Hindawi
Complexity
Volume 2021, Article ID 7638394, 14 pages
https://doi.org/10.1155/2021/7638394

Research Article

A Clonal Selection Optimization System for Multiparty Secure Computing

Minyu Shi,[1] **Yongting Zhang**,[1] **Huanhuan Wang**,[1] **Junfeng Hu**,[2] **and Xiang Wu**[1]

[1]*School of Medical Information and Engineering, Xuzhou Medical University, Xuzhou 221000, Jiangsu, China*
[2]*Xuzhou Medical University, Xuzhou 221000, Jiangsu, China*

Correspondence should be addressed to Xiang Wu; wuxiang@xzhmu.edu.cn

Received 23 April 2021; Accepted 19 June 2021; Published 9 July 2021

Academic Editor: Zhihan Lv

The innovation of the deep learning modeling scheme plays an important role in promoting the research of complex problems handled with artificial intelligence in smart cities and the development of the next generation of information technology. With the widespread use of smart interactive devices and systems, the exponential growth of data volume and the complex modeling requirements increase the difficulty of deep learning modeling, and the classical centralized deep learning modeling scheme has encountered bottlenecks in the improvement of model performance and the diversification of smart application scenarios. The parallel processing system in deep learning links the virtual information space with the physical world, although the distributed deep learning research has become a crucial concern with its unique advantages in training efficiency, and improving the availability of trained models and preventing privacy disclosure are still the main challenges faced by related research. To address these above issues in distributed deep learning, this research developed a clonal selective optimization system based on the federated learning framework for the model training process involving large-scale data. This system adopts the heuristic clonal selective strategy in local model optimization and optimizes the effect of federated training. First of all, this process enhances the adaptability and robustness of the federated learning scheme and improves the modeling performance and training efficiency. Furthermore, this research attempts to improve the privacy security defense capability of the federated learning scheme for big data through differential privacy preprocessing. The simulation results show that the proposed clonal selection optimization system based on federated learning has significant optimization ability on model basic performance, stability, and privacy.

1. Introduction

With the widespread application of artificial intelligence in dealing with the complex problems in the smart cities construction, deep modeling and simulation research have become a bridge connecting physical equipment and virtual information space. The related research of deep learning modeling enhances data openness and social involvement in all kinds of scientific research and uses complex scientific theoretical results and big data to modify model parameters, so that the model can better adapt to the simulation requirements of specific fields. Generally, the current deep learning schemes mostly target the centralized model training process, but these schemes usually have limitations in terms of model training efficiency and data utilization. This also means that the centralized models trained by these schemes cannot meet the needs of high-throughput big data calculation, which limits the degree of data sharing and the wide application of deep learning in the construction of smart cities and leads to the emergence of "data islands." Furthermore, with numerous published studies on data privacy disclosure and regulations such as the General Data Protection Regulation (GDPR), the privacy of data has been increasingly valued [1].

For the deep learning modeling and simulation process, there are many challenges in improving the global optimality of model parameters convergence and training efficiency. First of all, model optimization is the core of the deep learning process, and the function of the optimization algorithm is to minimize the value of loss function by some

training methods. In the deep learning process, the computer learns the mapping relationship $f_\theta: x \longrightarrow y$ between the input feature x and the output feature y through the sample set and predicts the possible output value for the new input value based on this relationship. The gradient descent algorithm is an iterative method used to solve least squares problems, and it is also the most commonly used method for solving nonconstrained optimization problems such as deep learning model parameters [2]. Classical batch gradient descent algorithms [3] usually use a large amount of calculation, and each iteration takes a long time. The stochastic gradient descent algorithm solves this problem to a certain extent [4]. To neutralize the limitations of these two algorithms, a small batch gradient descent algorithm is proposed [5]. In addition to the gradient descent algorithm, the genetic algorithm and its derivative algorithms also have many applications in data mining and model optimization [6, 7]. These types of algorithms realize randomized search by referring to the related principles of biological genetic and immunology. In the artificial immune algorithm, the selection process is used as the meritocratic screening mechanism to select the better individuals, and the mutation process is that the excellent individuals learn from each other and then adjust the original population according to a certain probability.

From the perspective of data management and mining, the expansion of the dataset size and the increase in the complexity of the data structure also bring challenges to deep learning. Due to the limited computing ability of a single device, the overall training time is greatly extended. Parallel strategies in distributed deep learning are mainly divided into data parallelism [8] and model parallelism [9]. In addition, some secure computing schemes for multiparty participation scenarios are also proposed [10]. The purpose of these schemes is to allow independent data owners to perform collaborative computing without trusting each other and relying on third parties. In order to reduce the limitation of a large amount of training data collection, improve the privacy security of the transmission process, and optimize the distributed learning relationship between various computing devices, the federated learning framework is proposed [11]. Federated learning is characterized by aggregating local model parameters generated from multiple mobile devices instead of local data for collaborative deep learning model training. In the process of distributed computing, local nodes only upload model parameters (such as gradient) instead of uploading original data to preserve user privacy. According to different application scenarios, horizontal federated learning [12] and vertical federated learning [13] were proposed separately. Horizontal federated learning is usually suitable for application scenarios with different samples but similar processing services, while vertical federated learning is more suitable for application scenarios with similar user groups but different processing services. In addition, federated transfer learning [14] is also proposed to complete the common modeling task under the condition that there is almost no overlap between the data characteristics and the target user groups of the participants. In these schemes, the upload of parameters replace the upload of data, which greatly improves the privacy security of data, but the performance of the trained model is lower than that of local training, which has also become an important factor restricting the development of distributed deep learning.

Deep learning modeling provides important conditions for computer simulation of complex problems in smart cities. The modeling process utilizes real data to train model parameters and adopts complex scientific theories and technologies to modify and optimize the model. In general, artificial intelligence technology and data science research have made rapid progress in recent years, but the effectiveness and privacy of existing deep learning schemes still need to be further optimized. First of all, classical batch gradient descent algorithms calculate the gradient for the whole dataset and update the model weight accordingly, which makes these algorithms slow to update and difficult to deal with huge datasets and leads to the waste of device memory resources. Second, since the setting of hyperparameters has a significant impact on the model training effect, the selection of hyperparameters brings greater difficulties to users. Third, in the existing deep learning modeling methods, the parameter update direction completely depends on the features extracted from the batch data, and these methods may make the convergence process of the model fall into the local optimum. In addition, the centralized model training method is difficult to adapt to large-scale application scenarios [15]. Aiming at the problems of big data processing and deep learning, some studies have proposed distributed learning methods suitable for general scenarios, but the performance of the trained models is poor and it is difficult to ensure information privacy in the training process.

This research introduces a novel federated learning scheme, which enables multiple clients to train the model collaboratively under a central server. At the same time, this strategy also realizes the decentralization of training data, which can reduce some systemic privacy risks brought by traditional centralized deep learning and data analysis methods. As the research background shows, the efficiency of the calculation process and the privacy of data using are key issues in the field of data mining. On the one hand, this research makes efforts to improve the performance of the model constructed by the federated learning scheme. For example, the strategy of clonal selection algorithm is used to optimize the gradient descent process to improve the global optimality of node model and then improve the quality of federally built model after distributed training update. On the other hand, on the basis of federated learning framework, the differential privacy mechanism improves the privacy security of the system and provides conditions for the maximization of data utilization.

This research explores and improves the distributed deep learning modeling scheme from three perspectives of model basic performance, operability, and privacy security. The research aims to meet the needs of large-scale data processing and multiparty computation. Using federated learning to improve the privacy security and communication efficiency in the distributed modeling process, the clonal

selection strategy to deal with the multiobjective optimization problem is introduced, combining the differential privacy mechanism to balance the efficiency and privacy of the data modeling process.

