Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9791865, 1 page
https://doi.org/10.1155/2023/9791865

Retraction

Retracted: Evaluation of the Online and Offline Mixed Teaching Effect of MOOC Based upon the Deep Neural Network Model

Wireless Communications and Mobile Computing

Received 1 August 2023; Accepted 1 August 2023; Published 2 August 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] G. Wang and X. Chen, "Evaluation of the Online and Offline Mixed Teaching Effect of MOOC Based upon the Deep Neural Network Model," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 2173005, 12 pages, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2173005, 12 pages
https://doi.org/10.1155/2022/2173005

Research Article

Evaluation of the Online and Offline Mixed Teaching Effect of MOOC Based upon the Deep Neural Network Model

Guangwei Wang and Xiaomei Chen

Aviation Service and Tourism Management College, Guilin University of Aerospace Technology, Guilin, 541004 Guangxi, China

Correspondence should be addressed to Guangwei Wang; wangguangwei198809@guat.edu.cn

Received 9 December 2021; Revised 11 January 2022; Accepted 10 February 2022; Published 19 March 2022

Academic Editor: Deepak Kumar Jain

This article is dedicated to discussing the online and offline mixed teaching evaluation of MOOC based on deep neural networks. Deep neural networks are an important means to solve various problems in various fields. It can evaluate the teaching attitude of teachers, the teaching content in the classroom, the teacher's narrative ability, the teaching methods used by the teachers, and whether the teaching methods are rigorous. And it can train on a large number of datasets evaluated by students on a certain course and get results. This article first explains the advantages of the neural network model and explains the reasons for the emergence of MOOCs and the mixing with traditional classrooms. It also explains some deep neural network (DNN) models and algorithms, such as BP neural network model and algorithms. This model uses backpropagation. When there is an error between the output sample of the neural network and the target sample, the error can be backpropagated to adjust the threshold and weight to make the error reach the minimum. The algorithm steps include forward propagation and backpropagation and are substituted into the gradient descent method to obtain the weight change of the output layer and the hidden layer. It also explains the Gaussian model in DNNs. The given training data vector in the Gaussian mixture model and the configuration of GMM are used for expectation maximization training using an iterative algorithm, and the unsupervised clustering accuracy ACC is applied to evaluate its performance. Use pictures to describe the mixed-mode teaching mode in the MOOC environment. It is necessary to consider teaching practice conditions, time, location, curriculum resources, teaching methods and means, etc. It can cultivate students' spatial imagination, engineering consciousness, creative design ability, drawing hand-made ability, and logical thinking abilities. It enables teachers to accept the fair and just evaluation of students. Finally, this article discusses the parallelization and optimization of GPU-based DNN models, splits the DNN models, and combines different models to calculate weight parameters. This article combines model training and data training in parallel to increase the processing speed under the same amount of data, increase the batch, increase the accuracy, and reduce the training shock. It can be concluded that its DNN model has greatly improved the training effect performance of the MOOC online and offline mixed course effect dataset. The calculation time is shortened, the convergence speed is accelerated, the accuracy rate is improved, and the acceleration ratio is increased, which compares with the same period of the previous year increase of more than 37.37%. The accuracy has increased, comparing with the same period of the previous year, an increase of more than 12.34%.

1. Introduction

1.1. Background. DNN models have made continuous breakthroughs in various fields, and they are also accompanied by shadows in life, such as artificial intelligence. There is no shortage of participation of DNN models. Whether it is machine learning or pattern recognition, when people use primitive technology to process data, they will always suffer great limitations due to the complexity and size of the data. At this time, the arrival of the DNN model is undoubtedly a long-term drought. It can express the complexity and abstract state of data with extremely simplified nonlinearity and turn it into a quantity that can be understood by the machine. In the era of educational informationization, traditional classroom education models and pure online learning have their own shortcomings, and MOOCs have emerged

from this. It can improve its teaching links. It has a wealth of teaching concepts, well-designed teaching links, high-quality teaching resources, effective learning models, and a strong teaching staff shining in the field of education. However, it also has certain shortcomings, such as students tend to lose self-discipline, miss classes, and fail to complete homework. Therefore, the MOOC and the classroom are mixed online and offline to make up for the shortcomings and limitations. When evaluating the teaching effect, there are many factors of teaching quality itself, and they bring different degrees of influence. In the era of DNN models, although there will be certain errors in the evaluation, it is more to optimize the evaluation model. It uses nonlinear mapping and dataset calculations to perform predictive analysis and calculations on its data to obtain the results. The optimized neural network model mentioned in this article will also improve the general DNN model while processing data. It uses parallel mode to improve accuracy and speed up convergence.

1.2. Significance. The purpose of this article is to study the application of DNNs in the evaluation of online and offline mixed teaching effects. Through the backpropagation network, the weights and thresholds of the network are continuously optimized, and the teaching mode is designed to cultivate students' good spatial imagination, engineering awareness, creative design ability, drawing ability, and logical thinking ability, using Gaussian mixture model to fit the function to maximize the expectation. This paper takes the 2016 to 2020 online and offline MOOC evaluation data collected by the educational administration system of a university as a sample, using caffe and optimized parallel dual GPU model to process data and train. The three datasets were trained 120000 times, 8000 times, and 4000 times, respectively, which proved that the optimized DNN model is indeed superior in performance. It has better guiding significance in realizing large-scale data processing. The results of this experiment have a certain guiding role in the evaluation of the online and offline mixed teaching effect of MOOC and classroom with the DNN model, and it has both theoretical and practical meanings.

1.3. Related Work. The application of DNN models in various fields has extremely optimistic development prospects. All fields have never stopped the research and exploration of neural networks, to have a brighter development prospect for DNN models. Albericio observed that most of the calculations performed by DNNs are essentially invalid. Because they involve multiplication, one of the inputs is zero. This observation inspired Cnvlutin (CNV). This is a value-based hardware acceleration method that eliminates most of these invalid operations and improves the performance and energy of the most advanced accelerators without loss of accuracy [1]. Xue has developed a multitask deep convolutional network that simultaneously detects the existence of the target and the geometric properties (position and direction) of the target relative to the region of interest. Secondly, the structured visual inspection is carried out by using the cyclic neuron layer [2]. Gong proposed a multiobjective sparse feature learning model based on autoencoder. The parameters of the model are learned by optimizing the reconstruction error and the sparsity of hidden units at the same time, so as to automatically find a reasonable compromise between them. Based on the multiobjective evolutionary algorithm, he designed a multiobjective induction learning program for the model [3]. Cai proposed a new nonlinear and non-Gaussian process monitoring method using weighted KICA (WKICA) based on Gaussian mixture model (GMM). In particular, in WKICA, he first used GMM to estimate the probability of KIC extracted by KICA. Then, according to the estimated probability, the important KICs that reflect the main process changes are distinguished, and a larger weight is assigned to capture important information during online fault detection [4]. To construct a circular linear joint distribution containing appropriate correlation items, Parui uses a semiwrapped Gaussian distribution. In addition, he constructed this joint distributed hybrid model (called SWGMM) [5]. Then, Chen adopted a dual parallel method in the RF training process. Based on PRF's parallel training process and the dependence of elastic distributed datasets, he created a task-directed acyclic graph (DAG) (RDD) object. Then, he calls different task schedulers for the tasks in the DAG [6]. Carrasco said that kernel-based formulas can be directly derived by applying duality theory. Each twin problem has the same SVR method structure, allowing fast training using efficient optimization algorithms. His duality theory provides a general formula for the twin SVR. Compared with the original TSVR, it brings better performance [7]. Although the research he cited has reached certain guiding conclusions, there are unavoidable errors that need to be further improved.

