

SABlockFL: a blockchain-based smart agent system architecture and its application in federated learning

Smart agent
system
architecture

133

Zhizhao Zhang, Tianzhi Yang and Yuan Liu
Software College, Northeastern University, Shenyang, China

Received 1 December 2019
Revised 5 February 2020
Accepted 9 February 2020

Abstract

Purpose – The purpose of this work is to bridge FL and blockchain technology through designing a blockchain-based smart agent system architecture and applying in FL. FL is an emerging collaborative machine learning technique that trains a model across multiple devices or servers holding private data samples without exchanging their data. The locally trained results are aggregated by a centralized server in a privacy-preserving way. However, there is an assumption where the centralized server is trustworthy, which is impractical. Fortunately, blockchain technology has opened a new era of data exchange among trustless strangers because of its decentralized architecture and cryptography-supported techniques.

Design/methodology/approach – In this study, the author proposes a novel design of a smart agent inspired by the smart contract concept. Specifically, based on the proposed smart agent, a fully decentralized, privacy-preserving and fair deep learning blockchain-FL framework is designed, where the agent network is consistent with the blockchain network and each smart agent is a participant in the FL task. During the whole training process, both the data and the model are not at the risk of leakage.

Findings – A demonstration of the proposed architecture is designed to train a neural network. Finally, the implementation of the proposed architecture is conducted in the Ethereum development, showing the effectiveness and applicability of the design.

Originality/value – The author aims to investigate the feasibility and practicality of linking the three areas together, namely, multi-agent system, FL and blockchain. A blockchain-FL framework, which is based on a smart agent system, has been proposed. The author has made several contributions to the state-of-the-art. First of all, a concrete design of a smart agent model is proposed, inspired by the smart contract concept in blockchain. The smart agent is autonomous and is able to disseminate, verify the information and execute the supported protocols. Based on the proposed smart agent model, a new architecture composed by these agents is formed, which is a blockchain network. Then, a fully decentralized, privacy-preserving and smart agent blockchain-FL framework has been proposed, where a smart agent acts as both a peer in a blockchain network and a participant in a FL task at the same time. Finally, a demonstration to train an artificial neural network is implemented to prove the effectiveness of the proposed framework.

Keywords Blockchain, Federated learning, Smart agent

Paper type Research paper

© Zhizhao Zhang, Tianzhi Yang and Yuan Liu. Published in *International Journal of Crowd Science*. Published by Emerald Publishing Limited. This article is published under the Creative Commons Attribution (CC BY 4.0) license. Anyone may reproduce, distribute, translate and create derivative works of this article (for both commercial and non-commercial purposes), subject to full attribution to the original publication and authors. The full terms of this license may be seen at <http://creativecommons.org/licenses/by/4.0/legalcode>

This work is supported in part by the National Natural Science Foundation for Young Scientists of China under Grant No. 61702090 and No. 61702084.



International Journal of Crowd
Science
Vol. 4 No. 2, 2020
pp. 133-147
Emerald Publishing Limited
2398-7294
DOI: 10.1108/IJCS-12-2019-0037

1. Introduction

With the advance of internet technology and artificial intelligence, more and more people and devices are connected and interacted in a distributed manner (Vlassis, 2007). Technologies such as parallel computing and distributed processing (including distributed artificial intelligence) have emerged and keep growing rapidly over the past 20 years (Bond and Gasser, 2014; Yang *et al.*, 2019a, 2019b).

Federated learning (FL) (Yang *et al.*, 2019a, 2019b), also known as distributed learning or collaborative learning, was proposed by Google in 2016 (McMahan *et al.*, 2016). The main goal of FL is to build a machine-learning model via data sets that are distributed across multiple data owners (Li *et al.*, 2019). Actually, it is the combination of distributed computation and deep learning. In common FL scenarios, there are many participants in possession of local data samples and a central server, called parameter server, which maintains the machine learning model. The training process can be briefly described as follows.

At the beginning stage, the data samples are distributed among participants, however, which are not necessarily independent and identically distributed. Then, each participant downloads the snapshot of the machine learning model maintained by the parameter server, then it trains this model in use of local data samples to generate parameter gradients. Then, each participant uploads its local gradients to the central server. Finally, upon receiving the gradients from all participants, the central server aggregates those gradients and updates the global model. This process is repeated until the loss function convergences.

The agent concept was firstly proposed by Minsky (1987). Minsky believed that certain individuals in society can seek solutions to problems after negotiation. These individuals are called agents. Agents should be socially interactive and intelligent. As then, the concept of agent has been introduced into the field of computer science and has quickly become a research hot topic. In addition to intelligence, an agent should have many other attractive features such as autonomy, reactivity, structural distribution, sociability and coordination of cooperation (Wooldridge and Jennings, 1994; Dragoni *et al.*, 2011). Each agent has limited capability in processing information, but through cooperating and coordinating with each other the multi-agent system could solve problems, which are nearly impossible to handle by a single individual. A multi-agent system can be regarded as a society composed by multiple agents, which is mostly organized in a distributed manner. The existing research studies mainly focus on the negotiation behavior happening among a group of autonomous agents, for example, how the distributed agents negotiate their goals; how the goal can be achieved through planning.