This study aims to explore the self-adaptability of the gradient descent process, the privacy security of the multiparty cooperative computing process, and the system coordination of the federated computing scheme. The main contributions are as follows:

(1) Propose an optimization mechanism combining the clonal selection principle of artificial immune algorithm with gradient descent algorithm. Avoid the convergence process of the model falling into local optimum and improve the performance of the client node models.

(2) Propose a stronger technical support for the privacy preserving of data and model parameters, and provide a novel method to avoid privacy disclosure for high-throughput big data modeling.

(3) Design a deep modeling scheme suitable for big data distributed computing tasks. This scheme improves the usability and privacy of the global model on the basis of the federated learning scheme.

This paper consists of five parts. The introduction explains the research significance and main contributions of this paper. The related works explain the concepts and principles involved in this paper. The third part describes the various operating processes of the system. In the experiment part, the performance of the system is tested through simulation experiment analysis. The conclusion part summarizes the full text and plans for future research directions.

2. Related Works

2.1. Data Privacy and Distributed Training Methods. As an important virtual simulation method for artificial intelligence to deal with complex problems in smart cities, deep learning training optimizes model parameters through input of large volume data and labels. However, it is difficult to improve the performance of the trained model and store the related parameters when the model training needs to deal with huge scale data. In addition, when the trained model contains more than hundreds of millions of features and the weight data of the model neural network are expanded to a high-throughput scale, it will be difficult for a single device to store relevant data. At the same time, due to the strong logical connection between various parameters, the distributed storage of data is also facing great difficulties. In order to achieve distributed management of data and collaborative training of the model, some studies such as [16] proposed the model training schemes for large-scale distributed systems. Meanwhile, the training process involves the collection and storage of a large amount of data, direct access and centralized collection of these data poses hidden dangers to privacy security.

The central server in the above distributed training schemes has the right to directly access the local data. In order to preserve the privacy of local data, the central server in decentralized distributed computing scheme only provides the communication function for the asynchronous computing process. The decentralized learning framework can effectively realize the asynchronous update of multiple computing node models while also preserving data privacy to a certain extent. However, the training performance of models in these methods and the communication efficiency between participants still need to be improved, and privacy security needs to be further strengthened. Simultaneously, as major companies pay more attention to data security and user privacy, emphasis on improving the data privacy security of the training process has become an important part of artificial intelligence research on a global scale.

In addition to the above-mentioned model training methods for preserving privacy, some data anonymization methods, access control technologies, and data noise adding mechanisms have been proposed. Data anonymization methods such as K-Anonymity [17] usually directly obfuscate names and user IDs, and access control technology sets data access qualifications through restrictions on personal identity and other information. However, many studies and related cases show that the above privacy preserving methods still have hidden dangers of potential attacks [18]. The differential privacy noise mechanism is a relatively novel data processing mechanism, which can effectively avoid background attacks and link attacks that may be suffered in previous methods. Differential privacy [19] mechanism is used to preserve sensitive information during data analysis and release process, suppose there is a privacy query function a_p, the query object of this function may correspond to the cumulative value, maximum value, or mean value and gradient of the dataset. For any output about the two adjacent datasets S and S', the domain is $A(y)$ and the range is $B(y)$. Through the mapping of this function, the attacker cannot obtain valid privacy information from the output associated with these two datasets, and the function is said to satisfy the differential privacy mechanism. Expressed by the formula such that the smaller the privacy budget ε, the higher the degree of privacy preserving, it also represents the increase of noise and the decrease of data availability.

$$\forall t \in B(y), S \approx S' : \frac{\Pr[a(S) = t]}{\Pr[a(S') = t]} \leq e^{\varepsilon}. \tag{1}$$

Distributed computing is mainly achieved through parallel processing strategies, and this type of processing method is mainly divided into two implementation methods: data parallelization method [20] and model parallelization training method [21]. The model parallel training process distributes the model parameters to different computing participants, and then each participant updates its assigned parameters. Since parallel modeling requires frequent interactions between all participants, this modeling process is relatively difficult to implement. The data parallel training method refers to splitting the training data, using multiple model instances at the same time, and using multiple split datasets for parallel training. The data parallel method is relatively easy to implement but requires higher

level of communication ability. Jain P et al. [22] proposed a stochastic gradient descent algorithm for data parallelism (parallelized stochastic gradient descent). In the process of parallel training, the training processes of multiple nodes are independent of each other. The training results and parameter update information $\nabla\omega_t$ need to be reported to parameter server (Algorithm 1).

Distributed secure multiparty computing is a novel research field of deep learning. First of all, compared with parallel learning, secure multiparty computing achieves higher privacy security through computing strategies and protocols. Secondly, just as secure multiparty computing schemes usually assume that there is no cost for the internal calculation of the node, these schemes actually face higher communication complexity, and the complexity of the algorithm is measured by the amount of communication between the nodes. However, the design of this type of calculation scheme needs to fully consider issues related to communication costs and system heterogeneity.

2.2. Gradient Descent and Model Optimization. In the process of deep learning, the optimization process of the model is usually converted into the optimization process of the loss function, and the optimization result of the loss function depends on the gradient descent process of the function, so different types of gradient descent algorithms are generally used to achieve this process as needed. The most classic gradient descent methods are batch gradient descent, these methods calculate the gradient of the entire dataset while the parameters are updated only once in the iteration process corresponding to each dataset. Therefore, this method slows down when processing large datasets, and the demand for space has increased significantly. The batch size setting plays an important role in the optimization process, it is related to statistical accuracy and hardware efficiency, and the batch size of data should generally not be too small or too large. Among the many gradient descent algorithms, stochastic gradient descent (SGD) [23] is the most common one (can be regarded as a special case of gradient descent minibatch = 1) at the present stage, it uses a sample (x, y) to update the parameters at each iteration, and the loss function is

$$F_l(\theta_0, \theta_1) = \sum_{i=1}^{m} \left(D_\theta(x) - y\right)^2. \tag{2}$$

And the gradient corresponding is

$$\frac{F_l(\theta_0, \theta_1)}{\theta_j} = \sum_{i=1}^{m} \left(D_\theta(x_i) - y_i\right)^2 x_j^i. \tag{3}$$

The update amount of model parameters is

$$\theta_j = \theta_j - \alpha\left(D_\theta\left(x^i\right) - y^i\right)x_j^i. \tag{4}$$

Since the algorithm randomly optimizes the loss function on a batch of training data in each round of iteration, the update speed of each round of parameters is greatly accelerated. In addition, the SGD algorithm introduces the noise in the dataset during the learning process, which improves the generalization error. However, the stochastic gradient descent algorithm cannot perform vectorization calculations in a sample, which results in a slower learning process. And because a single sample cannot represent the trend of the entire sample, the stochastic gradient descent process is prone to encounter local minimums or saddle points. In addition, stochastic gradient descent algorithm usually involves many iterations, and the search process of the algorithm in the solution space is relatively lacking rationality. The minibatch gradient descent algorithm [24] is a compromise method of batch gradient descent and stochastic gradient descent.

2.3. Adaptive Clonal Selection Strategy. Clonal selection algorithm (CSA) is an optimized method of artificial immune system (AIS) algorithm. This type of algorithm is search algorithm with the iterative process of generating and testing, which can increase the diversity of the population and achieve global optimal convergence. In the previous research, many researchers have proposed many schemes for the adaptation of the learning rate in the gradient descent process [25].