1.4. Innovation. This article analyzes the application of DNNs in the evaluation of MOOC's online and offline mixed-mode teaching effects. Combining the BP neural network model and algorithm gradient descent method, this article discusses the DNN and the online and offline mixed-mode teaching of MOOC. The DNN model of GPU is optimized in parallel, and the DNN model of the dual GPU is obtained. Combining model training and data training to improve the performance of DNNs in parallel, the error formed by the test results is very small. Moreover, it has fast convergence and high accuracy, which can be well applied in dataset training.

2. DNN Model and Algorithm and Mixed Curriculum Design

2.1. BP Neural Network Model and Algorithm

2.1.1. BP Neural Network Model. BP neural network is a backpropagation network. Its application scenarios include function approximation, pattern recognition, classification, and data compression. It is a multilayer prefeedback neural network that can be trained according to the error reversal propagation algorithm. It can learn and store complex mapping relationships without expounding the mathematical equations of the input-output model mapping relationship in advance [8]. It can apply the steepest descent method

and backpropagation to continuously optimize the weights and thresholds of the network so that the neural network reaches the least square error [9]. The topological structure of the BP neural network model is shown in Figure 1.

It can be seen that the BP neural network structure includes an input layer, a hidden layer, and an output layer.

2.1.2. BP Neural Network Algorithm and Learning Rules. Suppose the input samples are $A_1, A_2, \cdots, A_n$, the target samples are $B_1, B_2, \cdots, B_n$, and the output layer of the BP neural network is $O_1, O_2, \cdots, O_n$. When there is an error between the output sample of the neural network and the target sample, the error can be backpropagated to adjust the threshold and weight. It makes the error reach the minimum, and its algorithm steps include forward propagation and backpropagation [10].

According to the forward transfer of its information, let the input layer be a_1 to a_t, the hidden layer be $b1_x$, and the output layer be $b2_j$. It can be seen that the output of the x th neuron in the hidden layer is

$$b1_x = f1\left(\sum_{y=1}^{t} wl_{xy}a_y + c1_x\right), \quad x = 1, 2, \cdots, s1. \tag{1}$$

The difference between the output of the jth neuron in the output layer and the defined function is

$$b2_j = f2\left(\sum_{x=1}^{s1} w2_{jx}a_x + c1_j\right), \quad j = 1, 2, \cdots, s2,$$
$$E(W, B) = \frac{1}{2}\sum_{j=1}^{s2}\left(T_j - b2_j\right)^2. \tag{2}$$

Then, substitute the gradient descent method to obtain the weight change of the output layer and the hidden layer and the backpropagation of the error. $w2$ represents the weight of the output layer, and the weight of the output layer changes as

$$\Delta w2_{jx} = -\lambda\frac{\mu E}{\mu w2_{jx}} = -\lambda\frac{\mu E}{\mu b2_j}\cdot\frac{\mu b2_j}{\mu w2_{jx}} = -\lambda\left(T_j - b2_j\right)f2'b1_x = \lambda\mu_{jx}. \tag{3}$$

Among them,

$$\mu_{jx} = \left(T_j - b2_j\right)f2' = e_j f2', \quad e_j = T_j - b2_j,$$
$$\Delta c2_{jx} = -\lambda\frac{\mu E}{\mu c2_{jx}} = -\lambda\frac{\mu E}{\mu b2_j}\cdot\frac{\mu b2_j}{\mu c2_{jx}} = -\lambda\left(T_j - b2_j\right)f2' = \lambda\mu_{jx}. \tag{4}$$

The weight change input to the hidden layer is

$$\Delta w1_{jx} = -\lambda\frac{\mu E}{\mu w12_{jx}}1 = \lambda\mu_{jx}a_y, \tag{5}$$

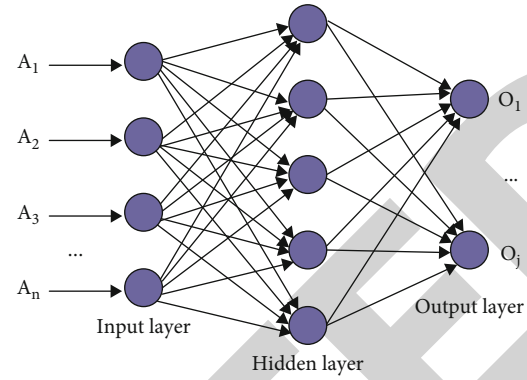

Figure 1: BP neural network model.

$$\Delta c_x = \lambda\mu_{xy}. \tag{6}$$

The activation function for $f1$ being a logarithmic sigmoid is

$$f1(k) = \frac{1}{1 + e^{-k}},$$
$$f1f1'(k) = \frac{0 - e^{-k}(-1)}{\left(1 + e^{-k}\right)^2} = \frac{e^{-k}}{\left(1 + e^{-k}\right)^2} = f1(k)[1 - f1(k)]. \tag{7}$$

For $f2$ is a linear activation function,

$$f2'(k) = k' = 1. \tag{8}$$

2.1.3. Improvement of BP Algorithm. Use the additional momentum factor to adjust the BP neural network algorithm, let W be the weight matrix, A is the input vector of the neural network, α is the momentum coefficient of the algorithm, and $\alpha \in (0, 1)$. It is known that the direction of the standard BP neural network algorithm to adjust the weight is only the direction of the error gradient at time T. The previous error value has not been adjusted. This situation will cause its instability, oscillation, and slow convergence. Using an additional momentum factor to adjust, it can reduce the oscillation and speed up convergence [11]:

$$\Delta W(T) = \lambda\mu A + \alpha\Delta W(T - 1). \tag{9}$$

Knowing $\Delta w = -\lambda(\mu E/\mu w)$, the learning rate is positively correlated with the convergence speed, but it will cause oscillations and even divergence. To obtain stable convergence, the learning rate will become slow, and we can improve the error function. It is known that $E = (1/2)\sum_{a=1}^{a}\sum_{j=1}^{m}\ln\left(g_j^a - p_j^a\right)^2$; as the number of learning increases, the value of $|g_j^a - p_j^a|$ will gradually become smaller. This situation will slow down the approximation speed of the function.