Meanwhile, with the fast development of blockchain technology in the past decade (Crosby *et al.*, 2016), smart contract technology has also received much attention in both academics and industry (Delmolino *et al.*, 2016; Liu *et al.*, 2019). People have developed many distributed applications (DAPP) based on blockchain smart contract platforms. A smart contract program is more than a computer program that can be executed automatically but can also be a participant in the system. A smart contract can send out, receive and store information and values, and can also respond to the received information according to the predefined logic. More importantly, such a smart contract is trustworthy and reliable, as the contract is non-repudiation and transparent once it is published in a block (Kolvart *et al.*, 2016; Kim *et al.*, 2019).

In this work, we aim to investigate the feasibility and practicality of linking the three areas together, namely, multi-agent system, FL and blockchain. A blockchain-FL framework, which is based on a smart agent system, has been proposed in Section 2. We have made several contributions to the state-of-the-art. First of all, a concrete design of a

smart agent model is proposed in Section 3, inspired by the smart contract concept in blockchain. The smart agent is autonomous and is able to disseminate, verify the information and execute the supported protocols. Based on the proposed smart agent model, a new architecture composed by these agents is formed, which is a blockchain network. Then, a fully decentralized, privacy-preserving and smart agent blockchain-FL framework (SABlockFL) has been proposed in Section 4, where a smart agent acts as both a peer in a blockchain network and a participant in a FL task at the same time. Finally, a demonstration to train an artificial neural network is implemented to prove the effectiveness of our proposed framework in Section 5.

2. Related work

2.1 Federated learning

To achieve higher accuracy, deep learning models often need large amount of training data, which can be solved by using distributed learning (Chilimbi *et al.*, 2014). Among these existing studies, FL is the widely adopted method in recent years, and it is also an effective solution for high scale secure multi-party machine learning scenarios where clients train a shared model under the organization of a secure central server.

However, the classical FL framework presents some data privacy vulnerabilities, even though the training dataset is divided and stored separately (Accountability Act, 1996; Ben-Or and Hassidim, 2005). For instance, an adversary poses as an honest client and tries to steal sensitive training data through careful observation and isolation of victims' model updates (Song *et al.*, 2017; Melis *et al.*, 2018). On the other hand, the authors in Hitaj *et al.* (2017) pointed out that a curious parameter server can learn private data through generative adversarial network learning. Therefore, has tried to solve it through replacing the parameter server with a blockchain network, which is called blockchain-federate learning (BlockFL) (Shayan *et al.*, 2018). Similar systems are already emerging in the internet of things (IoT) field (Lu *et al.*, 2019; Zhao *et al.*, 2019).

2.2 Multi-agent system

Multi-agent systems can be classified into two categories according to their architecture, namely, internal architecture and overall agent structure (Balaji and Srinivasan, 2010). The internal architecture can also be further divided into homogeneous and heterogeneous structures. The overall agent structure can also be divided into hierarchical, holographic and alliance structure.

There are many examples in using multi-agent systems to construct novel applications. For example, an agent-based multi-natural language anti-spam (MNLAS) model is proposed (Mohammed *et al.*, 2018). The MNLAS model considers visual and text information in emails during the spam filtering process. The model is composed by five types of agents, namely, the formal abbreviation agent, the feature extraction agent, the feature selection agent, the instance representation agent and the mail classification agent, and these agents complete different tasks collaboratively.

Multi-agent systems have also been widely applied in the development of distributed systems, aiming to achieve high system reliability, good availability, openness, resource sharing and re-usability. A distributed object-based agent framework for building enterprise DAPP is proposed by Zhou *et al.* (2000). A proxy communication language KQML is introduced in this framework, which is feasible for fast and easy development of a distributed system. This work also provides a general distributed architecture for multi-agent systems.

2.3 Smart contract and agent

The smart contract concept enables a blockchain (Nakamoto *et al.*, 2008; Wood *et al.*, 2014) system programmable, which provides the possibility of application with rich functionalities. A smart contract provides a trustless trust in bilateral or multi-party transactions without third-party intervention. Yuanyu Zhang studied a key access control problem on the IoT (Zhang *et al.*, 2018). A framework based on smart contract is proposed, which consists of multiple access control contracts, a judging contract and a registration contract to implement distributed, trusted access control in IoT systems.

The combination of smart contracts and agents can achieve high security in providing an intelligent service. Aleksandr builds a communication system between agents in a peer to peer network to enable multi-agents and smart contracts to work together (Kapitonov *et al.*, 2017). Based on this communication method, an architecture of an autonomous business activity protocol is proposed, toward to a self-managed unmanned aerial vehicles (UAVs) system where the UAVs transportation and airspace route management are conducted in an automatic manner. Different from the above work, our study aims to design a smart agent-based on smart contract spirit.

3. Smart agent and smart agent system

The conventional distributed computing technology is based on the classic client/server computing model, which decomposes a distributed application into two parts, namely, the client side and the server side. The server side can only provide services after receiving requests from the client side. These technologies are formed in the context of applications that share distributed resources and bear many limitations, especially in DAPP. In this section, we are to propose a new design of smart agent based on the smart contract concept.

For a smart agent, it should own the basic characteristics of an intelligent agent such as autonomy. Moreover, a smart agent should also have the features of a smart contract, for example, automatic execution once certain conditions are satisfied. In our design, when the condition that supports the occurrence of a certain activity is valid, the smart agent autonomously selects the corresponding business based on its own state. Moreover, the smart agents support the distributed operation mode, providing convenience for the development of DAPP.