According to the principle of clonal selection [26], similar to general genetic algorithms, the clonal selection strategy simulates the evolutionary process of artificial populations. In the immune process of biological cells, B cells continue to divide, and daughter cells change on the basis of their parents to better match the antigens. In the process of algorithm optimization, the problem is usually defined as an antigen, and the solution to the problem is a collection of antibodies. In a specific morphological space, randomly generated antibodies will try to match the antigen, and this process corresponds to the process of the algorithm trying to solve the problem. The fitness value represents the degree to which the individual is close to the optimal solution, and the higher the fitness is, the more likely it is to be inherited to offspring. This algorithm also represents an evolutionary strategy that can solve complex deep learning tasks.

The clonal selection algorithm directly operates on structural objects, without the limitation of derivation and function continuity, and has inherent implicit parallelism and better global optimization capabilities. In addition, the algorithm adopts a probabilistic optimization method, which can automatically obtain and guide the optimized search space without determining rules and adjust the search direction adaptively.

The clonal selection algorithm is usually used to deal with pattern recognition and multimode optimization problems, including three operators of cloning, selection, and mutation. Among these operators, the clone operator is mainly used to increase the number of candidate solutions in the scheme and create the required conditions for the mutation process. The mutation operator is used to maintain the diversity of the solutions in the population, which is conducive to each individual to fully explore the nearby solution area and improve the accuracy of the final

Input:
Original dataset A, stride E, number of nodes n, local iterations T_l
Central server splits A to n participating nodes: n
Batch size of each participating node $A_i = A/n$
For all participating nodes **parallel do**
Initialize the model parameters $\theta_0, \theta_1, \theta_2, \ldots, \theta_k$ of each node i
For all data in dataset A_i
Generate multiple parameter weight update vectors according to A_i
Calculate the corresponding loss function value
Select the optimal update vector
Update the parameters $\theta_0', \theta_1', \theta_2', \ldots, \theta_k'$ of the node model
Until number of iteration $t > T_l$ **do**
Update the global parameter weights for the node model i
$\omega_{i,t+1} \longleftarrow \omega_{i,t} - \gamma_t \nabla_\omega P_i(U_t, \omega_{i.t})$
End for
Aggregate the update information $\nabla\omega_t$ of all nodes
Output
Update central model parameters $\theta_0^c, \theta_1^c, \theta_2^c, \ldots, \theta_k^c$

Algorithm 1: Parallelized model training process.

population. When the above process is completed, individuals with higher affinity are selected and retained, and then the iterative process is repeated until the conditions are met. The general process of the clonal selection algorithm is as follows (Algorithm 2).

In this process, A_{ij}' is the fitness of the $i-$th antibody in the fitness vector A_j' of the mutated antibody population; c_i is the concentration of antibody i; ω is the weighting factor to adjust the weight of the influence of antibody affinity and concentration on the selection probability. Considering that in the early period of the algorithm, more antibodies with high affinity are retained. During the later period of model training using this algorithm, in order to avoid excessive similarity between antibodies, the custom function is usually used to adjust the value of the weight factor ω. The probability S_p of the antibody being selected into the next generation antibody population after mutation is calculated as follows:

$$S_p = \frac{\omega A_{ij}'}{(1-\omega)c_i}. \tag{5}$$

Then, replace the antibodies in the original population with some new mutant antibodies, and the antibodies with lower affinity are more likely to be replaced in this process. Finally, select t antibodies with higher fitness to form the next generation population.

$$Q(k+1) \longleftarrow T_c^S(T(k)) = [q_1(k+1), q_2(k+1), \ldots, q_t(k+1)]. \tag{6}$$

With the continuous in-depth research on the principles of artificial immune systems [27], many deep learning models and algorithms derived from immune principles have gradually been widely used in scientific research and engineering practice and have provided ideas for solving problems such as function optimization and combinatorial optimization. In particular, because the clonal selection algorithm can maintain the diversity of the population, the algorithm based on the immune principle shows the advantage of global convergence in the problem of combinatorial optimization. In most cases, immune algorithms have achieved better results than existing heuristic algorithms, which also inspired us that this type of algorithms will have broad application prospects in the field of artificial intelligence.

3. Methods

3.1. The Federated Learning-Based Clonal Selection Optimization System. With the development of next-generation information technology and the construction of smart cities, the degree of data openness and social involvement in all kinds of scientific research are increasing, data parallel training and model parallel training have become more popular methods for large-scale complex deep learning modeling. However, with the rapid increase of the number of data parallel training devices and the communication overhead between devices, it is necessary to design and implement an excellent secure distributed training system. An excellent distributed computing scheme means more efficient parameter update efficiency and lower communication costs, parameter update information in the centralized deep learning model is processed centrally through the central server, while in the decentralized structure, the training of each computing node is relatively independent, and the parameter update is communicated between different node parameters through the central server.

Decentralized model training schemes usually require higher communication costs and face more serious privacy risks, which brings a huge burden to the central server, and there is also risks of privacy disclosure in the centralized processing of data. Therefore, an adaptive clone selection strategy is used to optimize the selection process of model parameters and feature subsets for local node training in

Input:
Population size G, excellent antibody scale t, clone factor c, mutation probability = m, the maximum number of iterations = k_M
Define:
Population space $Q, Q(0) = \{q_i(0) | 1 \le i \le t\}$, affinity function is A
For each round $k = 1, 2, \ldots$ **do**
For each q_i
Calculate the value of A, select t excellent antibodies
Calculate c for t antibodies and amplify the population
Antibodies with the top t in the population value constitute set $T_c^C(Q(k))$
Clone population is adjusted as $Q'(k) \longleftarrow T_c^C(Q(k))$
Mutation and amplify according to $h(Q''(k) \longleftarrow T_c^m(Q'(k)/m))$
Select t antibodies as next generation population
$$Q(k+1) \longleftarrow T_c^S(Q(k)) = [q_1(k+1), q_2(k+1), \ldots, q_t(k+1)]$$
End if until $k > k_M$, termination calculation
Else $k = k + 1$, calculated value $A(k+1)$

Algorithm 2: Clonal selection algorithm (CSA).

federated learning schemes. Imitating the antibody affinity maturation process in the clonal selection principle, the deep learning model training process is optimized by using the powerful calculation process of biology. Compared with the traditional deep learning methods, the proposed model training method ensures the performance of the model to the greatest extent. In terms of privacy security, this optimization scheme not only retains the security advantages of federated learning in multiparty computing, but also guarantees the data processing level in combination with the differential privacy mechanism, providing a novel method for the balance of availability and privacy. The systematic framework and implementation method of this research are shown in Figure 1.

3.2. Differential Privacy Preprocessing of the Local Data. Similar to the classic federated learning process [28], the local model training under this framework receives the model issued by the central server at the initial stage of each iteration and then uses the locally collected data to optimize the training of the obtained model. When the local model completes a model parameter update, the change in the model weight is recorded, and these recorded data are uploaded to the central server. In this process, the local model training process of each computing node is relatively independent, and communication is achieved through the coordination of the central server [29]. At the beginning of the next iteration, the local model uses the parameters download from the central server as the initial model parameters and then uses the constrained stochastic gradient descent method to update the local model. This research retains the gradient descent method to find the minimum value of the loss function and then uses the clonal selection principle to constrain and optimize the gradient descent process.

In addition, after the data are processed by Laplacian noise that meets privacy requirements, users can directly perform query operations through the terminal model, while avoiding the possibility of recovering the original data due to the increase in the number of queries. In order to prevent parameter and data disclosure incidents during data processing and local training and to further improve the privacy security of the federated learning framework, we have added the differential privacy preserving mechanism in the data preprocessing period. This study uses the method of adding a differential mechanism in the image data processing link, converts image dataset into numerous data matrices, and adds noise to the matrix information. Compared with the method of adding noise to the data label, this method can minimize the impact on data availability while preserving data privacy.

