KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 7 (2021) 300–307

www.elsevier.com/locate/icte

An efficient parallel machine learning-based blockchain framework

Chun-Wei Tsai*, Yi-Ping Chen, Tzu-Chieh Tang, Yu-Chen Luo

Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, ROC

Received 28 February 2021; received in revised form 22 June 2021; accepted 2 August 2021
Available online 19 August 2021

Abstract

The unlimited possibilities of machine learning have been shown in several successful reports and applications. However, how to make sure that the searched results of a machine learning system are not tampered by anyone and how to prevent the other users in the same network environment from easily getting our private data are two critical research issues when we immerse into powerful machine learning-based systems or applications. This situation is just like other modern information systems that confront security and privacy issues. The development of blockchain provides us an alternative way to address these two issues. That is why some recent studies have attempted to develop machine learning systems with blockchain technologies or to apply machine learning methods to blockchain systems. To show what the combination of blockchain and machine learning is capable of doing, in this paper, we proposed a parallel framework to find out suitable hyperparameters of deep learning in a blockchain environment by using a metaheuristic algorithm. The proposed framework also takes into account the issue of communication cost, by limiting the number of information exchanges between miners and blockchain.

Keywords: Machine learning; Blockchain; Deep learning

1. Introduction

The development of artificial intelligence (AI) [1] has undergone great changes for years in which several successful applications were also presented to the public to provide us a more convenient life today. One of the important research branches of AI technologies is certainly the so-called machine learning (ML) [2], the three representative learning technologies of which are named supervised, unsupervised, and semi-supervised, respectively. Each of these three technologies can be used alone in an intelligent system; of course, they can also be combined if needed for solving a problem in question together. As an important research direction of AI, the basic idea of machine learning is to use labeled/unlabeled input data to find out the appropriate rules to classify the yet unknown data or to forecast events in the coming future.

As shown in Fig. 1, both supervised and unsupervised have a learning procedure; the main difference is that the first input data of supervised learning are labeled data while the input data of unsupervised learning are unlabeled data. This makes the design of learning procedures for them quite different because without labeled data, unsupervised learning needs to use less information than supervised learning to classify the unlabeled data. On the other hand, supervised learning will use the labeled data to construct the model to classify the unlabeled data. That is why supervised learning can typically provide a better accuracy rate than unsupervised learning. Of course, this does not mean that unsupervised is no longer useful; on the contrary, it is very useful when there is no easy way to get the labeled data for the machine learning system.

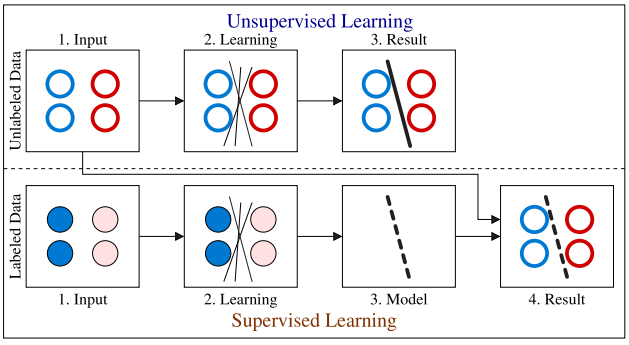

Fig. 1. Example illustrating supervised and unsupervised learning technologies to classify the unlabeled data via labeled/unlabeled data.

* Corresponding author.
E-mail address: cwtsai@mail.cse.nsysu.edu.tw (C.-W. Tsai).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2021.08.014

Since the AI and ML technologies have been applied to several modern information systems to make it possible for them to provide us smart services, a report of Gartner [3] pointed out that AI and relevant applications will create $2.9 trillion of business value in 2021 and continuously create more business values in the next five years. A later report [4] further showed that AI and machine learning technologies will strongly influence the applications around our daily life, such as agriculture, health care, space exploration, autonomous vehicles, and even artificial creativity. From these applications, it is not hard to imagine that AI and ML technologies will be used to make information systems provide us more appropriate and needed services. Although some recent news [5,6] pointed out that fames of Google DeepMind and IBM Watson did not bring these two companies large fortune in recent years, it does not mean that AI and ML technologies are falling down. It means that instead of applying AI and ML technologies to complicated systems or unknown research domains, applying them to simple systems or tasks may make it easier to show the possibility of these two technologies.

In addition to finding out feasible applications to which AI and ML technologies can be applied, how to keep intelligent systems from being attacked by abnormal users and how to protect the privacy of private and sensitive data in intelligent systems are two critical research issues today. Because machine learning and blockchain technologies are originated from two different disciplines, the design and goal of them are quite different. One important difference is in that most machine learning technologies are designed for an integrated view; that is, to combine the data and computations from one or more computers while most blockchain technologies are designed for a peer-to-peer view; that is, to divide the data and computations in different computers. This is the main reason that the parallel computing has become one of the most recent trends in the studies on combining machine learning and blockchain technologies into a single information system.