3.1 Design of a smart agent

By comparing the characteristics of agents and smart contracts, we find that both of them are autonomous and can complete the established functions without the participation of third parties. At the same time, both agents and smart contracts have their own distinguishing characteristics. Our design will combine agents and smart contract through implementing the functions of smart contract in agents, so as to autonomously disseminate, verify the information and execute supported protocols.

The main components and their relationship are showed in Figure 1. In Figure 1, a smart agent is composed by six basic functional elements, namely, perception tool (PT), logical distribution (LD), business process (BP) and contract agreement (CA), as well as address management (AM) and data access tool (DAT). Each element is specified as follows:

- *Perception tool (PT)*: The PT is the administrator of the system interacting with the external environment. In a realistic system, the number of instances/entities can increase or decrease according to the actual situation. The main duty of a PT includes: collecting system environment changes, making corresponding judgments for different changes and providing basic data for business processing.

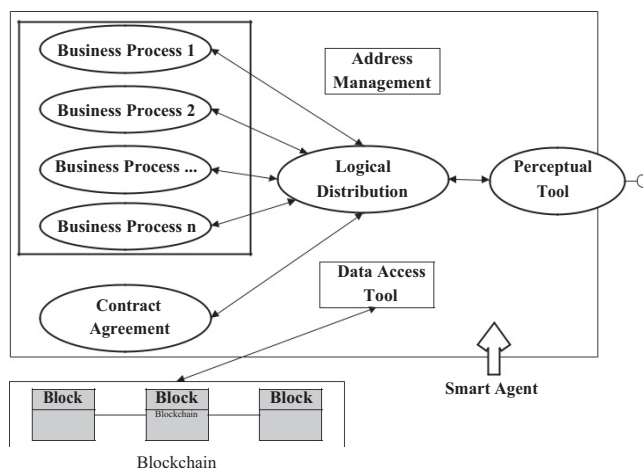


Figure 1.
Typical architecture
of a smart agent

- *Logical distribution (LD)*: It is the work manager of a smart agent and there is only one running instance in a running agent. It is mainly in charge of distributing the tasks to different business processing instances according to the sensed data and collected requirements, performing the corresponding business processes and receiving the BP results.
- *Business process (BP)*: It is the basic executive unit of a smart agent. To provide a certain service, an agent can instantiate multiple BPs simultaneously. The main functions include: receiving the message/data sent by its LD module, completing system-defined actions according to different input requirements and returning processing results to its LD module.
- *Contract agreement (CA)*: It defines the communication protocol between different agents, and only one instance is allowed in a running agent. As a contract maker for multi-party communication within a distributed system, its main task is to develop communication rules for distributed parties and make the communication process standardized.
- *Address management (AM) and data access tool (DAT)*: To enable the communication between distributed agents and exchange information with the blockchain ledger, a smart agent also requires AM and (DAT).

Based on the above design of a smart agent, the main features of a smart agent are summarized as follows.

- *Autonomy*: A smart agent has its own computing resources in BPs and behavior control mechanisms in PT and LD, which can determine and control its own behavior according to its internal state and perceived environmental information without direct external manipulation.
- *Interactivity*: A smart agent can implement flexible and diverse interactions with other smart agents (including people) in its address according to CA supported communication protocols. Thus, smart agents can effectively work together by interacting with each other.
- *Reactivity*: A smart agent can sense the environment (possibly the physical world, the user who manipulates the graphical interface or other agents) by PT and respond to the relevant events in a run-time manner via BPs.

- *Initiative*: An agent is able to behave properly so as to achieve the pre-defined goals in LD, presenting the goal-oriented initiative behavior.
- *Collaboration*: The collaboration of multiple agents can solve complex problems that are difficult or impossible to solve by a single entity. A smart agent achieves this feature through LD and CA models.
- *Distribution*: Information about the entire agent system, including data and control, is logically or physically distributed, without centralized control or global data storage. Each agent can solve problems in parallel, which improves the efficiency of a system composed of a set of smart agents.

With the above features of smart agents, an intelligent system can be realized. Next, we are to introduce a general architecture for smart agent systems.

3.2 Architecture of smart agent systems

When using smart agents to form a decentralized system, the business requirements in the system need to be designed into different BPs and communication contracts should be designed in CA according to the corresponding business. At the same time, the system environment that needs to be perceived is designed in PT, and the corresponding processing method is set in LD.

By deploying the smart agent in each node of the system network, an agent-to-agent distributed and the decentralized system is constructed. The network topology diagram is shown in Figure 2. It is noted that the blockchain network is also composed by the smart agents and each agent saves a replica of the complete blockchain ledger.

When the system network is built, the entire system can spontaneously handle the operations involved in the system at runtime.