3.3. Constrained Clonal Selection Optimization Process. To a certain extent, deep learning can actually be understood as an implementation method of optimization problems, and the neural network model in deep learning is actually equivalent to a very complex function. The optimization process of the parameters in this function is similar to the process of finding the optimal solution of the function. In the process of deep learning model training, the parameter optimization of neural network is usually transformed into the minimization of loss function. In our research, we use heuristic clonal selection strategy to achieve model optimization, and the optimization direction of the process is constrained by statically processing the dynamic problem.

In the classic stochastic gradient descent process, the direction of the gradient descent is often multidirectional fluctuations, and this fluctuation can prevent the gradient descent process from falling into a local optimum. However, due to the randomness of this fluctuation, the programmability of the model optimization process is reduced, and the model training performance is greatly unstable. In order to improve the performance of the trained model and be inspired by the principle of artificial immunity, we use the clonal selection algorithm to impose certain constraints on this random process and use the selection mechanism to prevent inefficient antibodies from entering the population. This method makes the adjustment of the mutation process

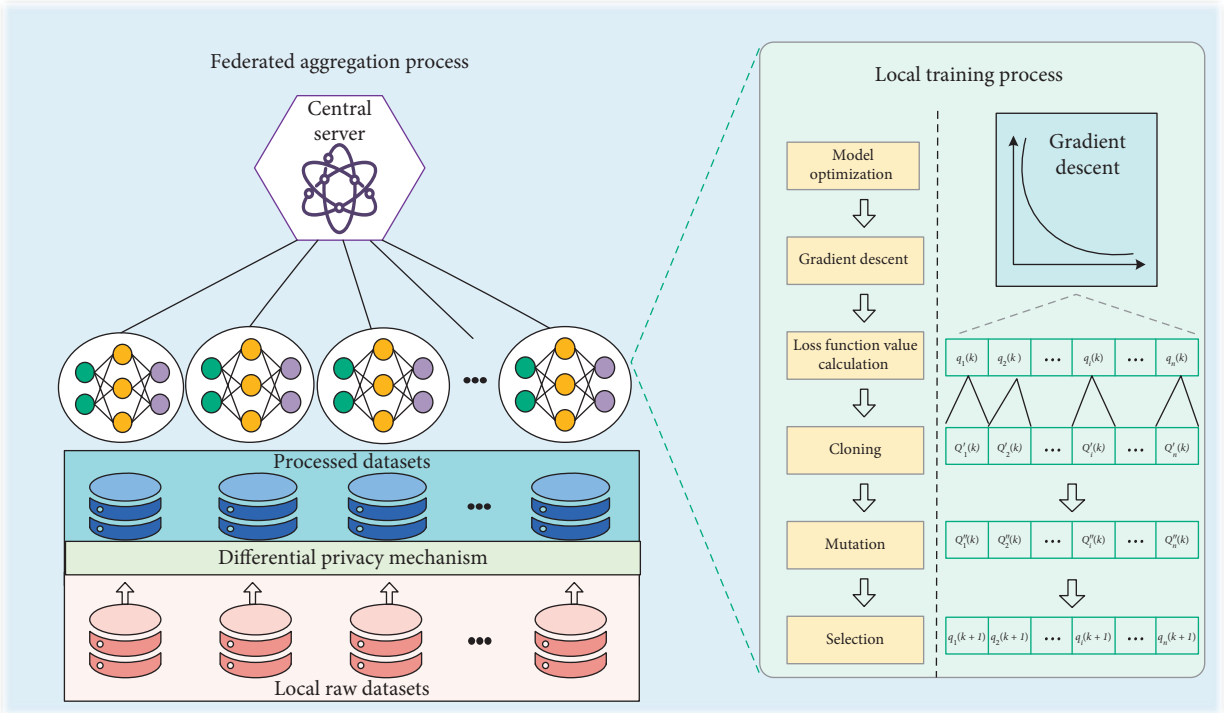


Figure 1: The framework of the federated learning-based clonal selection optimization system.

dependent on the fitness of the antibody and promotes the global optimality of model convergence. In the following CSGD algorithm (Algorithm 3), the objective function is set to represent the optimized model, the affinity value A is inversely proportional to the loss function value, and the iterative training process can achieve a smaller loss function value through continuous learning of the data, so as to achieve a more accurate model performance.

The CSGD algorithm adopts the feature information screening strategy in the clonal selection process to accelerate the convergence speed of the algorithm and improve the model training performance. At the same time, it uses the adaptive adjustment step size mechanism in the clone and mutation operators to improve the solution accuracy performance of the algorithm so that the algorithm avoids falling into the local maximum.

3.4. Aggregation Process of Federated Learning. As mentioned above, the optimization process of clonal selection based on federated learning includes two different processes: the local model training and the federated parameter aggregation and updating. During the local training process, we train the model based on the artificial neural network and use the clonal selection strategy to make the model achieve global convergence at the node, and then each local node extracts the parameter update information from the model after each round of training and upload the information to the central server. In the process of federated learning aggregation, each local client completes a training session and immediately uploads the latest parameter selection part, and the server immediately updates the global model and then broadcasts throughout the federation.

Compared with distributed deep learning, federated learning has less control over each node and only collects local model parameters after each iteration, making the model parameters of the federated center more globally representative and accurate. Updating the local models aims to increase the average contribution of the model to the central server during each iteration and reduce the uneven participation of equipment and data in the distributed computing scenarios.

In the federated learning process (Algorithm 4), the client uses the local data to train the model, and the central server collects the model parameters of each client and uses the related functions to aggregate the parameters and update the federated model. In the next iteration, the client-side node model uses its own local data to optimize the model sent by the central server, and the clonal selection strategy is adopted to restrict the gradient descent process. The client compares the model parameters of the current iteration of the local model with those of the previous iteration, calculates the update difference, and uploads it to the central server. The representation function of the central model parameter is $h(x)$, all clients that meet the requirements to participate in the training are represented as R_A.

In addition to effectively improving communication efficiency, federated learning can also coordinate various clients with unbalanced data to complete tasks together. In the calculation process at this stage, the central server of federated learning collects the model parameters after the

Input:
Objective function $R_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_i$, loss function $F(\theta_0, \theta_1) = \sum_{i=1}^{t} (R_\theta(x_i) - y_i)^2$, and affinity value is A
Initial:
Some sample datasets for features $D_1, D_2, \ldots, D_k$ ($D_k = (x^i, y^i)(i = 1, 2, 3, \ldots, t)$), x is sample features, P_θ^M: $\theta_0, \theta_1, \theta_2, \ldots, \theta_k$ are model parameters
For model P_θ^M: all $\theta_i = 0$, stride size $\alpha = 1$, mutation probability is m, clone probability is c, ending distance $= \beta$
Determine the gradient corresponding to θ_i, the current position, the gradient is $(\partial/\partial\theta_i)F(\theta_0, \theta_1, \ldots, \theta_n)$
For each iteration
For all random sample data **do**
Generate the gradient vector $g_1, g_2, \ldots, g_k$ and the corresponding value of F during model learning
Calculate A according to F
Reserve the gradient vector group $g_1, g_2, \ldots, g_n$ with bigger A
Optimal selection of gradient vector by clonal selection strategy:
Calculate c according to A, population adjust as $Q'(k) \longleftarrow T_c^C(Q(k))$
According to m, population adjusted as $Q''(k) \longleftarrow T_c^m(Q'(k)/m)$
Retain to original population size:
Keep t excellent vectors: $Q(k+1) = [q_1(k+1), q_2(k+1), \ldots, q_t(k+1)]$
Iterative condition judgment
If $k > k_{\max}$
Output:
Update the population P_θ^M based on the selected antibodies

Algorithm 3: Clone selective optimized gradient descent (CSGD).

