chen, Z. Ai, Proof of reputation: A reputation-based consensus protocol for blockchain based systems, in: Proceedings of the International Electronics Communication Conference, 2019, pp. 131—138.

[24] J. Huang, L. Kong, G. Chen, M.-Y. Wu, X. Liu, P. Zeng, Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism, IEEE Trans. Ind. Inf. 15 (6) (2019) 3689.

[25] R. Saltini, D. Hyland-Wood, IBFT 2.0: A safe and live variation of the IBFT blockchain consensus protocol for eventually synchronous networks, CoRR (2019) arXiv:1909.10194.

[26] K. Gai, Z. Hu, L. Zhu, R. Wang, Z. Zhang, Blockchain meets DAG: a BlockDAG consensus mechanism, in: M. Qiu (Ed.), Proceedings of 20th International Conference on Algorithms and Architectures for Parallel ICA3PP, in: Lecture Notes in Computer Science, 12454, 2020, pp. 110–125.

[27] Z. Tian, M. Li, M. Qiu, Y. Sun, S. Su, Block-DEF: A secure digital evidence framework using blockchain, Inform. Sci. 491 (2019) 151–165.

[28] A. Gervais, G.O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 3–16.

[29] M. Abdalla, F. Bourse, A.D. Caro, D. Pointcheval, Simple functional encryption schemes for inner products, in: Proceedings of 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, 2015, pp. 733–751.

[30] P. Datta, R. Dutta, S. Mukhopadhyay, Functional encryption for inner product with full function privacy, in: C. Cheng, K.-M. Chung, G. Persiano, B.-Y. Yang (Eds.), 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, vol. 9614, 2016, pp. 164–195.

[31] I. Kim, J.H. Park, S.O. Hwang, An efficient public key functional encryption for inner product evaluations, Neural Comput. Appl. 32 (17) (2020) 13117–13128.

[32] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, D.J. Wu, Function-hiding inner product encryption is practical, in: Proceedings of 11th International Conference of Security and Cryptography for Networks, 2018, pp. 544–562.

[33] S. Ricci, E. Ferreira, D.S. Menasche, A. Ziviani, J.E. Souza, A.B. Vieira, Learning blockchain delays: A queueing theory approach, ACM SIGMETRICS Performance Evaluation Review 46 (3) (2019) 122–125.

[34] Y. Bengio, N.L. Roux, P. Vincent, O. Delalleau, P. Marcotte, Convex neural networks, in: Proceedings of Neural Information Processing Systems (NIPS), 2005, pp. 123–130.

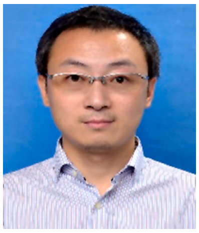
[35] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the 3rd International Conference on Learning Representations, 2015.

Yuan Liu is an Associate Professor at Software College of Northeastern University in Shenyang China. She received her B.Sc degree in the honor school, Harbin Institute of Technology, China, in 2010. She achieved her Ph.D. degree in School of Computer Engineering from Nanyang Technological University (NTU), Singapore, in 2014. From 2014 to 2015, she ever worked as Research Fellow at Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), NTU, Singapore. Her research interests include blockchain security, trust-based incentive mechanism design, multi-agent system, trust management, blockchain technology based reputation systems. Her research papers have been published in top international conferences and journals in the area of artificial intelligence and blockchain.

Yixiao Lan is a bachelor student in School of computer science and engineering, Northeastern University, China. His research interests include privacy protection of blockchain systems, machine learning based consensus mechanism.

Boyang "Albert" Li is a Nanyang Associate Professor at the School of Computer Science and Engineering, Nanyang Technological University, Singapore, and a recipient of the prestigious National Research Foundation (NRF) Fellowship from the Singapore NRF. His research interests mainly lie in Machine Learning, Multimodal Learning, and Computational Narrative Intelligence. Previously, he was a Senior Research Scientist at Baidu Research Sunnyvale and a Research Scientist and Group Leader at Disney Research Pittsburgh. In 2014, He received his Ph.D. from Georgia Institute of Technology. His research work has been covered by major media outlets such as Engadget, TechCrunch, New Scientist, and National Public Radio.

Chunyan Miao received her B.Sc. degree from Shandong University, China in 1988, and Master's and Ph.D. degrees from Nanyang Technological University, Singapore, in 1998 and 2003 respectively. She is currently a Full Professor and the Chair of the School of Computer Science and Engineering, Nanyang Technological University, and director of the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY). Her research focuses on infusing intelligent agents into interactive new media (virtual, mixed, mobile, and pervasive media) to create novel experiences and dimensions in game design, interactive narrative, and other real world agent systems.



Zhihong Tianis currently a Professor, and Dean, with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangdong Province, China. He received his B.S., M.S., and Ph.D. degree in Computer Science and Technology from Harbin Institute of Technology, Harbin, China in 2001, 2003, and 2006 respectively. He is honored as Changjiang Scholars of the Ministry of Education, Pearl River Scholar in Guangdong Province. He is also a part-time Professor at Carlton University, Ottawa, Canada. Previously, he served in different academic and administrative positions at the Harbin Institute of Technology. He has authored over 200 journal and conference papers. His research interests include computer networks and cyberspace security. His research has been supported in part by the National Natural Science Foundation of China, National Key research and Development Plan of China, National High-tech R&D Program of China (863 Program). He also served as a member, Chair, and General Chair of a number of international conferences. He is a Distinguished Member of the China Computer Federation, and Senior Member of IEEE.