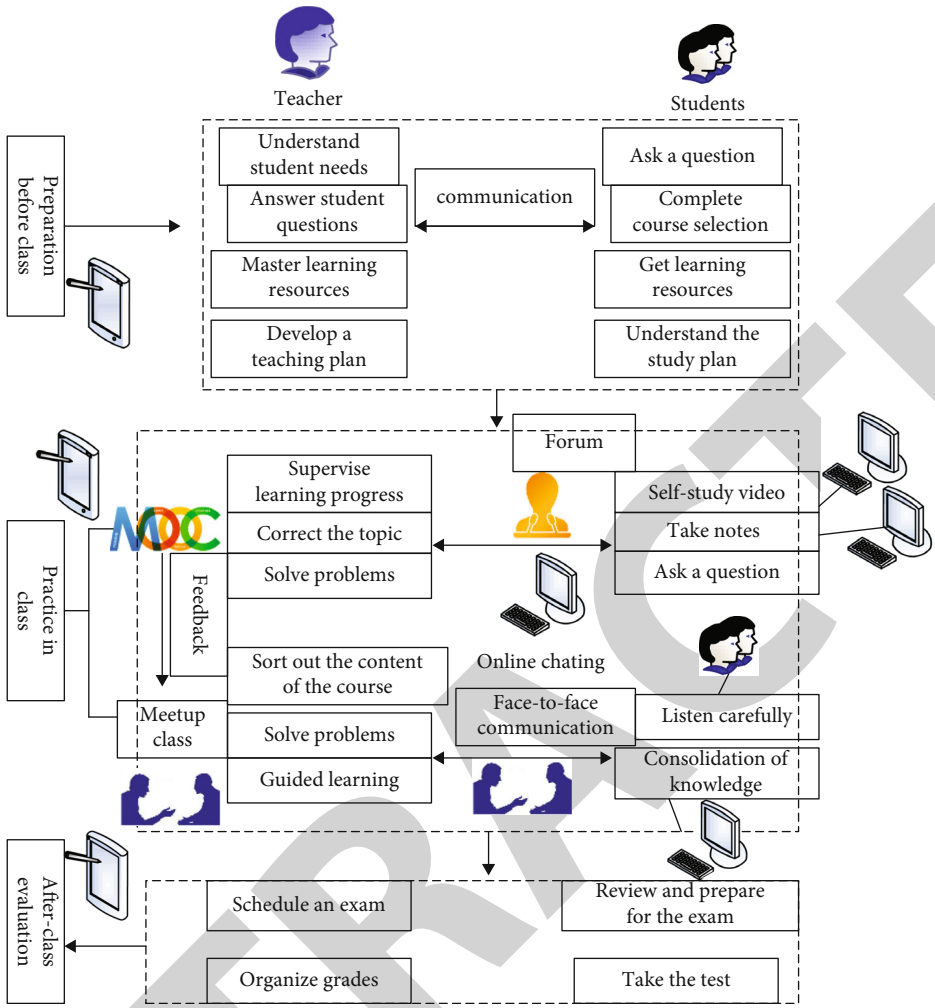

Figure 2: Course operation procedures.

Formula (12) is the penalty term.

$$E = \frac{1}{2}\sum_{a=1}^{a}\sum_{j=1}^{M}\ln\left[1+\left(g_j^a - p_j^a\right)^2\right], \tag{10}$$

$$E = \frac{1}{2}\sum_{a=1}^{a}\sum_{j=1}^{m}\left(g_j^a - p_j^a\right)^2 + \frac{1}{2}\sum_{a=1}^{a}\left(\sum_{j=1}^{m}\left(g_j^a - p_j^a\right)^2 \cdot \sum_{y=1}^{H}\left(h_{Ry} - 0.5\right)^2\right), \tag{11}$$

$$E = \frac{1}{2}\sum_{a=1}^{a}\sum_{j=1}^{m}\left(g_j^a - p_j^a\right)^2 + a(w), \quad a(w) = \frac{\beta}{t}\sum_{xy}\left|w_{xy}\right|^t. \tag{12}$$

2.2. Design of Online and Offline Hybrid Teaching Mode in MOOC Environment. To design a teaching model, it is necessary to formulate goals in the cognitive domain, motor skill domain, and emotional domain to cultivate good spatial imagination, engineering awareness, creative design ability, drawing and handwriting ability, and logical thinking ability for students [12]. It needs to consider teaching practice conditions, time, location, curriculum resources, teaching methods and means, etc. It includes practice in class and evaluation after class. Flow is shown in Figure 2.

Then, carry out the design of teaching evaluation, allowing students to participate in the evaluation of teachers anonymously. The school sets the dimensions and evaluation conditions. Although this method is single, the MOOC is extensive. It can be open and fair while requiring teaching quality and ability [13]. It can be anonymous or remove the highest and lowest evaluations and try to use the balance value to calculate. The mixed teaching mode in the MOOC environment is shown in Figure 3.

This design is based on the teaching survey function of a certain platform and uses a questionnaire survey mode to collect students' opinions. The premise is that the authenticity of students' evaluations of teachers is high, and no false evaluations are made. It advocates going out into the real world, avoiding shortcomings and accepting suggestions from students, thereby improving the quality of teaching [14].

2.3. Clustering Algorithm under DNN-Gaussian Mixture Model. Let a represent the Gaussian mixture model data vector of d-dimensional continuous values, I represent the

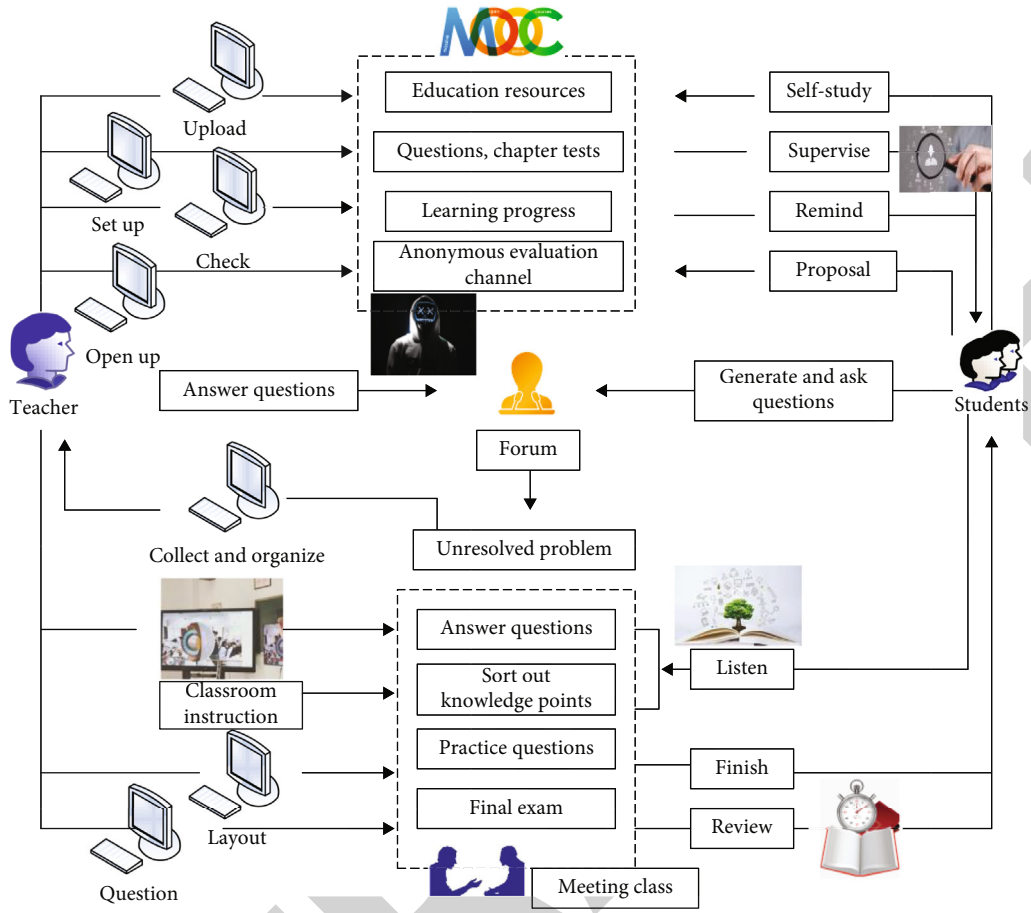

Figure 3: The mixed teaching model in the MOOC environment.

number of models, and G_i be the probability of being selected for the i-th type in the sample set [15]. In this case, $\sum_{i=1}^{I} \alpha_i \circ \varphi(a|\theta_i)$ is the Gaussian distribution density, representing the i-th Gaussian mixture model component, and its probability distribution model is as follows:

$$P(a|\theta) = \sum_{i=1}^{I} G_i \varphi(a|\theta),$$
$$\varphi(a|\theta_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a-\mu_i)^2}{2\sigma_i^2}\right). \tag{13}$$