Initial:
Parameters function $h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_i$, d is the maximum engagement of each client, the maximum iterations of node is $I_{\max}$
Federated central sever **do**:
For each weight θ_i of model **do**
For each iteration $i = 1, 2, \ldots$ **do**
Local optimization process:
Select a fixed number of eligible clients $R_i \longleftarrow R_A$
For each clients $k \in R_i$ **do**
Optimize parameters with local data: $\theta_{i+1}^k \longleftarrow (C_k, \theta_i)$
$\theta_{i+1} \longleftarrow \sum_{k=1}^{|R_i|} (n_k/n)\theta_{i+1}^k$
For client C_k: Client parameter update (C_k, θ_i) **do**
Divide the dataset D_A into several subdatasets D_s
For local iteration $i \longrightarrow [1, I_{\max}]$ **do**
Update $\theta \longleftarrow \theta - \eta \nabla lr(\theta; b(b \in D_s))$
Return θ to the federated central server

Algorithm 4: Federated learning process.

adaptive gradient descent optimization of each client and imitates the classic federated learning model to continuously optimize the model.

4. Experiments

In this section, we evaluate our proposed adaptive clonal selection federated learning model for stabilizing and improving the performance of federated edge learning platforms. These experiments are executed on the computer Intel Core i5-1135G7 2.40 GHz, 16 GB of RAM memory, and Windows 10 operating system. In addition, the MNIST handwritten datasets (http://yann.lecun.com/exdb/mnist/) are used as training data. In order to observe the training effect of this novel optimized model, this research compare the CSGD model with the classic gradient descent model. This experiment uses the PyTorch deep learning framework, and CNN neural network is used in the training process of local nodes. In the simulation experiments, the accuracy and stability of the systematic optimization model and the classic model are compared. The scheme efficiency of local model training and federated model training was also compared through the overall running time. Finally, the balance performance between privacy security and usability of this system is evaluated.

4.1. The Performance of the Training Process and Accuracy. To compare the accuracy performance of the federated clonal selection optimization model and the classic federated

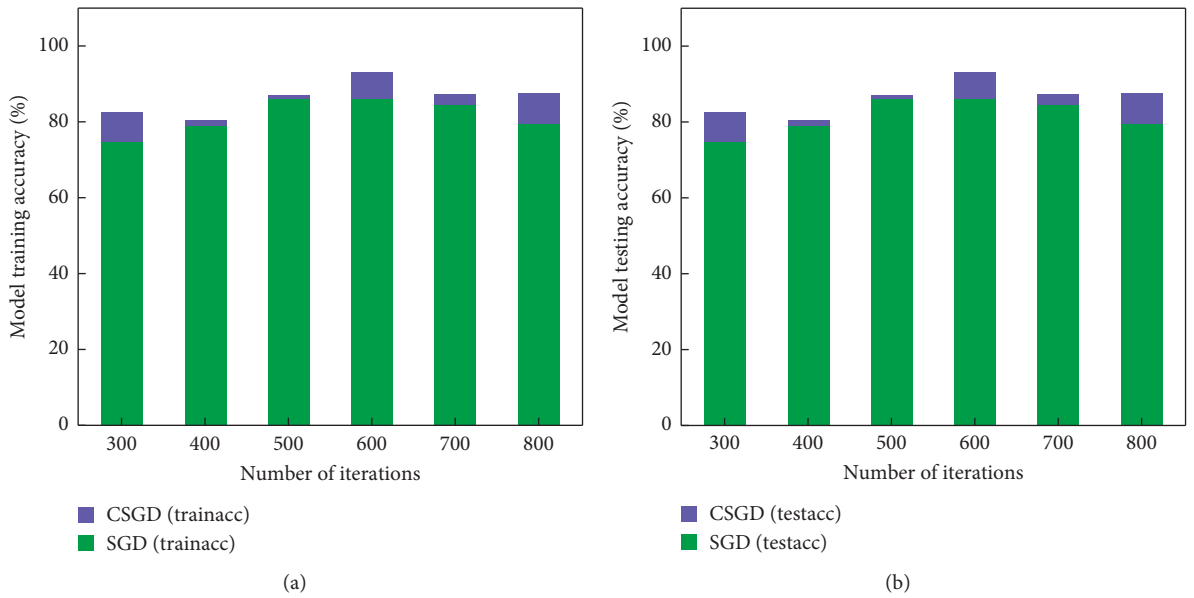


Figure 2: Model accuracy performance corresponding to different iteration times.

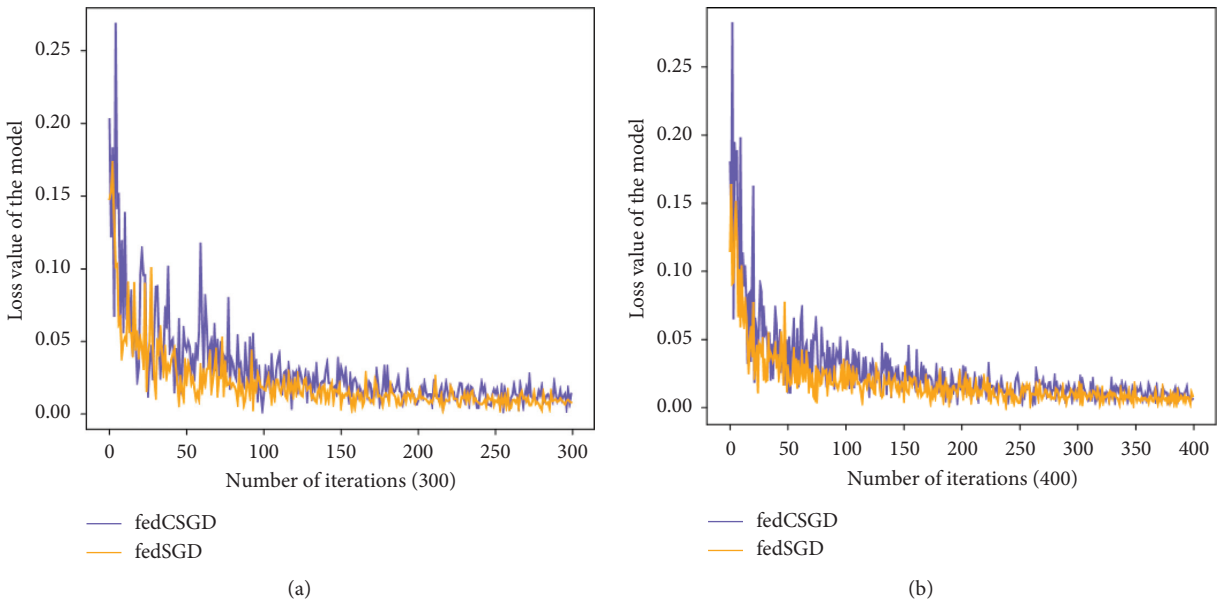


Figure 3: Continued.