4. Smart agent blockchain-based federated learning

In this section, a demonstration based on the above smart agent system of training FL neural network is implemented. The goal of the system is to effectively train a neural

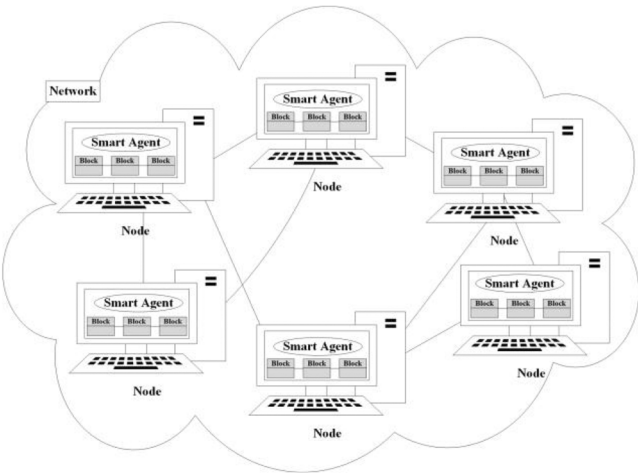


Figure 2.
Network topology of
a smart agent system

network and protect the local data in a proper manner. Intermediate gradients are stored in the blockchain, making our systems decentralized, reliable and privacy-preserving.

The neural network training system consists of smart agents where each agent is composed by six main components, namely, environmental perception (PT), local server (LD), model trainer (BP), partner control (CA), key management (AM) and chain pipe (DAT). These components correspond to a smart agent design in Section 3.

4.1 The module design

A user i is trying to join in our system, and it cannot do anything until it passes the authentication that it owns the private key Key_i in the blockchain network. After registration, a unique identity Id_i generated by Key_i is assigned to the user. From then on, once the user wants to interact with our system, its private key is the only valid credential. Next, we are going to introduce the detailed design of the main components and process procedures.

4.1.1 Environment perception. This module is equivalent to PT in our proposed framework, and it provides the smart agent with communication interfaces with external environments. External environments will send many instructions to the agent, including receiving hyperparameters and begin the training, sending final model and destroy the model. PT is responsible for listening the information from external environments and forwarding it to LD.

4.1.2 Local server. This module is equivalent to LD in our proposed framework. In a traditional FL task, we need a central server to aggregate the gradients from all participants in an *iteration*. However, in our design, the central server is substituted with a blockchain network. Besides, the LD is also in charge of gradient generation and model update procedure.

4.1.3 Model trainer. This module is equivalent to BP in our proposed framework. As a basic executive unit of the smart agent, it implements gradients generation and model update, which will be specifically described in the next section.

4.1.4 Partner control. This module is equivalent to CA in our proposed framework. Its main function is to communicate with other smart agents via generating the authentication token Aum of the intermediate gradients for the authenticated users set. It is well known that the machine learning model has commercial value. So if we want to put it on the market, the authenticated users set should have good a scalability to add or remove.

4.1.5 Key management. This module is equivalent to AM in our proposed framework. It has two functions. On one hand, it is in charge of AM where each smart agent has a pair of keys, i.e. public key Id and privacy key Key . On the other hand, a user who wants to join the FL is also required to pass the authentication to ensure the user has a valid private key in the blockchain network.

4.1.6 Chain pipe. This module is equivalent to a DAT, and it provides smart agents with two interfaces to interact with the blockchain, namely, writing and reading data from the blockchain.

4.2 The main procedure design

In this section, gradient generation protocol, block generation protocol and model update protocol are specifically designed.

The FL under study is operated by a set of data owners, we might as well call them smart agents here. Considering there is a set of smart agents $A = \{1, 2, 3, \dots, N_A\}$ with $|A| = N_A$, the i -th smart agent owns a set of data samples S_i with $|S_i| = N_i$. All data samples

$S = \cup_{i=1}^{N_A} S_i$ with $|S| = N_S$. The k -th data sample $s_k \in S$ is given as $s_k = \{x_k, y_k\}$ for a d -dimensional column vector $x_k \in \mathbb{R}^d$ and a scalar value $y_k \in \mathbb{R}$. Global model has a loss function and there are many kinds of loss functions such as cross entropy. Without loss of generality, the mean square error algorithm is chosen as the loss function:

$$l(w) = \frac{1}{N_S} \left(\sum_{i=1}^{N_A} \sum_{s_k \in S_i} \frac{1}{2} (x_k^T w - y_k)^2 \right) \quad (1)$$

The training of a federated model consists of multiple steps called *iteration*. In i -th iteration, smart agents use their local training data and their last snapshots of the global model state w_t to compute the gradient of the loss function $\nabla l(w_i, x_i, y_i)$ to upload it to the central server, then the server uses the gradients to update the global model. The global model are updated at each iteration i as follows:

$$w_{t+1} = w_t - \eta_t \left(\lambda w_t + \frac{1}{N_t} \sum_{(x_i, y_i) \in A_t} \nabla l(w_t, x_i, y_i) \right) \quad (2)$$

where η_t indicates a degrading learning rate and λ is a regularization parameter that prevents over-fitting.

The method of stochastic gradient descent (SGD) is a general learning algorithm that can be used to train various models, including deep learning, This is the reason of choosing SGD based FL in our study. If the SGD has run for a fixed number of iterations or the magnitude of the gradient has fallen below a threshold, the model training then finishes and the global model is considered as the optimal resulting model w^* .

In traditional FL, the global update is computed by the server, and in our proposed framework the server entity is replaced with a blockchain network.

4.2.1 Gradient generation procedure. After a BP receives the iteration number i from LD, as mentioned above, it fetches the local data and its last snapshot of the global model state and computes the gradient of loss function $\nabla l(w_t, x_i, y_i)$. Then, $\sum_{(x_i, y_i) \in A_t} \nabla l(w_t, x_i, y_i)$ will be

returned to associated LD and the CA will generate tokens $Aum(ID, i)$ and distribute them to the authenticated users set. The time sequence of the gradient generation protocol is shown in Figure 3.