In this formula, $\theta_i = (\mu_i, \sigma_i^2)$ and μ_i represent the average value, and σ_i^2 represents the variance. Its mean vector, variance matrix, and mixed weights are unified to form parameters. This process is completed by a complete Gaussian mixture model, and the parameters are expressed as

$$\theta = \{\mu_i, \sigma_i^2, G_i\}, \quad (i = 1, 2, \cdots, I). \tag{14}$$

The given training data vector in the Gaussian mixture model and the configuration of GMM are used to perform expectation maximization training on it using an iterative algorithm. Estimate the parameters of GMM and make the distribution of the feature vector trained by the Gaussian mixture model the best match under certain circumstances [16]. For M training vectors $A = \{a_1, a_2, \cdots, a_M\}$, assuming that each vector is independent of each other, the GMM likelihood can be expressed as

$$P = (A|\theta) = \prod_{m=1}^{M} P = (A_m|\theta). \tag{15}$$

The value of its likelihood function is calculated by the logarithm, and the expression of the likelihood function that maximizes its logarithm is as follows:

$$\max \ln p(A|\theta) = \sum_{m=1}^{M} \ln \sum_{i=1}^{I} G_i \varphi(A|\theta_i). \tag{16}$$

The formula is a nonlinear function of the parameters of the Gaussian mixture model. It is impossible to directly maximize the expectation of the function. Iterative method is used to maximize its likelihood to obtain an expectation maximization algorithm under a certain state [17]. Then, apply the EM algorithm, the principle of which is to estimate a new parameter from the original parameter, let its parameter be $\overline{\theta}$, and its value is greater than the original parameter.

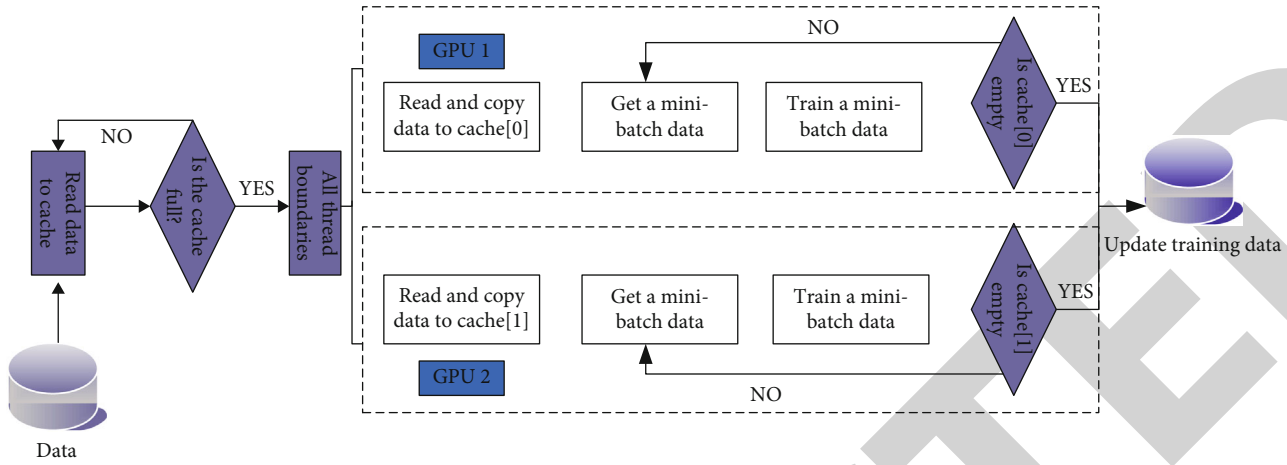

Figure 4: Minibatch parallel training data.

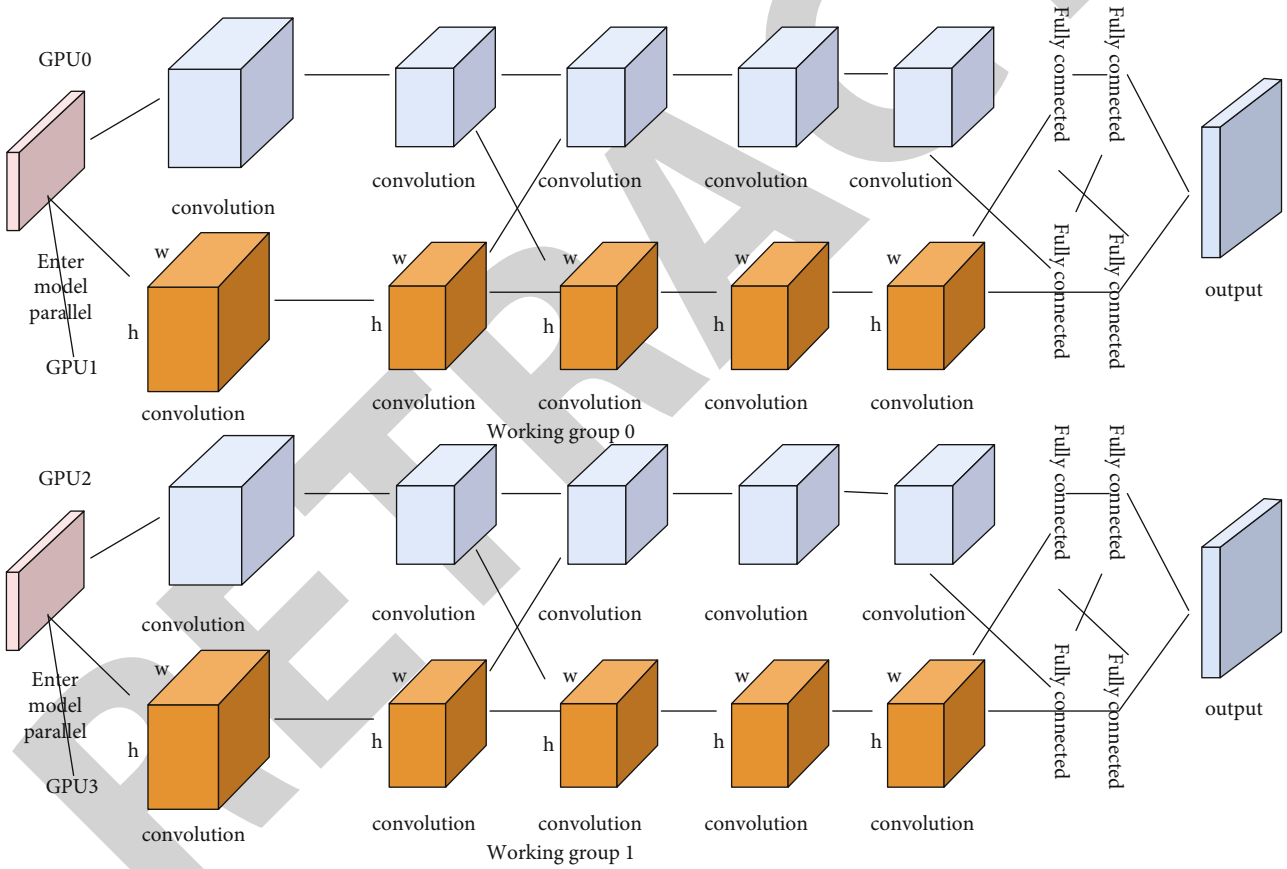

Figure 5: Hybrid design of two parallel methods.