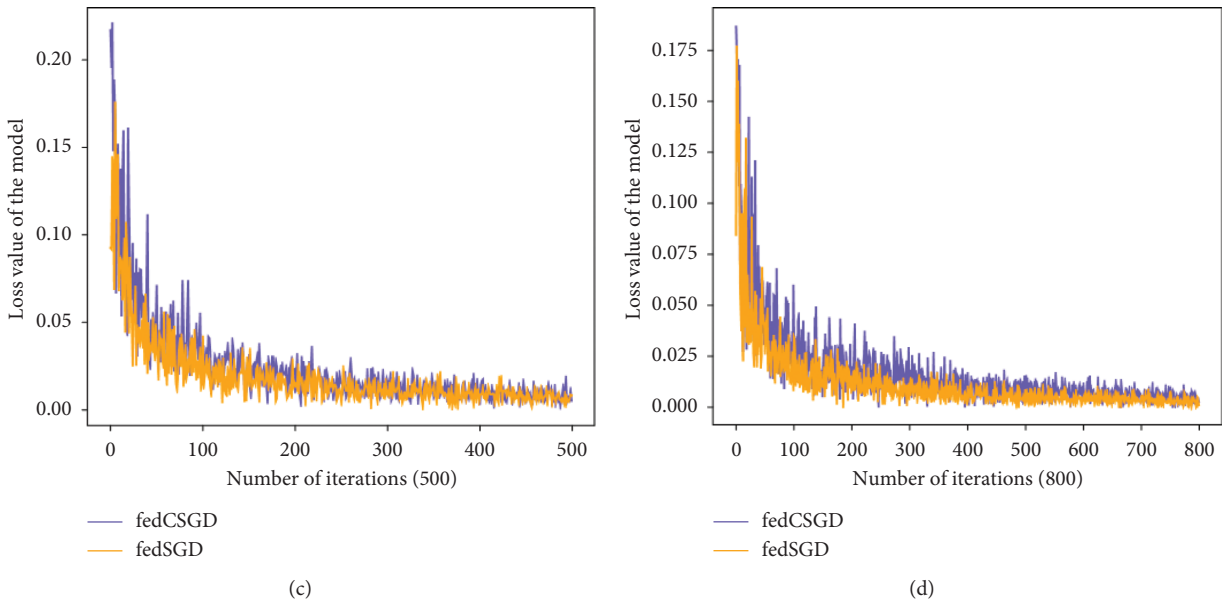


Figure 3: Loss function value corresponding to different iteration times.

learning model, this research conducted experiments under different model training iterations. It aims to evaluate the performance of the two algorithms by comparing the model performance under the same number of iterations training or communication consumption. This experiment shows the performance difference of two algorithms used in federated learning under different iteration times, and the model performance was evaluated by training accuracy and testing accuracy.

According to the results in Figure 2, it can be seen that the accuracy performance of the model trained by CSGD algorithm is better than or almost equal to SGD algorithm in most cases; the performance of the CSGD model is also more stable and the advantages of the CSGD algorithm are more obvious after more iterations. In addition, this experimental result suggests that the influence of abnormal nodes exit in the federated training process of the CSGD model is less than that of the classical stochastic gradient descent model.

As it can be seen from Figure 3, during different training iterations, especially in the early training periods, the loss function value of the model trained by the CSGD algorithm fluctuates greatly, and these fluctuations also provide a variety of options for the optimization direction of the model. With the increase of the number of iterations, the model trained by the CSGD algorithm corresponds to a lower loss function value, which means that the more diversified the choices of optimization direction, the better performance of model in the later period of training.

4.2. Convergence Efficiency of the Training Process. In this experiment, the optimized model and the general CNN local training model are compared under the same parameters setting. For example, we compared the efficiency and accuracy of the fedCSGD and cnnSGD models under the same number of training iterations and data processing volume.

It can be seen from the experimental results in Figure 4 that, under the same data scale and number of iterations, the federated learning model fedCSGD takes significantly less running time to complete the same number of iterations than cnnSGD model, which is related to the aggregated improvement of model performance by federated learning and the performance optimization of clonal selection. In addition, the average accuracy performance of the algorithm used in this study is slightly worse than that of the cnnSGD algorithm, but because the training time required for the two algorithms is quite different, the conclusion of this study is that the fedCSGD algorithm has higher training efficiency.

4.3. Comparison of the Model Stability under Different Hyperparameters. Considering that the artificial immune clonal selection algorithm has the advantages of the global optimality of the convergence results, this experiment aims to evaluate whether the proposed optimized model has stronger stability in response to changes in some hyperparameters. In this experiment, we use the number of clients in federated learning as a variable to test the different performance of the two models. Theoretically, the number of participating clients is related to the quality of the model, and the performance of training model can reflect the stability of the federated learning framework during the asynchronous update process to a certain extent.

It can be seen from the experimental results in Figure 5(a) that the optimization model proposed in this study has better accuracy performance under the same number of clients and may perform better when the number of clients is small. More importantly, in the experiments corresponding to different numbers of customers as shown above, the accuracy of the model proposed in this study is more stable. In Figure 5(b), the fedCSGD model also shows robustness in the process of loss function optimization for

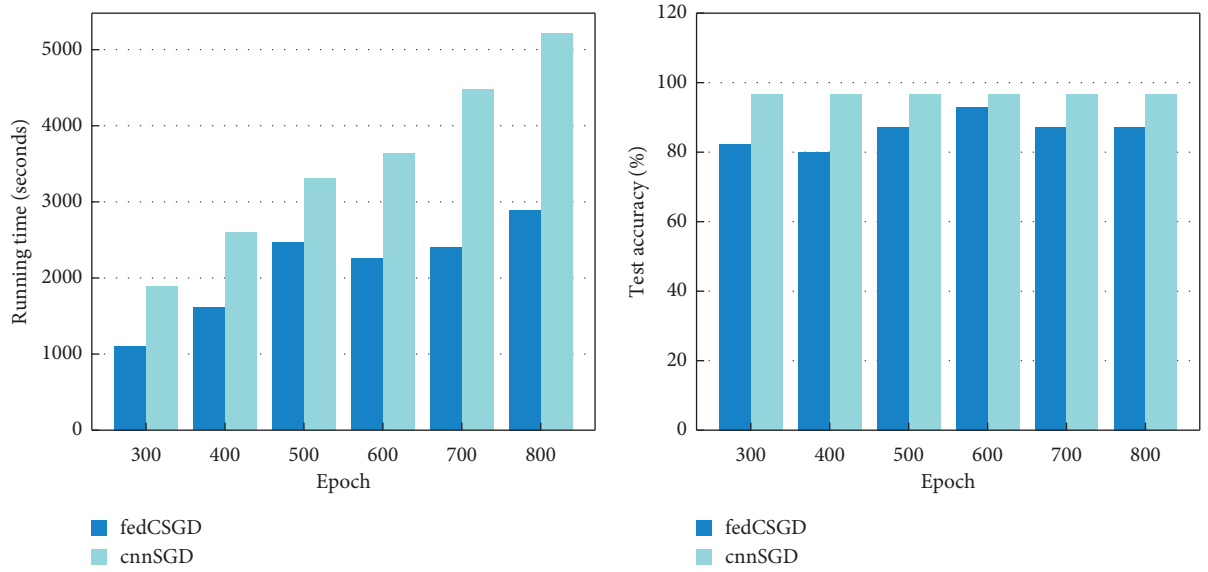
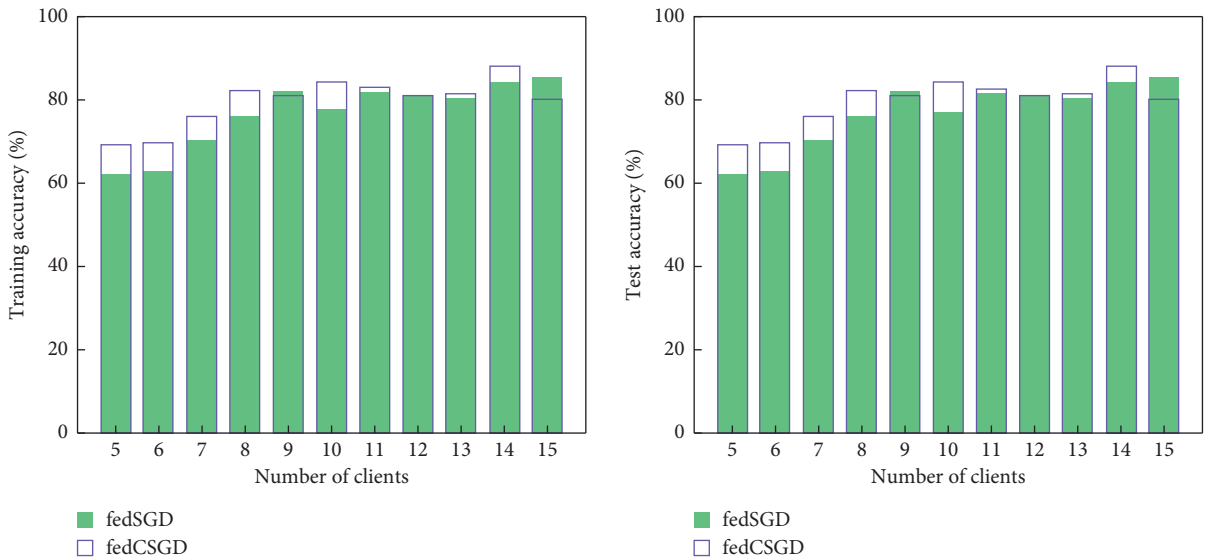