Several recent studies [7,8] have made some discussions to show that using blockchain for intelligent systems might be able to deal with the security and privacy issues they are facing. Salah et al. [7] pointed out that works on blockchain technologies for intelligent systems classify AI into five types: (1) decentralized AI applications, (2) decentralized AI operations, (3) blockchain types for AI applications, (4) decentralized infrastructure for AI applications, and (5) the role of consensus protocols for AI applications. Tanwar et al. in a later study [8] focused on discussing the integration of ML algorithms and blockchain, and they divided these integrations into four groups: goal oriented, layers oriented, counter-measures, and smart applications.

Now the question arises, how do we combine ML with blockchain technologies. That is why this paper is focused on the machine learning-based blockchain frameworks because a suitable ML framework would help us develop an efficient way to combine ML with blockchain technologies or to apply ML to blockchain systems. The main contributions of this study can, therefore, be summarized as follows:

1. It first provides a brief review on the parallel ML frameworks with blockchain technologies and applying them to blockchain systems, which is useful as a roadmap to integrate ML and blockchain.
2. It then gives a detailed description of the proposed parallel deep learning framework with blockchain the focus of which is on seeking "good" hyperparameters of deep learning by using a metaheuristic algorithm to further improve the accuracy that relies on the cooperation of all nodes.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to parallel machine learning frameworks. Section 3 begins with the basic idea of the blockchain, followed by a detailed description of the parallel machine learning-based framework with blockchain technologies or in blockchain environments. Section 4 provides descriptions and discussions of the proposed parallel deep learning framework for blockchain. Section 5 shows the experimental results of the proposed framework. Finally, Section 6 draws the open issues in this research field and gives some potential research topics of this study.

2. Parallel machine learning frameworks

Since parallel computing is an intuitive and useful way to reduce the response time of machine learning algorithms, many efficient ways to speed up the machine learning algorithms have been presented dated back to the 1980s or even earlier. Since the 1990s, how to apply machine learning algorithms to parallel computing environments has been a critical research topic. Although the definitions of early studies [9] for distributed computing and parallel computing are quite different, the boundary of them has become vague in recent studies. The main reason is that each processor (node) in distributed computing has its own memory while all processors (nodes) will access a shared memory, but some recent studies in machine learning research domain used the name parallel machine learning algorithm to describe the algorithms they proposed for which each node has its own memory. Note that as far as this paper is concerned, the term "parallel" means not only parallel computing but also "distributed" computing.

Also, because the research domain of genetic algorithm (GA) was very bustling in the 1990s, several parallel versions of GAs have then been presented in those years [10,11]. In [10], Cantú-Paz divided parallel genetic algorithms into three representative categories; namely, (1) global single-population master–slave, (2) single-population fine-grained, and (3) multiple-population coarse-grained. For the global single-population master–slave, the master node has a global population and is responsible for sending the computing tasks to the slaves for the evolution of GA, such as selection or crossover operations. This means that most operations of the GA can then be computed by more than one slave; therefore, the response time of GA can be significantly improved compared to using only a single computing node. The single-population fine-grained is typically used for distributed computing systems. The basic idea is that chromosomes and

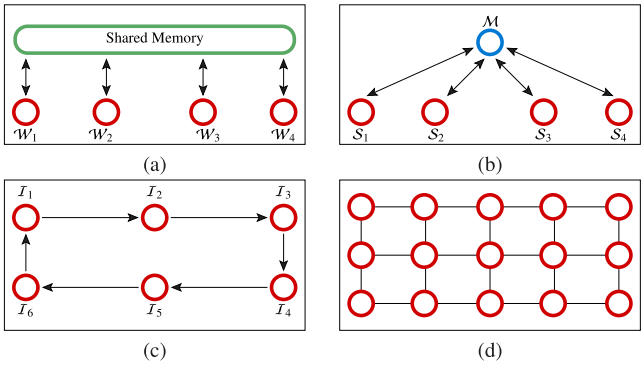

Fig. 2. Example illustrating parallel machine learning algorithms via the perspective of searched information exchange. (a) Multithreading or shared memory, (b) master–slave, (c) coarse-grained, and (d) fine-grained.