The new parameters can be used as the initial parameters of the next iteration, and this operation is continued to be repeated to guide the acquisition of the best convergence threshold. This initial parameter can be estimated through some form of binary vector quantization. The number of clusters that has been set initially in the cluster is I, and the introduced variable represents the probability that the m-th training data comes from the i-th Gaussian mixture model. The probability can be calculated as

$$\Pr\left(i|a_m,\theta\right) = \frac{G_i\varphi(x|\theta_i)}{\sum_{m=1}^{I} G_i\varphi(a|\theta_i)}. \tag{17}$$

In each iteration of the EM algorithm, in order to ensure the monotonic increase of the parameter likelihood value,

the likelihood value can be recalculated using the following formula:

$$\overline{G_i} = \frac{1}{M}\sum_{m=1}^{M} \Pr(i|a_m, \theta),$$
$$\overline{\mu_i} = \frac{\sum_{m=1}^{M} \Pr(i|a_m, \theta) a_m}{\sum_{m=1}^{M} \Pr(i|a_m, \theta)}, \quad (18)$$
$$\overline{\sigma_i^2} = \frac{\sum_{m=1}^{M} \Pr(i|a_m, \theta) a_m}{\sum_{m=1}^{M} \Pr(i|a_m, \theta)} - \overline{\mu_i^2}.$$

In this formula, its iterative algorithm points to any element in the vector, respectively. Cluster analysis algorithms play an important role in various fields. Its effectiveness is formed by three different standards, namely, internal, external, and relative. The internal evaluation criterion is that the structure of the clustering algorithm and the performance characteristics of the dataset are regarded as the evaluation result. The data structure is in an unknown state. The relative evaluation criteria of clustering algorithms are formed by a variety of structures and manifested in different algorithms. The external clustering algorithm evaluation criterion is to select the best dataset clustering method through prior information, such as the number of clusters in the best state. The internal clustering algorithm can use the best clustering algorithm and dataset without additional information [18]. Cross-validation is widely used in supervised learning, and cross-validation can also be used for accuracy prediction in unsupervised learning. Obtain a Gaussian mixture model from a certain set of data, and calculate the number of clusters i. This article uses unsupervised learning evaluation criteria and other algorithm tests and uses the unsupervised clustering accuracy ACC to evaluate its performance:

$$\mathrm{ACC} = \max_{n} \frac{\sum_{k=1}^{m} 1\{l_k = nw_k\}}{M}. \quad (19)$$

In this formula, M represents the number of sample data, l_k represents the true label of the k-th data, the predicted value of the k-th data generated by the clustering algorithm is w_k, and n represents the prediction area between the cluster and the label.

3. The Parallelism of GPU-Based DNN Model and Application in MOOC Mixed Teaching Effect Dataset

3.1. Parallel and Optimization of GPU-Based DNN Models

3.1.1. Parallel Optimization Design Ideas. The pipelined BP algorithm is based on the division of training layers. The GPU in this algorithm is to define prebuf, curbef, and resbuf; curbef stores the current training dataset, and prebuf stores the dataset required for the next iteration. The previous iteration layer calculates, stores the calculation result in resbuf, and plays the role of transferring data [19]. Its model can be layered into multiple GPUs. When the dataset faced by DNNs is getting larger and larger, only the model cannot satisfy its training, and the data needs to be trained in batches. At this point, it can use minibatch to divide the training data to complete parallel training. The process is shown in Figure 4.

Table 1: Example of data after normalized sample 1.

Index/sample	1	2	3	4	5	6	7	8
$a1$	0.83	0.69	0.85	0.89	0.79	0.97	0.96	0.82
$a2$	0.94	0.93	0.68	0.71	0.91	0.72	0.71	0.85
$a3$	0.96	0.91	0.72	0.99	0.74	0.96	0.74	0.92
$a4$	0.68	0.99	0.92	0.75	0.84	0.71	0.75	0.92
$a5$	0.74	0.99	0.87	0.99	0.79	0.94	0.78	0.82
$a6$	0.86	0.7	0.92	0.69	0.81	0.95	0.92	0.7
$a7$	0.9	0.83	0.81	0.71	0.73	0.88	0.7	0.84
$a8$	0.86	0.87	0.69	0.75	0.85	0.81	0.95	0.88
$a9$	0.92	0.81	0.98	0.99	0.96	0.88	0.75	0.67
$a10$	0.99	0.85	0.79	0.9	0.73	0.73	0.7	0.94
$a11$	0.84	0.73	0.82	0.73	0.72	0.81	0.94	0.77
$a12$	0.79	0.75	0.79	0.8	0.73	0.88	0.98	0.71
$a13$	0.81	0.79	0.91	0.98	0.79	0.86	0.95	0.94
$a14$	0.8	0.79	0.93	0.97	0.88	0.68	0.81	0.96
$a15$	0.77	0.8	0.95	0.87	0.68	0.72	0.78	0.81
$a16$	0.8	0.87	0.74	0.81	0.9	0.98	0.87	0.9
$a17$	0.88	0.89	0.75	0.9	0.68	0.82	0.92	0.92
$a18$	0.99	0.83	0.69	0.94	0.69	0.67	0.67	0.92
$a19$	0.86	0.67	0.93	0.87	0.68	0.77	0.85	0.76
$a20$	0.85	0.82	0.97	0.89	0.75	0.85	0.94	0.85
$a21$	0.93	0.76	0.99	0.7	0.92	0.95	0.7	0.86
$a22$	0.98	0.73	0.67	0.78	0.7	0.94	0.87	0.96
$a23$	0.91	0.7	0.78	0.76	0.85	0.98	0.9	0.8
b	0.92	0.73	0.74	0.86	0.84	0.79	0.7	0.79

The data is in the training process; it is best to reduce the total communication overhead and parameter exchange period. It can reversely distribute the data trained by the GPU when shards are merged [20].

3.1.2. Parallel Optimization of Multiple Models. First, the DNN model can be split. When it is difficult to concentrate training on big data, it can be divided into small model blocks. Combining different models to calculate the weight parameters, in this state, the GPU will generate a relative calculation time, and the GPU only runs a part of the model. The entire system will mobilize the model operation engine to train the model. In this process, the data needs to be consistent to prevent deviations in the final results. Compared with data parallelism, model parallelism will produce a lot of consumption, resulting in poor acceleration effect [21]. Model and data parallelism have certain limitations on the GPU, and the learning rate needs to be reduced to achieve

a higher accuracy rate. Therefore, this article mixes the two training methods, as shown in Figure 5.

In a DNN, each level has a specific connection method. In general, full connections only exist in certain computational neural layers in the neural network structure, and there are parts that are independent of other connections. Training can be split on multiple GPUs. This behavior can reduce the forward and backward propagation time of training .

3.1.3. Minibatch Parallel Optimization Training. The three-level thread structure of GPU has a large number of computing cores, which is suitable for intensive training such as DNN models. The use of GPU training can improve training efficiency. For the wear resistance of DNNs, the characteristics can be optimized. DNNs favor gradient descent algorithms for learning. It can be seen that when the gradient of the batch data is calculated, the network parameters need to be updated. Using batch processing will increase the GPU running speed. Using its characteristics of good floating point operations, the matrix and vector algorithms of DNNs are transformed into matrix and matrix algorithms. When selecting batch data, select the set composed of training data randomly. DNNs do not require ordered data, so they can meet their characteristics. When training GPU, we follow the principle of double GPU twice as single GPU. Under certain circumstances, increasing the data batch will form certain advantages. The graphics memory and computing core on the GPU can be used to improve the efficiency of convolution calculations. The decrease in the number of data iterations during training will increase the processing speed for the same amount of data, resulting in larger batches and higher accuracy, resulting in smaller training shocks. The premise is that the batch size is too large to reduce the test performance. It can be seen that this improved parallel method can improve the performance during model training.