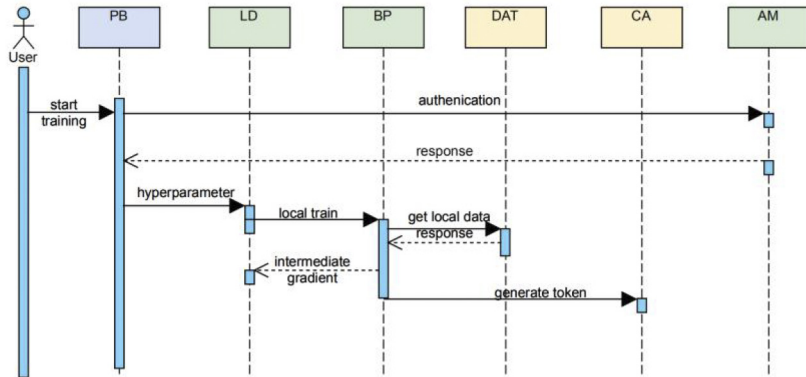


Figure 3.
Time sequence of
gradient generation
procedure

4.2.2 Block generation procedure. In our proposed SABlockFL framework, the blocks and their verification are designed so as to exchange parameters updates truthfully.

At first, the LD will use tokens issued by its and other smart agents' CA to get gradients, which will be packaged into a block. The LD has a function of mining a new block in this procedure. Each block consist of its head and body, and the body stores the gradients, as well as its digital signature. The header contains the output value of the POW, block generation rate and a pointer to the previous block.

The process of mining a block is similar to that in the BitCoin (Nakamoto *et al.*, 2008). The miner who is also the LD randomly generates a hash value by changing its input number, i.e. nonce, until hash value is smaller than a given difficulty value. Once the miner M succeeds in finding a suitable nonce, the miner then generate a new block and propagate it to all the other miners, i.e. other LD. Those miners who receive the generated block are verified and valid blocks are to be added to their local ledgers via DAT. The time sequence of the block generation is shown in Figure 4.

4.2.3 Model update procedure. As soon as the block is added to the ledger, the BP will be awakened by a signal. It will read the block from its ledger and compute equation (2) to update its local model. The time sequence of this procedure is shown in Figure 5.

The procedure will not stop until the SGD has run for a fixed number of iterations or the magnitude of the gradient has fallen below a threshold.

4.3 Implementation of smart agent blockchain-based federated learning system

We use JADE 4.5.0, which compiles with FIPA specification to simulate a multi-agent system. In the developed system, each node is equipped with six functions in the six modules correspondingly. The agent information running on each node is shown as in Figure 6.

We implement the designed framework in an Ethereum development platform to present its effectiveness. Based on the implement SABlockFL system, we are going to demonstrate how an artificial neural network is trained, which will be used to solve the classical XOR problem (Miller *et al.*, 1989).

In the PT, we implement the interface to get hyperparameters from the environment. The LD has three behaviours, namely, getting intermediate gradients from BP, packing gradients