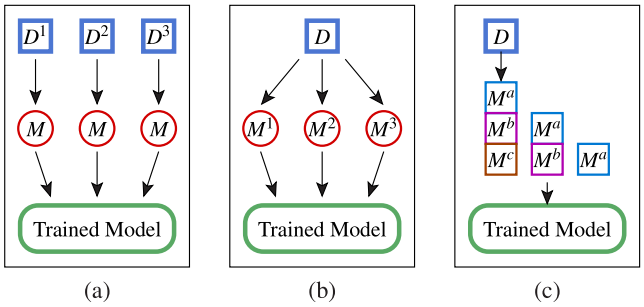

Fig. 3. Example illustrating parallel machine learning algorithms via the perspective of model construction. (a) Data parallelism, (b) model parallelism, and (c) layer pipelining.

computing tasks will be dispatched to different computing nodes while the information exchange will be restricted to the neighbor nodes. The multiple-population coarse-grained is also referred to as the island model GA because it will divide the entire population into a set of subpopulations. Each island has its own subpopulation and sends a chromosome (i.e., searched information) to another island after a certain number of generations.

These three different parallel models can also be found in different metaheuristic algorithms, such as particle swarm optimization (PSO). That is why Lalwani et al. [12] also used these three categories to divide the parallel PSOs and discussed the possibilities of PSO in other distributed computing environments, such as CPU vs. GPU or topology of PSO. Because ant colony optimization (ACO) provides an effective way for solving several combinatorial optimization problems, the study [13] based on the number of colonies and cooperation or not to divide the parallel ACO into four categories that are cellular, multicolony, master–slave, and parallel independent runs. In addition to parallel unsupervised learning algorithms (e.g., GA, PSO, and ACO); of course, master–slave, coarse-grained, and fine-grained are three typical classifications that cover most parallel machine learning algorithms.

As shown in Fig. 2, we add a new category multithreading (or shared memory) to these three categories to cover the parallelism in the same computer but using multiple threads for parallel computing. Fig. 2(a) shows that all of the threads $\mathcal{W}_i$ share the same information via shared memory. Fig. 2(b) shows that although the slave nodes $\mathcal{S}_i$ of master–slave will also share the same information via master node $\mathcal{M}$ like multithreading, the main differences are that slave nodes are controlled by master node and all the slave nodes are different. Fig. 2(c) and (d) show that each node of the coarse-grained passes its searched information to another node but each node of fine-grained passes its searched information to a set of neighbor nodes.

Several recent studies [14,15] that expounded on how to make machine learning (including deep learning) algorithms work on distributed computing environments have attracted the attention of researchers from different disciplines today. One of the main reasons is that most deep learning algorithms are computation intensive and thus time consuming; as such, how to accelerate these algorithms is related to the response time of such an intelligent system which is a main concern of users today. As shown in Fig. 3, parallel machine learning algorithms can be divided into three different categories to explain different ways for constructing learning models based on the classification described in [14,15]; namely, data parallelism, model parallelism, and layer pipelining. Fig. 3(a) shows that a simple way to achieve parallel computing for machine learning is to divide the data (e.g., D^1, D^2, and D^3) and dispatch them to different nodes. Each node will then construct and send its model to a node to integrate these models. A representative way is the so-called minibatch, which has been widely used in recent deep learning studies [14]. Fig. 3(b) shows that in model parallelism (also called network parallelism), each node will receive a copy of the entire data and construct different parts of the model M^i. After that, these different parts of model will be sent to a node to aggregate all of them to construct a complete model [15]. As shown in Fig. 3(c), the overlapping computing can also be considered in the design for parallel deep learning algorithm [14]. The partition can be the computation tasks of one layer and next layer or computation tasks of training data elements and other training data elements.

In summary, there exist many different ways to classify the parallel machine learning and deep learning algorithms. For example, Verbraeken et al. [15] classified the parallel machine learning algorithms based on topology structures into the following categories; namely, (1) ensembling, (2) tree, (3) parameter server, and (4) peer to peer. In [14], Ben-Nun and Hoefler pointed out that model consistency, parameter distributed and communication, and training distribution are three important factors to classify the distributed deep learning methods. By using these factors, we can then divide them into four categories to explain how to change the models, weights, and parameters for parallel deep learning. They are (1) synchronous and parameter server, (2) synchronous and decentralized, (3) asynchronous and parameter server, and (4) stale-synchronous, decentralized. From these classifications of parallel machine learning and deep learning algorithms, we can therefore understand how they are designed, how they

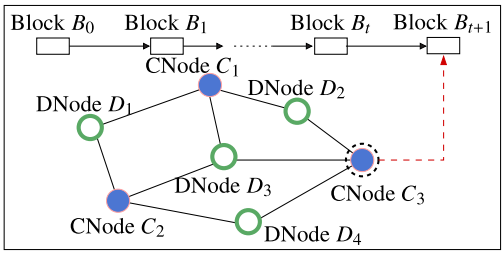

Fig. 4. Example illustrating parallel machine learning algorithm of [21] for blockchain.