3.2. Experimental Preparation and the Optimized Model's Effect on Big Data Processing. This article uses GCC compiler to compile C++ code, NVCC compiler to compile GPU code, the number of single GPU graphics card cores is 2880, the basic frequency is 710 MHz, the dynamic speedup frequency is 876 MHz, and the memory capacity is 12 GB. Obtain online and offline MOOC evaluation data from 2016 to 2020 from the educational administration system of a university. Dataset format ($a1, a2, \cdots, a22, b$), a total of 20880 sample data. Remove the higher and lower ones and those that are inconsistent with the facts, leaving 16962 sample data. This experiment is improved based on caffe and compared with the improved caffe. The GPU library for accelerating machine learning with cuDNN mainly highlights the performance and memory overhead. In this experiment, cuDNN is used to accelerate the GPU matrix. After normalization of some samples, Tables 1 and 2 can be obtained.

The experimental dataset is divided into three parts: A, B, and C, and the three parts have the same amount of data. These datasets have been preprocessed during the collection process and can be directly used for DNN learning. The test network structure is a comparative test based on the network structures of LeNet, CaffeNet, and GoogleNet. Use the A dataset for experiments on the LeNet network structure, while CaffeNet corresponds to B and GoogleNet corresponds to C. The CaffeNet network structure is shown in Figure 6.

Table 2: Examples of data after normalized sample 2.

Index/sample	1	2	3	4	5	6	7	8
$x1$	0.81	0.89	0.7	0.76	0.97	0.84	0.76	0.86
$x2$	0.86	0.76	0.71	0.9	0.85	0.85	0.95	0.96
$x3$	0.7	0.8	0.95	0.68	0.71	0.88	0.96	0.92
$x4$	0.98	0.95	0.84	0.87	0.93	0.77	0.8	0.99
$x5$	0.81	0.88	0.69	0.75	0.73	0.76	0.85	0.75
$x6$	0.69	0.83	0.92	0.95	0.83	0.87	0.73	0.86
$x7$	0.9	0.69	0.99	0.9	0.89	0.99	0.87	0.87
$x8$	0.83	0.88	0.78	0.94	0.9	0.8	0.84	0.85
$x9$	0.83	0.74	0.97	0.68	0.93	0.83	0.7	0.79
$x10$	0.84	0.83	0.94	0.84	0.71	0.84	0.81	0.8
$x11$	0.69	0.93	0.98	0.72	0.95	0.85	0.76	0.94
$x12$	0.73	0.93	0.72	0.9	0.82	0.69	0.98	0.8
$x13$	0.67	0.73	0.95	0.89	0.79	0.67	0.76	0.83
$x14$	0.84	0.97	0.71	0.85	0.98	0.82	0.9	0.76
$x15$	0.96	0.67	0.99	0.74	0.79	0.96	0.87	0.78
$x16$	0.92	0.75	0.83	0.89	0.75	0.76	0.7	0.68
$x17$	0.73	0.86	0.9	0.88	0.85	0.7	0.71	0.86
$x18$	0.75	0.82	0.82	0.71	0.94	0.7	0.82	0.96
$x19$	0.69	0.71	0.7	0.7	0.89	0.67	0.96	0.87
$x20$	0.83	0.98	0.72	0.67	0.97	0.81	0.83	0.73
$x21$	0.96	0.85	0.86	0.91	0.91	0.99	0.95	0.82
$x22$	0.93	0.95	0.94	0.71	0.68	0.83	0.68	0.99
$x23$	0.86	0.91	0.99	0.9	0.84	0.88	0.86	0.91
y	0.95	0.79	0.75	0.9	0.79	0.87	0.87	0.8

Caffe does have advantages when dealing with small-scale datasets. However, in the case of large dataset processing, there is no better effect. Here, we select a large-scale MOOC online and offline mixed curriculum effect evaluation dataset and use the optimized DNN model for testing.

3.3. Experimental Test and Result Analysis. We will perform performance testing and comparison. In this experiment, 120000, 8000, and 4000 iterations of training were performed on the three datasets, respectively, and the training time comparison and training accuracy comparison of the improved caffe and improved caffe were compared. The test result is shown in Figure 7.

When testing the A group of data, it can be seen that with 120000 iterations, the time required for caffe to complete takes 19.8 seconds. The parallel solution based on the dual GPU model only takes 12.4 seconds, and its speedup

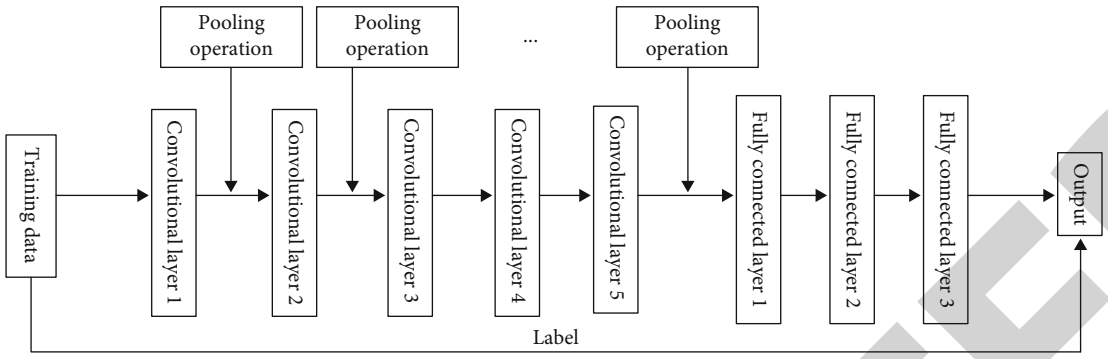

Figure 6: CaffeNet network structure.

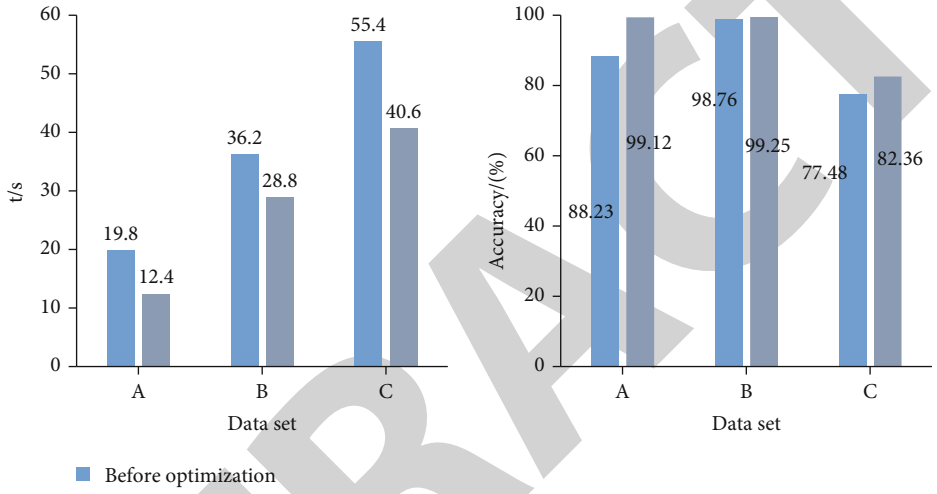

Figure 7: Comparison of the time and accuracy required to complete training before and after optimization.

ratio reaches 37.37%. The accuracy rate is also as high as 99.12%, a year-to-year increase of 12.34% in the DNN model training. In the iterative training process, this article will further analyze the accuracy and convergence of the algorithm on the A dataset for continuous DNN learning. The results are shown in Figure 8.

The figure shows that the accuracy of parallel training rises faster and the convergence is faster, while the convergence and accuracy rise rate of caffe is smaller. At the end of the training, the loss value of the optimized DNN algorithm approaches 0 infinitely.