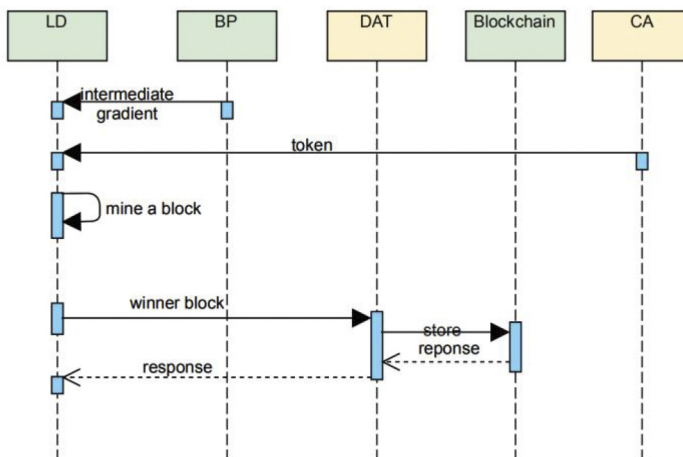


Figure 4.
Time sequence of
block generation
procedure

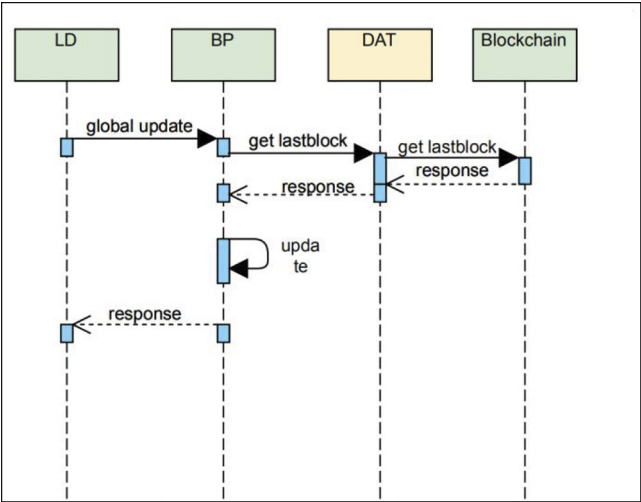


Figure 5.
Time sequence of
model update
procedure

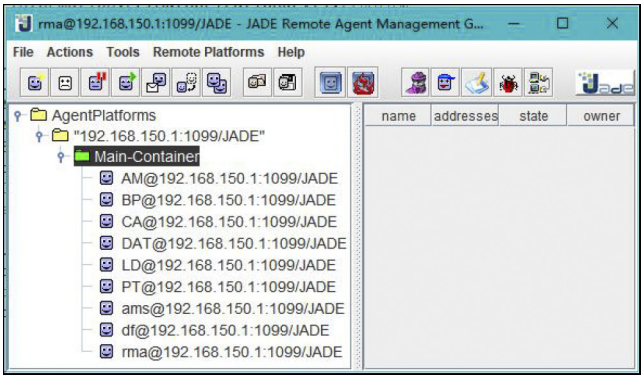


Figure 6.
Smart agent in a node

into a candidate block and mining a new block. The AM has two main functions, namely, generating identity authentication and verifying identity authentication.

We use Ganache [1], which is a private Ethereum blockchain client for developers, to simulate a local blockchain network with 10 nodes, where the rate of block generation is 1 block per second. Their accounts and private keys are as follow (Figure 7).

The parameter settings are summarized in Table 1, which are created by a FL initiator to PT and will be recorded in the genesis block.

After the system parameters are passed to LD by the PT, the smart agent proceed to pass an authentication process. Now, that our smart agents network and the blockchain network are overlapped, and each smart agent has no access to the blockchain network until it owns a private key of these nodes and passes the authentication executed by AM as shown in Figure 8.

Next, let us begin to train our neural network. The dataset is distributed among nodes. When the smart agent has been proved to be a legitimate user, its LD instantiates a BP to compute the intermediate gradient i.e. $\nabla l(w_l, x_i, y_i)$ via local data and sends it as a

```
Ganache CLI v6.7.0 (ganache-core: 2.8.0)

Available Accounts
=====
(0) 0xc1285D57B3503aed84B3D14bDB02108522fAB9fe (100 ETH)
(1) 0x03083455652F06994B6dC8889f2DE21E135a5F17 (100 ETH)
(2) 0xa31DA120e4dAA7D56358654eFBF5E76A2B61dF4D (100 ETH)
(3) 0xfa045Be8D28bac059C9947C24A07A630e993DfE5 (100 ETH)
(4) 0x021dc0A890C5C2abEE1D7F09e6Eb70bcf73972DF (100 ETH)
(5) 0x79739248EcAF0722A9b2272D000C6276fCa7dFfb (100 ETH)
(6) 0x768c781cA73EadC8D528C11Bd878749CabC55840 (100 ETH)
(7) 0x2950c3d5544180376351bF72B2337bF431e4EB91 (100 ETH)
(8) 0x9ACb43fa63cdd9691B36f8E20890d21cEBF84BD2 (100 ETH)
(9) 0x99A5D27A3c0927c585009fE3620863b7A775393B (100 ETH)

Private Keys
=====
(0) 0xd1a7f309dd8c6844a484af2ef0235f55aaf81e2c193c24b092d397c0ab008a6e
(1) 0x613b0bafedc756aad996c146db58521cfaca180cbe1049a694a6ff95c3dad9e4
(2) 0x50e06a956f3530868b0268e7ee7fd0f8d08708f9e2b8cda42fdd9c5e4f3aa28
(3) 0xf92337778caaa3321312e8ec835c56f9f5d5fadbe90a1aaf569ec7a04a3aa850
(4) 0x7161994032d0075db4b2b1f86d9b1d2ed4531499cbe68b9381ba61b3f81cbc2c
(5) 0x3bf8b64e0f5d2062b345ba8b2e1d3e363596f86105681661112b65de6ec64162
(6) 0x9ef4536073bc6ad6403a89050b65de1160f132ed37cd011a3cc546af181db7e2
(7) 0x659f89148f1766dba907a35b5dac6c94e25da46c18a4a42b9f8e1000f36418e
(8) 0x016e4b33e918cd0d272f319c7de32a52ded223cd5601ddc526ff59df67881f41
(9) 0x14563f16bb5f485a5e0a6d148f992f60cc16451dbd9c52a659bb0de5b3df2da
```

Figure 7.
Nodes in local
Ethereum stimulator
ganache

Parameter	Value	
# of neurons in input layer	2	
# of neurons in hidden layer	2	
# of neurons in output layer	2	
η	0.05	
λ	0.1	
Iteration	5,000	

Table 1.
Parameter settings in
SABlockFL

```
Console
<terminated> printer [Java Application] /opt/jdk1.8.0_25/bin/java (2019年11月28日 下午2:44:23)
PT:user begin a new blockchain-federated learning task
PT:hyperparameter are as follow:
    number of neurons in the input layer: 2
    number of neurons in the hidden layer: 5
    number of neurons in the output layer: 1
    initial learning rate: 0.05
    regularization paprameter: 0.1
    number of itrations: 0.1
AM:pass authentication!
    account:0xfa045Be8D28bac059C9947C24A07A630e993DfE5
```

Figure 8.
Smart agent receives
parameters

transaction to Ganache. For instance, the 4th node gets an intermediate gradient and sends it out, then receives a transaction receipt. This transaction process is shown in [Figure 9](#).