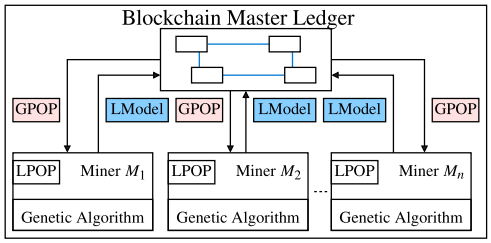

Fig. 5. Example illustrating the design of blockchain as a shared memory for machine learning algorithm in [25].

work, and how a node passes its information to the other nodes in different ways. According to our observation, the design of parallel machine learning for blockchain might fall into one or more parallel categories. In what follows, the discussion will be on some recent studies the focus of which is on combining parallel machine learning with blockchain system.

3. Parallel machine learning frameworks for blockchain

That the blockchain can provide a secure and private way to transmit messages between information systems on the internet and will drastically change the design of part of information transmission in some information systems has been pointed out in several studies [16–18]. The development of blockchain can be dated back to the bitcoin white paper published in 2008. Because it was born in that year, most internet technologies and hardware have significantly matured compared with those in the 1980s. The blockchain was designed on a distributed computing environment because the distributed "ledger" mechanism makes it possible to keep the "transactions" in more than one node. These characters make it possible for the blockchain to provide a more robust way to achieve the goals of permanent tracking, provenance history, and immutability. Of course, the design of blockchain also takes into consideration the security and privacy issues for the data (transactions) exchange and transmission in such a new network environment. Several recent studies [7,8] have shown that the two promising research topics are (1) using AI to enhance the performance of a blockchain system and (2) using blockchain to improve the security and privacy of data and model transmission on an AI system. That AI is used to improve the prediction accuracy of bitcoin value [19] can be regarded as an example of showing that AI is useful for blockchain. Another example can be found in [20], where Weng et al. used blockchain technology for the AI system to provide a secure way for a node to pass its trained AI model to other nodes in the same system. These two examples show that combining AI and blockchain has become a critical research topic in recent years.

In [21], Chen et al. presented a simple way to combine AI and blockchain. In this system, the data D will be first divided into a set of subsets of data to be distributed to k parties (also called data holders). As shown in Fig. 4, in addition to the data holders (i.e., DNode D_i), this network also contains a set of computing nodes (i.e., CNode C_i). Each data holder will first create pseudo-identity and calculate the local gradients based on the current global model while the computing nodes will compete to obtain the authority of appending a new block to the chain for solving an optimization problem. In this example, if the computing node C_3 beats the other computing nodes, it will play the role of finding the sum of gradient descent directions, calculating and updating the global gradient, and creating a new block B_{t+1} to the chain. This kind of strategy contains the characteristics of shared memory and master–slave because only one computing node C_i will update the model to the chain in each time slot but the computing node C_i is not the same in different time slot. Chen et al. in the same study also illustrated that this kind of integrated system can also provide a privacy and secure way for machine learning system.

In [22], Lu et al. used a similar way to combine federated learning and blockchain. In this study, Lu and his colleague let a participant node (data holder) construct a new learning model based on the models from other participant nodes which have similar data. All the participant nodes in the same group (not all the nodes in the network) will elect a participant node as the leader which has the best accuracy (in terms of the mean absolute error; MAE) to add the trained model to the blockchain and broadcasts this block to all the other members. Kim et al. [23] presented an architecture called blockchainedFL (BlockFL) which contains devices and miners. Each device will compute and upload the local trained model to its associated miner in this new blockchain network while each miner will exchange and verify its local model to other miners. In BlockFL, each miner will broadcast its local model to other miners to let them verify this model. In [24], Hieu et al. also let all miner nodes compete their computation power to solve a cryptographic puzzle with proof of work (PoW). This means that the winner is the miner who finds the solution first and this miner will be able to generate a new block to add to the blockchain.

As shown in Fig. 5, Mureddu et al. [25] attempted to combine genetic algorithm with blockchain to solve the scheduling problem of smart grid. In their design, each miner node m_i has its local population (LPOP). The blockchain master ledger contains a global population (GPOP), and each miner will use the global and local populations to perform the genetic algorithm. After the end of evolutionary process of each miner, the miner will add the new searched result (e.g., a scheduling solution) to the blockchain master ledger if it is better than those already saved in the master ledger.