Figure 4: Convergence efficiency of different models.



(a)

Figure 5: Continued.

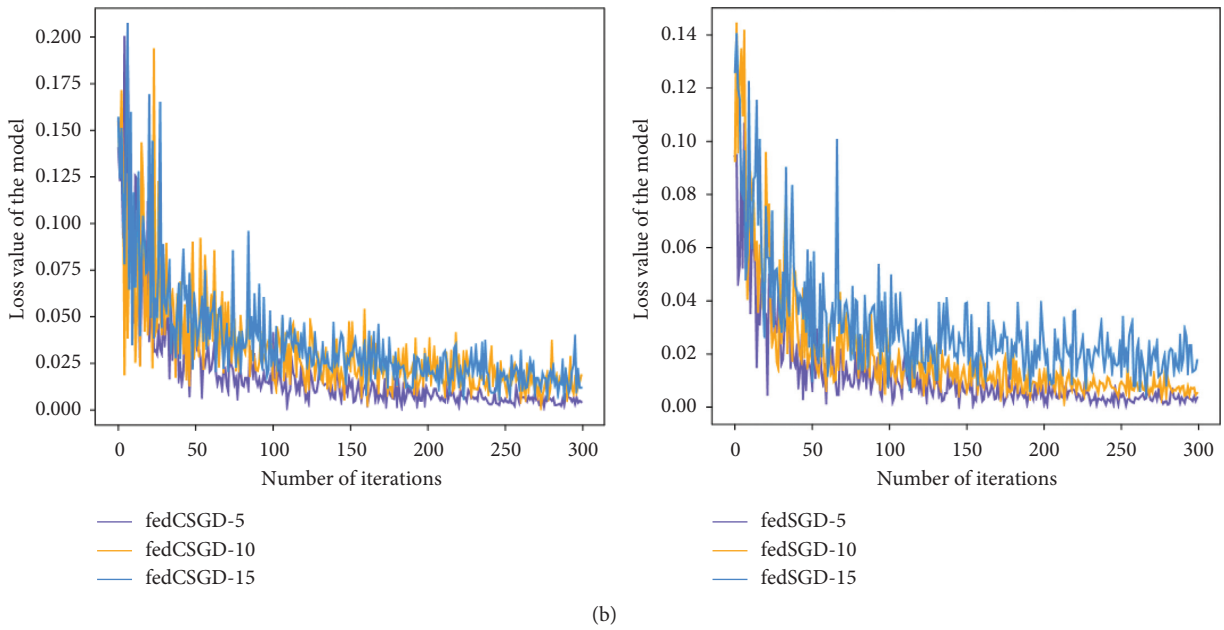


Figure 5: Stability of the model performance under different parameter settings.

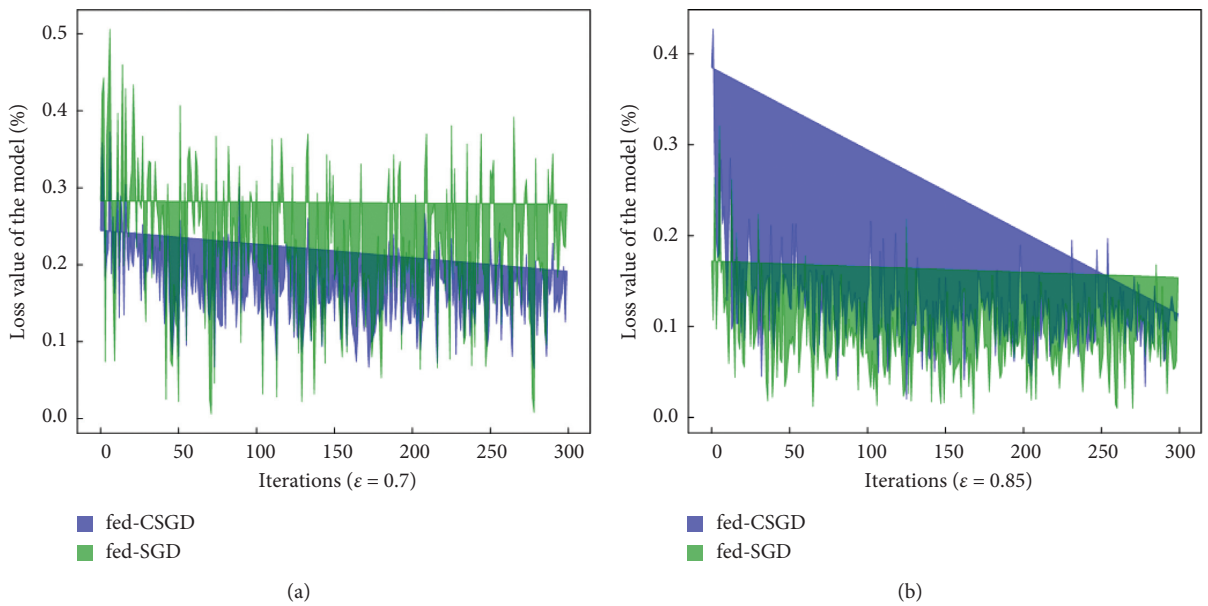


Figure 6: Model performance under the differential privacy mechanism.

different number of customers. The optimized model has less fluctuation and better performance, and the model's loss value decreasing process is more stable. This reminds us that the optimized model is compatible with more usage scenarios, and at the same time, it is easier to use for operators with less experience.

4.4. Performance Balance between Privacy and Availability. In this experiment, we added differential privacy processing mechanism in the model training process and compared the performance of the fedSGD model and fedCSGD model after adding noise during the training process. It can be seen from the results, under the same privacy budget, that the model performance of the two algorithms fluctuates greatly due to the influence of differential privacy noise, but the fedCSGD algorithm still performs better in terms of convergence efficiency. Moreover, the fedCSGD algorithm is more stable than SGD when dealing with different privacy budget parameter settings. This may be because the uncertainty caused by stochastic gradient descent is further amplified in the process of clonal selection, and as the noise increases, the difference in the performance of the models trained by the two algorithms is no longer obvious.