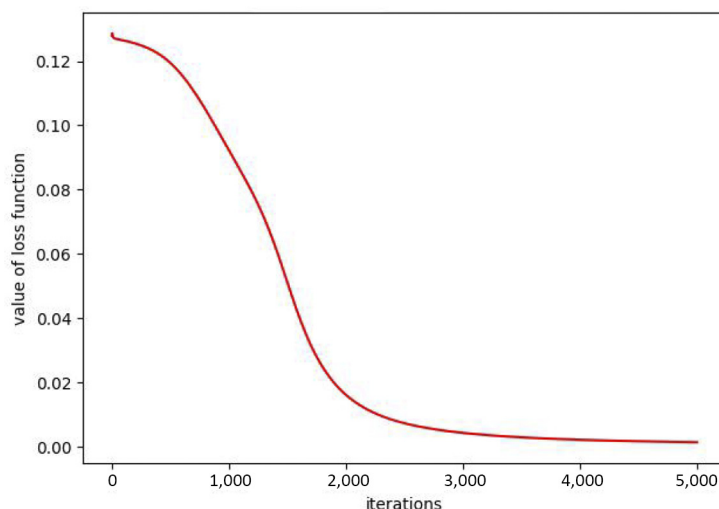
At the same time, the global model can only be accessed by those agents with legitimate private keys as justified by the training of the model. The process of accessing the global model is shown in [Figure 10](#).

Finally, let us take a look at the effectiveness of our system. After five thousand iterations, the loss function is gradually close to zero as presented in [Figure 11](#), which means the global model has converged.

Figure 9.
Fourth smart agent
sends a transaction
and receives receipt

Figure 10.
Global model
accessed by the 4th
agent

Figure 11.
Convergence of the
loss function with
iterations



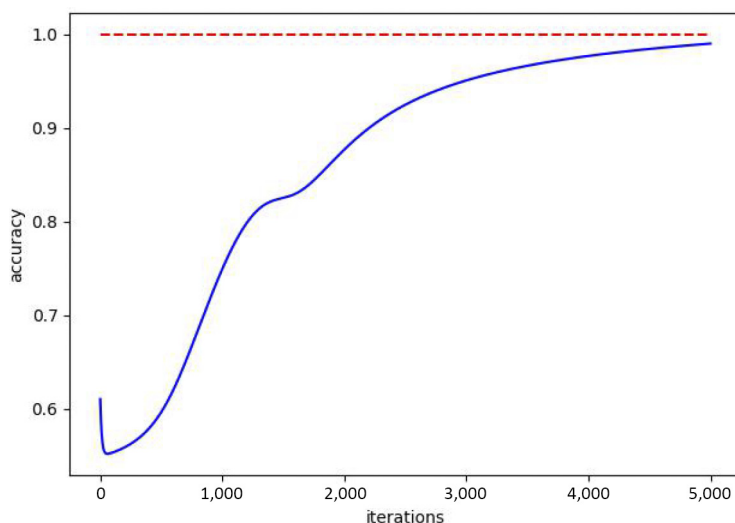


Figure 12.
Accuracy of global
model with iterations

5. Conclusion and future work

Inspired by the concept of smart contract, this work proposes a smart agent model. A smart agent can actively handle users' adaptive needs and void suffering from the over monitor/control of the third party. Based on this smart agent system, a SABlockFL framework has been proposed, which is not under the vulnerability of the central server's information leakage attack. Finally, we demonstrate the application of the proposed architecture in training a machine learning task in solving a neural network for classic XOR problem. The performance of our designed framework shows that the proposed smart agent architecture is practical and effective in developing secure distributed application.

The application of blockchain technology has brought another advantage that we can send rewards to the participants and design an incentive mechanism (Kang *et al.*, 2019) to encourage them to contribute quality data, which will be the focus of our future work. Furthermore, we also plan to evaluate the proposed framework in training extensive machine learning tasks such as MINIST and CIFAR. We also plan to use some secure multiple party computation technologies such as homomorphic encryption techniques.

Note

1. www.trufflesuite.com

References

- Accountability Act (1996), "Health insurance portability and accountability act of 1996", Public law, 104: 191.
- Balaji, P.G. and Srinivasan, D. (2010), "An introduction to multi-agent systems" *Wiley and Sons*, Vol. 4 No. 2, pp. 125-128.
- Ben-Or, M. and Hassidim, A. (2005), "Fast quantum byzantine agreement" *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, Citeseer*, pp. 481-485.