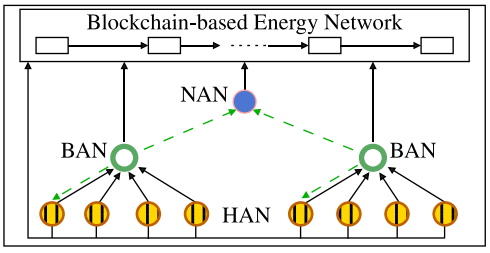

Fig. 6. Example illustrating the design of smart grid and blockchain [26].

As shown in Fig. 6, Ferrag and Maglaras [26] presented a blockchain based energy scheme to let the nodes in the smart grid network sell their excess energy to other neighboring nodes (e.g., HAN) or neighbors on the other side of the street, e.g., building area network (BAN) and neighborhood area network (NAN). The dot line in Fig. 6 represents energy coin for the consumer in order to sell/buy energy from neighboring nodes. The home area network (HAN) is at the bottom of Fig. 6; it can provide the surplus energy to the other neighborhoods if they need. This kind of strategy is similar to the shared memory parallel computing way to share the data of nodes of HAN, BAN, and NAN. Since letting all computing nodes calculate the models and upload them to the blockchain or a primary node to cooperatively construct a well-trained model is an intuitive way to combine the AI and blockchain, a similar design can also be found in [20]. The main difference of these studies is the way to exchange or aggregate the trained models from different nodes.

From the above discussions of parallel machine learning frameworks, it is now much easier to understand how they divide the data into different participant nodes and how they exchange the trained models on blockchain environments. It can be easily seen from recent studies [21–26] that the integration of machine learning and blockchain technologies has its limitations and restrictions that can be summarized as follows:

- Using Byzantine mechanism, miner nodes have to be verified by no less than 51% of all the nodes in the network before the information they have can be appended to the blockchain. This incurs a lot of additional computation and communication costs and thus will degrade the performance of miner nodes in solving an optimization problem.
- When attempting to apply blockchain and machine learning technologies to devices in an internet of thing environment, the computing restrictions of such devices need to be taken into account.
- Because the communications between miners and blockchain or between miners involve quite a lot of waiting, verification, and computation, a parallel machine learning framework for such an environment needs to reduce the frequency of communications between miners and blockchain or between miners.

According to our observation, dividing the data into a set of data holders and considering the blockchain as a common memory/database for all the miners are usually used for most studies in this research domain. How to choose the miner that can add the searched information to the blockchain is a critical issue in this research domain. Letting all the miners compete in terms of the accuracy of trained model or the response time of the miner to find the hash value is a typical way in such a system. In summary, since the parallel machine learning for blockchain is still at an early stage of development, there are still a lot of possibilities to develop much better ways to enhance the performance of such a system.

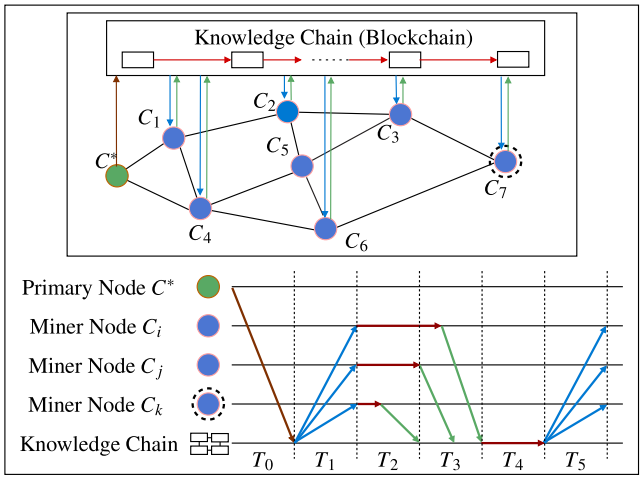

Fig. 7. Example illustrating the design of the proposed framework in different time slots.

4. A parallel deep learning framework for blockchain

Several integrated studies on combining the machine learning and blockchain make it possible for every data holder or computing node to send the request to all the nodes in the same network to train an integrated model together. However, in some cases, we still need a primary node to make a learning plan to find out a "good trained model" or "applicable hyperparameters". For this reason, we present a new parallel deep learning framework, called parallel deep learning with knowledge chain (PDLKC).