It can be seen from the experimental results in Figure 6 that, under the same privacy budget between 0.7 and 0.85, the loss value curve performance of the proposed optimized CSGD model is more stable than that of the SGD model in responding to changes in the privacy budget. As can be seen from Figure 6(a), the loss function value of the optimized model is generally low and the fluctuation is relatively small under the condition of small privacy budget and large noise experimental settings. As can be seen from Figure 6(b), when the privacy budget is larger, the loss function value corresponding to the optimized model fluctuates greatly in the early period, but finally reaches a smaller loss function value due to the advantage of the evolutionary direction, which also indicates that the optimized algorithm has a better antinoise ability in the face of changes in the privacy budget. This also reminds us that the CSGD algorithm can achieve a better balance between privacy and utilization of data, and at the same time ensure that the convergence effect of the model is less affected by noise.

5. Conclusions

The construction of smart cities aims to use intelligent technology to associate new virtual information space with physical resources, dealing with complex issues through artificial intelligence and deep learning modeling. In this paper, the modeling process of deep learning was optimized according to the construction requirements of innovative smart cities, and the usability of the models trained by deep learning to deal with complex problems was improved from the aspects of data processing capacity, training efficiency, and privacy security. To tackle these issues in deep learning modeling of big data, this research introduces the federated learning framework, which aims to improve the existing distributed deep learning model from the two aspects of training process optimization and privacy security improvement. In this research, the clonal selection principle is used to constrain the classic gradient descent process, and the federated learning scheme and differential privacy noise mechanism are used to enhance the privacy security and achieve the higher data availability.

In order to improve the level of data openness and social involvement in the construction of smart cities, this framework adjusts the large-scale distributed deep learning scheme, and biological principles are used to optimize the gradient descent process of each node's local modeling. It also realizes the combination of artificial immune clonal selection principle and gradient descent process. The simulation experiments results show that the fedCSGD system improves the training performance and efficiency of the model, has more stability in the different setting of hyperparameters, and achieves a degree of balance between privacy security and data availability. The system provides a method with higher privacy security during processing and using of the data containing sensitive information, breaks the dilemma of data islands, and provides a distributed multiparty computing framework suitable for complex modeling problems and more maneuverable.

In the future work, we will further improve the convergence efficiency of model training and the availability of artificial intelligence in processing scenarios and enhance the interactivity of the overall operation process of the system. In addition, biological datasets will be used instead of traditional datasets in related studies to achieve the availability of the model in biological big data processing and privacy preserving.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 62003291), the Xuzhou Science and Technology Project (Grant no. KC20112), the Project of Philosophy and Social Science Research in Colleges and Universities in Jiangsu Province (Grant no. 2020SJA1056), the Industry-University-Research Cooperation Project of Jiangsu Science and Technology Department (Grant no. BY2018124), and the National Science and Technology Foundation Project (Grant no. 2019FY100103).

References

[1] T. Glenn and S. Monteith, "Privacy in the digital world: medical and health data outside of HIPAA protections," *Current Psychiatry Reports*, vol. 16, no. 11, p. 494, 2014.

[2] R. Avinash, "Stochastic gradient descent–whale optimization algorithm-based deep convolutional neural network to crowd emotion understanding," *Computer Journal*, vol. 63, no. 2, pp. 267–282, 2020.

[3] A. Soodabeh and V. Manfred, "A learning rate method for full-batch gradient descent," *Műszaki Tudományos Közlemények*, vol. 13, no. 1, pp. 174–177, 2020.

[4] P. Toulis and E. M. Airoldi, "Asymptotic and finite-sample properties of estimators based on stochastic gradients," *Annals of Statistics*, vol. 45, no. 4, pp. 1694–1727, 2017.

[5] K. Jakub, J. Liu, P. Richtarik, and M. Takac, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2016.

[6] S. Yavari, M. J. V. Zoej, M. Mokhtarzade et al., "Comparison of particle swarm optimization and genetic algorithm in rational function model optimization," in *Proceedings of the XXII ISPRS Congress*, Melbourne, Australia, September 2012.

[7] Y. F. ZHANG and S. Bhattacharyya, "Genetic programming in classifying large-scale data: an ensemble method," *Information Sciences*, vol. 163, no. 1–3, pp. 85–101, 2004.

[8] M. Diaz, B. Rubio, E. Soler, and J. M. Troya, "A border-based coordination language for integrating task and data parallelism," *Journal of Parallel & Distributed Computing*, vol. 62, no. 4, pp. 715–740, 2002.

[9] R. Barnes, C. Lehman, and D. Mulla, "Distributed parallel D8 up-slope area calculation in digital elevation models," in *Proceedings of the International Conference on Parallel & Distributed Processing Techniques & Applications*, Las Vegas, NV, USA, July 2012.

[10] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, Chicago, IL, USA, November 1982.

[11] H. B. MCMAHAN, E. MOORE, D. RAMAGE et al., "Communication-efficient learning of deep networks from decentralized data," 2016, https://arxiv.org/abs/1602.05629.

[12] Q. Yang, Y. Liu, T. J. Chen, and Y. X. Tong, "Federated machine learning: concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, 2019.

[13] J. Konen, H. B. Mcmahan, F. X. Yu et al., "Federated learning: strategies for improving communication efficiency," 2016, https://arxiv.org/abs/1610.05492.

[14] Y. Liu, Y. Kang, C. P. Xing, T. J. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2020.

[15] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, vol. 78, pp. 65–74, 2016.

[16] E. P. Xing, Q. Ho, W. Dai et al., "Petuum: a new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 1335–1344, 2015.

[17] L. Sweeney, "k-ANONYMITY: a model for protecting privacy," *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2012.

[18] N. Papernot, P. Mcdaniel, I. Goodfellow et al., "Practical black-box attacks against machine learning," 2016, https://arxiv.org/abs/1602.02697.

[19] C. Dwork, "Differential privacy: a survey of results," in *Proceedings of the Theory and Applications of Models of Computation. TAMC 2008*, Xi'an, China, April 2008.

[20] S. Pal, E. Ebrahimi, A. Zulfiqar et al., "Optimizing multi-GPU parallelization strategies for deep learning training," *IEEE Micro*, vol. 39, no. 5, pp. 91–101, 2019.

[21] A. B. Hernandez, M. S. Perez, S. Gupta, and V. Muntes-Mulero, "Using machine learning to optimize parallelism in big data applications," *Future Generation Computer Systems*, vol. 86, pp. 1076–1092, 2018.

[22] P. Jain, P. Netrapalli, S. M. Kakade, R. Kidambi, and A. Sidford, "Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification," *Journal of Machine Learning Research*, vol. 18, 2018.

[23] P. Netrapalli, "Stochastic gradient descent and its variants in machine learning," *Journal of the Indian Institute of Science*, vol. 99, no. 2, pp. 201–213, 2019.

[24] M. Li, T. Zhang, Y. Chen, and AJ. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA, August 2014.

[25] D. Kingma and J. Ba, "ADAM: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[26] M. S. Hashemipour and S. A. Soleimani, "Artificial immune system based on adaptive clonal selection for feature selection and parameters optimisation of support vector machines," *Connection Science*, vol. 28, no. 1, pp. 1–16, 2016.

[27] D. Dasgupta, S. Yu, and F. Nino, "Recent advances in artificial immune systems: models and applications," *Applied Soft Computing*, vol. 11, no. 2, pp. 1574–1587, 2011.

[28] S. Wang, T. Tuor, T. Salonidis et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[29] X. T. Zhang, X. M. Zhu, J. Wang, H. Yan, K. Chen, and W. D. Bao, "Federated learning with adaptive communication compression under dynamic bandwidth and unreliable networks," *Information Sciences*, vol. 540, 2020.