- Bond, A.H. and Gasser, L. (2014), *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, MA.
- Chilimbi, T., Suzue, Y., Apacible, J. and Kalyanaraman, K. (2014), "Project adam: building an efficient and scalable deep learning training system", *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp. 571-582.
- Crosby, M., Pattanayak, P., Verma, S. and Kalyanaraman, V. (2016), "Blockchain technology: beyond bitcoin", *Applied Innovation*, Vol. 2 Nos 6/10, p. 71.
- Delmolino, K., Arnett, M., Kosba, A., Miller, A. and Shi, E. (2016), "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab" *Proceedings of International conference on financial cryptography and data security*, pp. 79-94.
- Dragoni, N., Lostal, E., Gadyatskaya, O., Massacci, F. and Paci, F. (2011), "A load time policy checker for open multi-application smart cards" *Proceedings of IEEE International Symposium on Policies for Distributed Systems and Networks*, pp. 153-156.
- Hitaj, B., Ateniese, G. and Perez-Cruz, F. (2017), "Deep models under the gan: information leakage from collaborative deep learning" *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM*, pp. 603-618.
- Kang, J., Xiong, Z., Niyato, D., Xie, S. and Zhang, J. (2019), "Incentive mechanism for reliable federated learning: a joint optimization approach to combining reputation and contract theory" *IEEE Internet of Things Journal*, Vol. 6 No. 6, pp. 10700-10714.
- Kapitonov, A., Lonshakov, S., Krupenkin, A. and Berman, I. (2017), "Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs" *Proceedings of Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pp. 84-89.
- Kim, H. Park, J. Bennis, M. and Kim, S.-L. (2019), "Blockchained on-device federated learning", *IEEE Communications Letters*.
- Kolvart, M., Poola, M. and Rull, A. (2016), "Smart contracts" *The Future of Law and ETechologies*, Vol. 1, pp. 133-147.
- Li, T. Kumar Sahu, A. Talwalkar, A. and Smith, V. (2019), "Federated learning: challenges, methods, and future directions" arXiv preprint arXiv:1908.07873.
- Liu, Z., Luong, N.C., Wang, W., Niyato, D., Wang, P., Liang, Y.-C. and Kim, D.I. (2019), "A survey on blockchain: a game theoretical perspective" *IEEE Access*, Vol. 7, pp. 47615-47643.
- Lu, Y., Huang, X., Dai, Y., Maharjan, S. and Zhang, Y. (2019), "Blockchain and federated learning for privacy-preserved data sharing in industrial iot" *IEEE Transactions on Industrial Informatics*, Vol. 16 No. 6.
- McMahan, H.B. Moore, E. Ramage, D. and Hampson, S. (2016), "Communication-efficient learning of deep networks from decentralized data" arXiv preprint arXiv:1602.05629.
- Melis, L. Song, C. De Cristofaro, E. and Shmatikov, V. (2018), "Inference attacks against collaborative learning" arXiv preprint arXiv:1805.04049, 13.
- Miller, G.F., Todd, P.M. and Hegde, S.U. (1989), "Designing neural networks using genetic algorithms" *ICGA*, Vol. 89, pp. 379-384.
- Minsky, M. (1987), "The society of mind" *Personalist Forum*, Vol. 3 No. 1, pp. 19-32.
- Mohammed, M.A., Gunasekaran, S.S., Mostafa, S.A., Mustafa, A. and Ghani, M.K.A., (2018), "Implementing an agent-based multi-natural language anti-spam model" *Proceedings of International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)*, pp. 1-5.
- Nakamoto, S. (2008), "Bitcoin: a peer-to-peer electronic cash system".
- Shayan, M. Fung, C. Yoon, C.J.M. and Beschastnikh, I. (2018), "Biscotti: a ledger for private and secure peer-to-peer machine learning" arXiv preprint arXiv:1811.09904.
- Song, C., Ristenpart, T. and Shmatikov, V., (2017), "Machine learning models that remember too much" *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM*, pp. 587-601.

-
- Vlassis, N. (2007), *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*, Morgan and Claypool Publishers, New York, NY.
- Wood, G. (2014), "Ethereum: a secure decentralised generalised transaction ledger" *Ethereum Project Yellow Paper*, Vol. 151, pp. 1-32.
- Wooldridge, M. and Jennings, N.R. (1994), "Agent theories, architectures, and languages: a survey" *Proceedings of the International Workshop on Agent Theories, Architectures, and Languages*, pp. 1-39.
- Yang, X., Kang Yang, Y.T. and Liu, Y. (2019a), "A blockchain based smart agent system architecture" *Proceedings of the 4th International Conference on Crowd Science and Engineering*, pp. 33-39.
- Yang, Q., Liu, Y., Chen, T. and Tong, Y. (2019b), "Federated machine learning: concept and applications" *ACM Transactions on Intelligent Systems and Technology (Technology)*, Vol. 10 No. 2, pp. 1-19.
- Zhang, Y., Kasahara, S., Shen, Y., Jiang, X. and Wan, J. (2018), "Smart contract-based access control for the internet of things" *IEEE Internet of Things Journal*, Vol. 6 No. 2.
- Zhao, Y., Zhao, J., Jiang, L., Tan, R. and Niyato, D. (2019), "Mobile edge computing, blockchain and reputation-based crowdsourcing iot federated learning: a secure, decentralized and privacy-preserving system" arXiv preprint arXiv:1906.10893.
- Zhou, J., Quan Yuan, W.U., Teng, M., Wang, H.M. and Sun, H.Y. (2000), "An agent framework based on distributed object" *Journal of Computer Research and Development*, Vol. 1, pp. 188-194.

Further reading

- Bottou, L. (2010), "Large-scale machine learning with stochastic gradient descent" *Proceedings of COMPSTAT*, Springer, pp. 177-186.
- Du, M., Ma, X., Zhang, Z., Wang, X. and Chen, Q. (2017), "A review on consensus algorithm of blockchain" *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, pp. 2567-2572.

Corresponding author

Yuan Liu can be contacted at: liuyuan@swc.neu.edu.cn

For instructions on how to order reprints of this article, please visit our website:

www.emeraldgrouppublishing.com/licensing/reprints.htm

Or contact us for further details: permissions@emeraldinsight.com