As shown in Fig. 7, in time slot T_0, the primary node C^* will first send the data and a set of training tasks (e.g., a set of learning rates) to blockchain, also called the knowledge chain in this paper. After that, all the miner nodes will receive the data and training tasks from the knowledge chain, in time slot T_1. In time slots T_2 and T_3, all the miner nodes in the network will attempt to train the models with the hyperparameters from the primary nodes via knowledge chain and randomly select one of the training tasks. This means that all the miner nodes will use different parameter settings with the same data to train the model. In this way, it will not have more than one miner node to train the models by using the same parameter value. After that, all the miner nodes will use their trained model, results, and settings of hyperparameters to update the knowledge chain. In time slot T_4, the knowledge chain will get the up to date trained model. After getting all the results

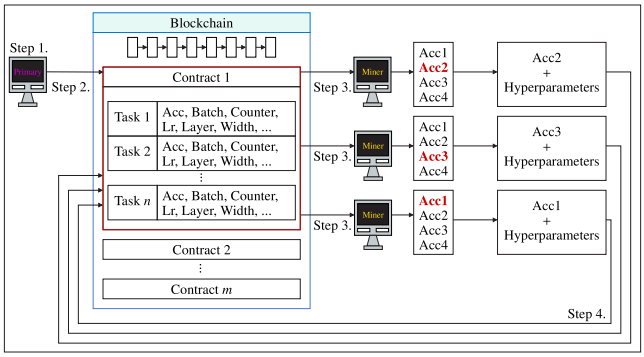

Fig. 8. A simple example illustrating the design of proposed framework.

from miner nodes, a competition will be held between these results. The best result in this round will be used to create the new computation tasks in the next round for all miner nodes. In time slot T_5, after the knowledge chain gets the new trained model and results, it can send another training tasks to all miner nodes to continue finding out better ways for classification or prediction problems. It can be easily seen that the proposed framework makes it possible for the miner nodes to receive the new tasks from the knowledge chain to further find better models and hyperparameters. In addition to finding out good hyperparameters, this framework can also be used for neural architecture search problems.

Fig. 8 gives a simple example to explain how the proposed framework works. In this example, a metaheuristic algorithm (simulated annealing; SA) is used to find a suitable set of hyperparameters for a deep neural network (DNN). Initially, in step 1, the primary node will first initialize the training tasks (e.g., different batch sizes, counters, learning rates, number of layers, number of neurons in each layer) and upload them to blockchain in step 2. These training tasks can be regarded as the starting point $s = \{s_1, s_2, \ldots, s_n\}$ of SA. In the ideal case, in step 3, all the miner nodes will then download their computation tasks from the blockchain to further train the model for the same data with different hyperparameters. Each miner node will first train the model with the starting point s_i and then train the model with s_i' generated by the transition operator from s_i. After a certain number of training processes, in step 4, each miner node will upload the hyperparameter settings that give it the best results to blockchain. This kind of design is used to reduce the number of communications between miner nodes and blockchain. After the blockchain gets the hyperparameters and accuracy results from all miner nodes, it will then update the best model, hyperparameters, and accuracy results on the chain and also use them to generate new computation tasks, i.e., starting points for the metaheuristic algorithm.

5. Simulation results

To evaluate the performance of the proposed framework (PDLKC), the MNIST dataset (http://yann.lecun.com/exdb/mnist/) is used to train the classification model of DNN. The grid search (GS) and simulated annealing (SA) [27] are used for finding suitable hyperparameters for DNN; that is, for solving the hyperparameter optimization problem [28]. Both GS and the proposed framework were implemented in Python and Ethereum (web3 5.20.0 and solidity 0.5.0) was used as the blockchain. Three PCs running Ubuntu 18.04.5 LTS are used in the simulation and the programs of the miner nodes are written in Python 3.6.9. As shown in Table 1, these three PCs have different CPUs, GPUs, and sizes of memory, and even run different versions of TensorFlow.

Table 1
The environments of miner nodes.

Miner	CPU	Memory
M1	AMD Ryzen 5 3600X 6-Core Processor	32 GB
M2	Intel Core i9-9900X CPU(3.60 GHz)	32 GB
M3	Intel Core i9-9820X CPU(3.30 GHz)	128 GB
Miner	**GPU**	**TensorFlow**
M1	NVIDIA GeForce RTX 3080 and CUDA 11.1	2.4.0
M2	NVIDIA GeForce RTX 2080 Ti and CUDA 11.2	2.3.0
M3	NVIDIA GeForce RTX 2080 Ti and CUDA 11.2	2.4.1

Table 2
Comparison of grid search, SA, and the proposed framework on centralized and distributed environments in terms of time (in seconds).