When testing the data of group B, it can be seen that with 8000 iterations, the time required for caffe to complete takes 36.2 seconds. The parallel scheme based on the dual GPU model only takes 28.8 seconds, its speedup ratio reaches 20.44%, and the accuracy rate is as high as 99.25%. In the DNN model training, the year-to-year increase was 0.50%. During the iterative training process, this article will further analyze the accuracy and convergence of the continuous DNN learning algorithm on the B dataset. The result is shown in Figure 9.

The model in this figure converges faster than caffe from beginning to end in the iterative process. It can be seen that the parallel model has an excellent effect.

When testing the C group data, it can be seen that in the case of 4000 iterations, the time required for caffe to complete takes 55.4 seconds. The parallel solution based on the dual GPU model only takes 40.6 seconds, its speedup ratio reaches 26.71%, and the accuracy rate is as high as 82.36%. In the DNN model training, an increase of 6.30% year on year. During the iterative training process, this article will further analyze the accuracy and convergence of the continuous DNN learning algorithm on the C dataset. The results are shown in Figure 10.

The model in the figure also converges faster than caffe from beginning to end in the iterative process, and the accuracy is significantly higher. It can be seen that the effect of the parallel model is excellent.

Then, it can be trained to obtain the comparative error of the actual evaluation result of the experimental data evaluation results, as shown in Table 3.

It can be seen that the error has been very small and is completely within the acceptable range.

It can be seen that after applying the dual GPU model parallelism, its DNN model has greatly improved the training effect of the MOOC online and offline mixed course effect dataset. The calculation time is shortened, the convergence speed is accelerated, and the accuracy rate is

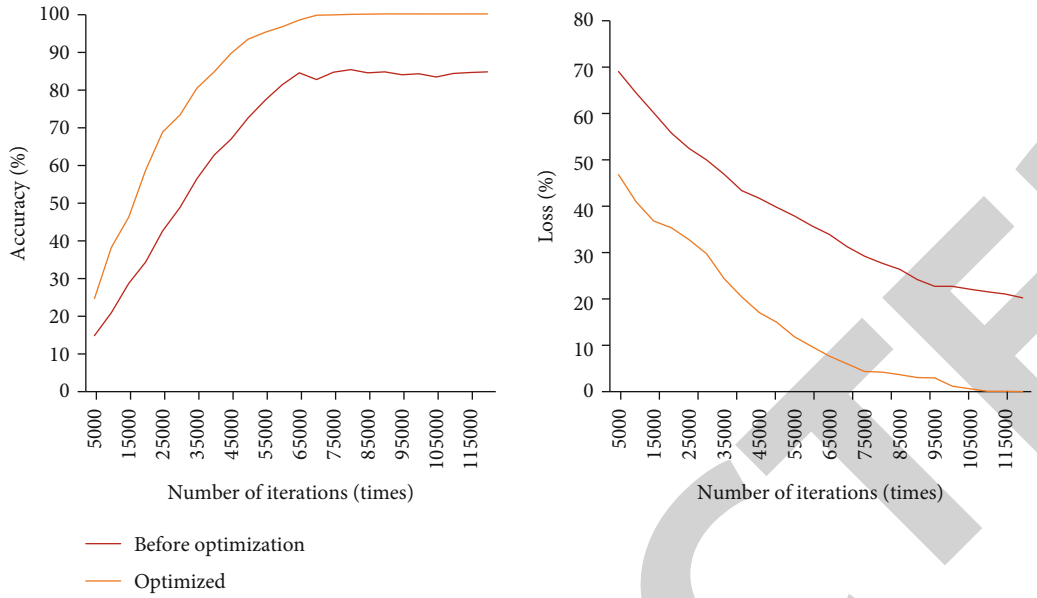

Figure 8: Comparison of accuracy and loss before and after optimization of A dataset training.

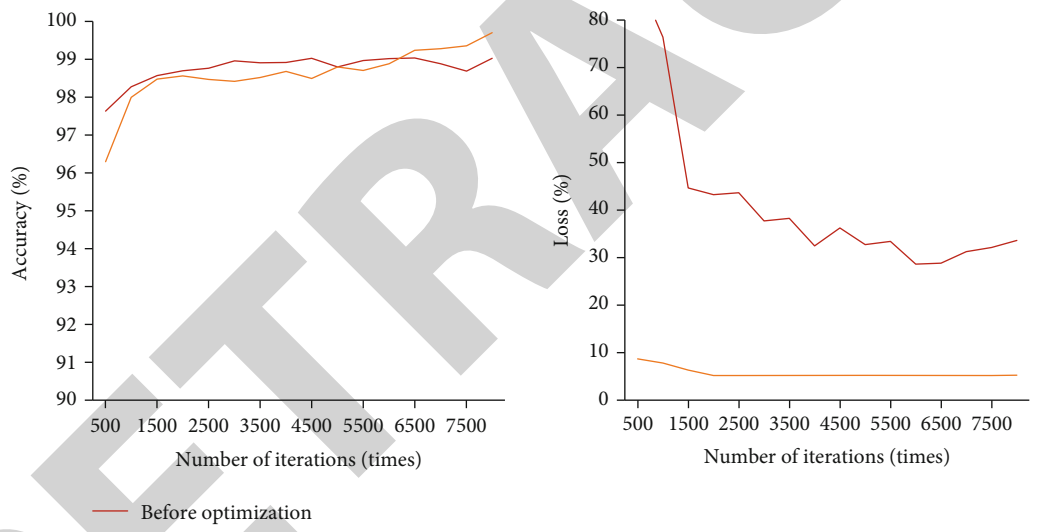

Figure 9: Comparison of accuracy and loss before and after optimization of B dataset training.

improved. Its optimization model parallel method has increased the speedup ratio by more than 37.37%, and the accuracy can be increased by more than 12.34%.

4. Discussion

This article focuses on the evaluation of the online and offline mixed teaching effect of MOOC based on the DNN model. This article first explains the related concepts of the deep network model. This article has already described the application of MOOC in the teaching field. Then, this article explains some DNN models and algorithms, such as BP neural network model and algorithms. BP neural network is a backpropagation network. Multilayer prefeedback neural network can be trained by error reversal propagation algorithm, including input layer, hidden layer, and output layer. The improved BP algorithm can use additional momentum factors to reduce oscillations and speed up convergence. This paper designs a mixed-mode teaching in a MOOC environment, including course operation procedures and teaching modes. This article explains the clustering algorithm based on DNN-Gaussian mixture model. It calculates the probability density function with Gaussian component density weighted sum and trains the function value with expect maximization. Finally, this paper discusses the parallelism of GPU-based DNN model and its application in MOOC mixed teaching effect dataset. In this paper, we can draw conclusions from parallel optimization ideas and processes with experimental testing of its optimization effects. The error caused by the parallel mode training dataset is very small, the speedup ratio before optimization is increased by more than 37.37%, and the accuracy can be increased by