	Centralized computing				Distributed computing	
GPU	RTX 2080 Ti		RTX 3080		RTX 2080 Ti + RTX 3080	
Method	GS	SA	GS	SA	GS	PDLKC(SA)
Training	14,387	5,447	7,992	4,072	2,955	1,514
Total	14,387	5,447	7,992	4,072	5,254	2,521

For the grid search, the hyperparameter settings are as follows: the learning rate is set equal to 0.0001, 0.0005, 0.001, 0.005, and 0.01; the batch size to 16, 96, 176, and 256; the number of hidden layers to 1, 3, 5, 7, and 9; the number of neurons in each hidden layer to 16, 181, 346, and 512, and the number of epochs to 10; For SA, the hyperparameter settings are as follows: the learning rate is set equal to a number in $[0.0001, 0.01]$; the batch size to a number in $[16, 256]$; the number of hidden layers to a number in $[1, 9]$; and the number of neurons in each hidden layer to a number in $[16, 512]$; and the number of epochs to 10. For the proposed framework, all the parameter settings are exactly the same as with SA except that each miner node will first train a model via the solution s_i received from blockchain and then train another three models via s_i', s_i'', and s_i''' that are transited from s_i, s_i', and s_i'', respectively.

Table 2 shows the computation time of GS and SA on a single PC (i.e., centralized computing) as well as GS and the proposed framework on a multiple PCs with blockchain (i.e., distributed computing) environments. Note that the proposed framework with SA is called PDLKC(SA) in this paper. This comparison shows that the accuracy rates of GS and SA are 98.34% and 98.28%, respectively. In this comparison, GS and SA are first run on a single PC with different kind of GPU; namely, NVIDIA GeForce RTX 2080 Ti and NVIDIA GeForce RTX 3080. It can be easily seen that GS takes a longer training time than SA on these two GPUs. It can also be seen that GS and SA take a shorter training time on NVIDIA

GeForce RTX 3080 than on NVIDIA GeForce RTX 2080 Ti. For the distributed computing environment, we applied GS and PDLKC(SA) to the blockchain environment that contains three miner nodes, each of which has a different computing power. The simulation results show that the proposed framework not only takes a shorter training time but also runs faster than GS in a blockchain environment. Our observation shows that this is due to the fact that the miner nodes of the proposed framework do not upload the results to blockchain very frequently. This means that when each miner node gets a solution s (i.e., a set of hyperparameters) from blockchain, it will generate a few solutions based on the solution s to train the DNN models. Once a set of models are obtained, each miner node will upload one of the best solutions to the blockchain. In this way, the communication costs between miner nodes and the blockchain as well as the waiting time caused by other miner nodes can be reduced.

It can be easily seen from this comparison the possibilities of applying ML to a blockchain environment to speed up its response time. The proposed framework can be used to integrate both supervised and unsupervised learning methods for solving an optimization problem. Due to the restrictions in the implementation of Ethereum, there are two limitations on the proposed framework. One is that all the parameters (e.g., hyperparameters) have to be encoded as an integer number while the other is that the blockchain environment cannot handle too many computation tasks (or transactions). In summary, the strength of the proposed framework is that it provides a possible way to integrate ML and blockchain while at the same time reducing the training time of ML. The weaknesses of the proposed framework is that still some unnecessary waiting times incur among the miner nodes.

6. Conclusion

In this paper, we presented a simple but useful parallel deep learning framework for blockchain environment. The proposed framework inherits the characteristic of blockchain, and it can provide a secure way to get the trained model because the model on the knowledge chain needs to be verified by most miner nodes. However, the proposed framework also confronts the same research challenges that also occurred in other parallel machine frameworks with blockchain environment that are (1) additional communication costs to transfer the data, models, and parameters and (2) some unnecessary waiting for the synchronization and communication. In the future, we will attempt to design better ways to solve the above open issues to further enhance the performance of the proposed framework.

CRediT authorship contribution statement

Chun-Wei Tsai: Conception and design of study, Drafting the manuscript. **Yi-Ping Chen:** Conception and design of study, Drafting the manuscript. **Tzu-Chieh Tang:** Conception and design of study, Implementation and acquisition of data. **Yu-Chen Luo:** Conception and design of study, Implementation and acquisition of data.

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

The authors would like to thank the editor and anonymous reviewers for their valuable comments and suggestions on the paper. This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST108-2221-E-005-021-MY3 and MOST110-2218-E-110-007-MBK.

References

[1] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, third ed., Prentice Hall Press, USA, 2009.

[2] E. Alpaydin, Introduction to Machine Learning, second ed., The MIT Press, 2010.

[3] Gartner, Gartner says AI augmentation will create $2.9 trillion of business value in 2021, 2019, [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-08-05-gartner-says-ai-augmentation-will-create-2point9-trillion-of-business-value-in-2021.

[4] A. Johari, AI applications: Top 10 real world artificial intelligence applications, 2020, [Online]. Available: https://www.edureka.co/blog/artificial-intelligence-applications/.

[5] G. Marcus, DeepMind's losses and the future of artificial intelligence, 2019, [Online]. Available: https://www.wired.com/story/deepminds-losses-future-artificial-intelligence/.

[6] L. Plummer, Why isn't IBM's watson supercomputer making money? 2017, [Online]. Available: https://www.wired.co.uk/article/ibm-watson-supercomputer-profit.

[7] K. Salah, M.H.U. Rehman, N. Nizamuddin, A. Al-Fuqaha, Blockchain for AI: Review and open research challenges, IEEE Access 7 (2019) 10127–10149.

[8] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P.K. Singh, W. Hong, Machine learning adoption in blockchain-based smart applications: The challenges, and a way forward, IEEE Access 8 (2020) 474–488.

[9] Wikipedia, Distributed computing, 2020, [Online]. Available: https://en.wikipedia.org/wiki/Distributed_computing.

[10] E. Cantú-Paz, A survey of parallel genetic algorithms, Calc. Paralleles Reseaux Syst. Repartis 10 (2) (1998) 141–171.

[11] E. Alba, J.M. Troya, A survey of parallel distributed genetic algorithms, Complexity 4 (4) (1999) 31–52.

[12] S. Lalwani, H. Sharma, S.C. Satapathy, K. Deep, J.C. Bansal, A survey on parallel particle swarm optimization algorithms, Arab. J. Sci. Eng. 44 (4) (2019) 2899–2923.

[13] M. Pedemonte, S. Nesmachnow, H. Cancela, A survey on parallel ant colony optimization, Appl. Soft Comput. 11 (8) (2011) 5181–5197.

[14] T. Ben-Nun, T. Hoefler, Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, ACM Comput. Surv. 52 (4) (2019) 1–43.

[15] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, J.S. Rellermeyer, A survey on distributed machine learning, ACM Comput. Surv. 53 (2) (2020) 1–33.

[16] J.-H. Lee, M. Pilkington, How the blockchain revolution will reshape the consumer electronics industry, IEEE Consum. Electron. Mag. 6 (3) (2017) 19–23.

[17] Y. Liu, F.R. Yu, X. Li, H. Ji, V.C. Leung, Blockchain and machine learning for communications and networking systems, IEEE Commun. Surv. Tutor. 22 (2) (2020) 1392–1431.

[18] J. Polge, J. Robert, Y. Le Traon, Permissioned blockchain frameworks in the industry: A comparison, ICT Express 7 (2) (2021) 229–233.

[19] W. Gao, C. Su, Analysis of earnings forecast of blockchain financial products based on particle swarm optimization, J. Comput. Appl. Math. 372 (2020) 1–14.

[20] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, W. Luo, DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive, IEEE Trans. Dependable Secure Comput. (2020) 1–18, http://dx.doi.org/10.1109/TDSC.2019.2952332, In Press.

[21] X. Chen, J. Ji, C. Luo, W. Liao, P. Li, When machine learning meets blockchain: A decentralized, privacy-preserving and secure design, in: Proceeding of IEEE International Conference on Big Data Big Data, 2018, pp. 1178–1187.

[22] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Blockchain and federated learning for privacy-preserved data sharing in industrial IoT, IEEE Trans. Ind. Inf. 16 (6) (2019) 4177–4186.

[23] H. Kim, J. Park, M. Bennis, S.-L. Kim, Blockchained on-device federated learning, IEEE Commun. Lett. 24 (6) (2019) 1279–1283.

[24] N.Q. Hieu, T.T. Anh, N.C. Luong, D. Niyato, D.I. Kim, E. Elmroth, Resource management for blockchain-enabled federated learning: A deep reinforcement learning approach, 2020, pp. 1–5, arXiv preprint arXiv:2004.04104.

[25] M. Mureddu, E. Ghiani, F. Pilo, Smart grid optimization with blockchain based decentralized genetic algorithm, in: Proceeding of IEEE Power & Energy Society General Meeting PESGM, 2020, pp. 1–5.

[26] M.A. Ferrag, L. Maglaras, Deepcoin: A novel deep learning and blockchain-based energy exchange framework for smart grids, IEEE Trans. Eng. Manage. 67 (4) (2019) 1285–1297.

[27] C.-W. Tsai, C.-H. Hsia, S.-J. Yang, S.-J. Liu, Z.-Y. Fang, Optimizing hyperparameters of deep learning in predicting bus passengers based on simulated annealing, Appl. Soft Comput. 88 (2020) 1–9.

[28] M. Feurer, J.T. Springenberg, F. Hutter, Initializing bayesian hyperparameter optimization via meta-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2015, pp. 1128–1135.