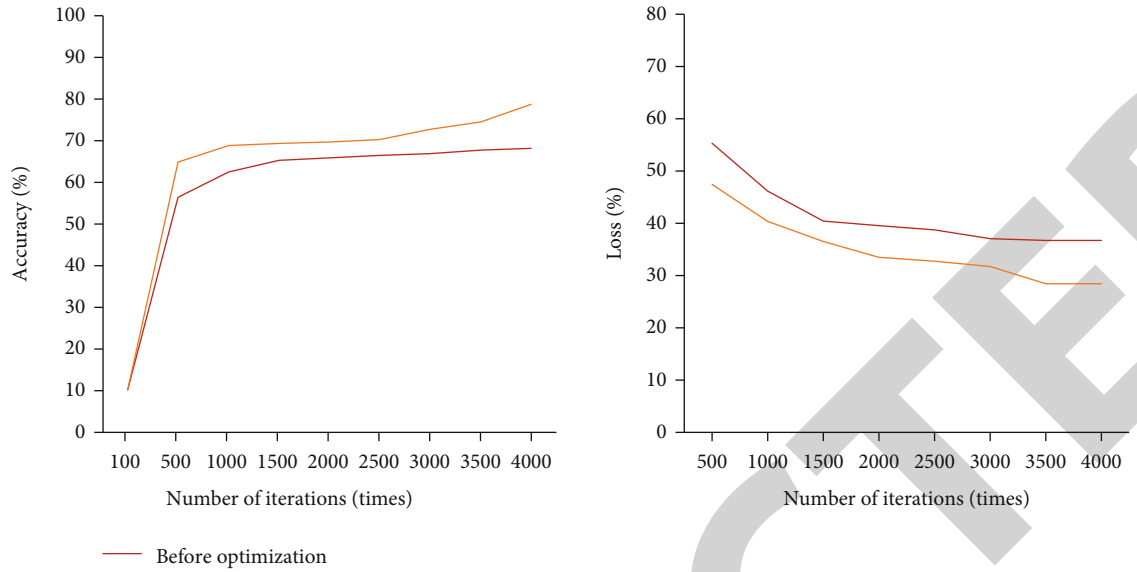

Figure 10: Comparison of accuracy and loss before and after optimization of C dataset training.

Table 3: Comparison of experimental optimization model, test data, and actual data.

	1	2	3	4	5	6	7	8
Actual evaluation results	0.921	0.811	0.785	0.623	0.830	0.846	0.672	0.859
Model test results after optimization	0.923	0.812	0.786	0.625	0.827	0.847	0.675	0.862
Deviation	0.002	0.001	0.001	0.002	0.003	0.001	0.003	0.003

more than 12.34%. It can be seen that after the optimization of this DNN model, there will be a good development prospect in the evaluation of the online and offline mixed teaching effect of MOOC and other fields.

5. Conclusion

For the improvement of the deep network model to test the effect of online and offline teaching in MOOC, this article describes the design ideas of parallel optimization. GPU uses curbef to store the current training dataset, prebuf to store the dataset required for the next iteration, and resbuf to store the calculation results. Use minibatch to divide the training data and then optimize it. The model training and data training methods are mixed and designed to obtain an optimized deep network model, and then, the experiment design is carried out. Use the 2016 to 2020 online and offline MOOC evaluation data of a school to divide it into several datasets for testing. Test and analyze the model performance before and after optimization. The time required for caffe to complete is longer than that of the parallel solution based on the dual GPU model, and the accuracy is not enough. The convergence speed of the parallel scheme based on the dual GPU model is also faster. And by comparing with the actual data, it is found that the error formed by the test result is very small. It can be seen that this dual GPU model parallel scheme is very feasible.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by "Research and Practice on the Effect of MOOC Teaching in Application-Oriented Universities" (2016JGZ170) Teaching Team of Guilin University of Aerospace Technology (2020JXTD03).

References

[1] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 1–13, 2016.

[2] J. Li, M. Xue, and D. Prokhorov, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 690–703, 2017.

[3] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3263–3277, 2015.

[4] L. Cai, X. Tian, and C. Sheng, "Monitoring nonlinear and non-Gaussian processes using Gaussian mixture model-based weighted kernel independent component analysis," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 28, no. 1, pp. 122–135, 2017.

[5] A. Roy, S. K. Parui, and U. Roy, "SWGMM: a semi-wrapped Gaussian mixture model for clustering of circular–linear data," *Pattern Analysis and Applications*, vol. 19, no. 3, pp. 631–645, 2016.

[6] J. Chen, K. Li, Z. Tang et al., "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Transactions on Parallel & Distributed Systems*, vol. 28, no. 4, pp. 919–933, 2017.

[7] M. Carrasco, J. López, and S. Maldonado, "Epsilon-nonparallel support vector regression," *Epsilon-nonparallel support vector regression. Applied Intelligence*, vol. 49, no. 12, pp. 4223–4236, 2019.

[8] X. Xu, D. Cao, Y. Zhou, and J. Gao, "Application of neural network algorithm in fault diagnosis of mechanical intelligence," *Mechanical Systems and Signal Processing*, vol. 141, p. 106625, 2020.

[9] S. Schwartz, J. J. Montero Jimenez, M. Salaün, and R. Vingerhoeds, "A fault mode identification methodology based on self-organizing map," *Neural Computing and Applications*, vol. 32, no. 17, pp. 13405–13423, 2020.

[10] C. H. Chen, F. Song, F. J. Hwang, and L. Wu, "A probability density function generator based on neural networks," *Physica A: Statistical Mechanics and its Applications*, vol. 541, article 123344, 2020.

[11] A. A. A. Mohd Amiruddin, H. Zabiri, S. A. A. Taqvi, and L. D. Tufa, "Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems," *Neural Computing and Applications*, vol. 32, no. 2, pp. 447–472, 2020.

[12] T. Kawabata and K. Yura, "1P253 multiple protein docking guided by low-resolution image of complex using Gaussian mixture model under the symmetric constraint (bioinformatics-structural genomics, poster presentations)," *Oecologia*, vol. 86, no. 1, pp. 140–143, 2007.

[13] B. Zhao, Y. Zhong, A. Ma, and L. Zhang, "A spatial Gaussian mixture model for optical remote sensing image clustering," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 12, pp. 5748–5759, 2016.

[14] D. Wang, Z. Li, L. Cao et al., "Image fusion incorporating parameter estimation optimized Gaussian mixture model and fuzzy weighted evaluation system: a case study in time-series plantar pressure data set," *IEEE Sensors Journal*, vol. 17, no. 5, pp. 1407–1420, 2017.

[15] E. Putin, P. Mamoshina, A. Aliper et al., "Deep biomarkers of human aging: application of deep neural networks to biomarker development," *Aging*, vol. 8, no. 5, pp. 1021–1033, 2016.

[16] U. Shaham, A. Cloninger, and R. R. Coifman, "Provable approximation properties for deep neural networks," *Applied and Computational Harmonic Analysis*, vol. 44, no. 3, pp. 537–557, 2018.

[17] Q. Yu, Y. Yang, F. Liu, Y. Z. Song, T. Xiang, and T. M. Hospedales, "Sketch-a-Net: a deep neural network that beats humans," *International Journal of Computer Vision*, vol. 122, no. 3, pp. 411–425, 2017.

[18] Y. B. Wang, Z. H. You, X. Li et al., "Predicting protein–protein interactions from protein sequences by a stacked sparse auto-encoder deep neural network," *Molecular BioSystems*, vol. 13, no. 7, pp. 1336–1344, 2017.

[19] G. Chen, H. Meng, Y. Liang, and K. Huang, "GPU-accelerated real-time stereo estimation with binary neural network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2896–2907, 2020.

[20] S. Tang and F. Yu, "Construction and verification of retinal vessel segmentation algorithm for color fundus image under BP neural network model," *The Journal of Supercomputing*, vol. 77, no. 4, pp. 3870–3884, 2021.

[21] Y. J. Liang, C. Ren, H. Y. Wang, Y. B. Huang, and Z. T. Zheng, "Research on soil moisture inversion method based on GA-BP neural network model," *International Journal of Remote Sensing*, vol. 40, no. 5-6, pp. 2087–2103, 2019.