Research Article
A Novel Restricted Boltzmann Machine Training Algorithm with Fast Gibbs Sampling Policy

Qianglong Wang, Xiaoguang Gao, Kaifang Wan, Fei Li, and Zijian Hu

School of Electronic and Information, Northwestern Polytechnical University, Xi’an 710129, China

Correspondence should be addressed to Kaifang Wan; wankaifang@nwpu.edu.cn

Received 9 November 2019; Revised 12 February 2020; Accepted 21 February 2020; Published 20 March 2020

Copyright © 2020 Qianglong Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The restricted Boltzmann machine (RBM) is one of the widely used basic models in the field of deep learning. Although many indexes are available for evaluating the advantages of RBM training algorithms, the classification accuracy is the most convincing index that can most effectively reflect its advantages. RBM training algorithms are sampling algorithms essentially based on Gibbs sampling. Studies focused on algorithmic improvements have mainly faced challenges in improving the classification accuracy of the RBM training algorithms. To address the above problem, in this paper, we propose a fast Gibbs sampling (FGS) algorithm to learn the RBM by adding accelerated weights and adjustment coefficient. An important link based on Gibbs sampling theory was established between the update of the network weights and mixing rate of Gibbs sampling chain. The proposed FGS method was used to accelerate the mixing rate of Gibbs sampling chain by adding accelerated weights and adjustment coefficients. To further validate the FGS method, numerous experiments were performed to facilitate comparisons with the classical RBM algorithm. The experiments involved learning the RBM based on standard data. The results showed that the proposed FGS method outperformed the CD, PCD, PT5, PT10, and DGS algorithms, particularly with respect to the handwriting database. The findings of our study suggest the potential applications of FGS to real-world problems and demonstrate that the proposed method can build an improved RBM for classification.

1. Introduction

Deep learning (DL) is an important branch of machine learning that employs complex model structures or different nonlinear transformation methods to conduct high-dimensional abstract feature modeling of data [1]. DL structures such as the deep neural network, convolutional neural network, and generative adversarial neural network have been widely used in computer vision, automatic speech recognition, natural language processing, audio recognition, and bioinformatics [2]. Deep belief network (DBN) is a generative graphical model, which can be regarded as a type of deep neural network [3] that can perform classification tasks excellently [4]. DBNs are stacked with restricted Boltzmann machines (RBMs), and the whole network can be trained by a greedy layer-wise learning algorithm from the bottom to the top [5]. Therefore, the effect of RBM training will directly affect the quality of the DBN.

RBM is an important model employed in DBNs and is one of the widely used models of the Markov random field (MRF). It was proposed by Smolensky in 1986 based on the Boltzmann machine (BM) [6]. The RBM is a stochastic generative neural network that can learn the probability distribution from the input datasets. RBMs are applied in dimension reduction [7], classification [8], collaborative filtering [9], feature learning [10], topic modeling [11], radar target automatic recognition [12], chip synthesis [13], and speech recognition [14]. RBMs can be trained by either supervised or unsupervised learning depending on the different tasks. DL and RBM have obvious advantages for massive unlabeled data, particularly in the context of big data. The contrastive divergence (CD) algorithm proposed by Hinton et al. in 2002 is one of the most popular standard algorithms for RBM training [15]. It uses K step sampling chain to approximate the target gradient. The CD algorithm initializes the Gibbs chain...
samples from the training set samples. In general, one-step Gibbs sampling (sampling step $K = 1$) is sufficient to acquire an impressive result strategy. However, as the classification accuracy of the CD algorithm is not optimal, many refined learning algorithms for RBM training have been introduced, which adopt different sampling techniques and aim to reduce the bias in the gradient approximation [16]. The persistent contrastive divergence (PCD) algorithm proposed by Tieleman in 2008 is also based on Gibbs sampling. However, instead of using the training data to reinitialize the sampling chain in each iteration, PCD employs the results of the previous sampling for continuous operation [17]. To accelerate the PCD algorithm, the fast persistent contrastive divergence (FPCD) algorithm was proposed by Tieleman et al. in 2009 [18], with the introduction of additional fast parameters to improve the sampling speed. In 2010, Desjardins et al. [19] and Cho et al. [16] proposed the parallel tempering algorithm (PT) based on sampling technology. The PT algorithm was designed to overcome the limitations of traditional sampling algorithms in multimode target distribution. Experiments have shown that the PT algorithm can achieve better generative models for RBM training while effectively preventing any possible divergence. The number of hidden units can also be changed for better performance [20]. In recent years, improvements in RBM algorithms have been mushrooming. Li et al. proposed the dynamic Gibbs sampling (DGS) algorithm in 2016, which can effectively overcome the problem of sampling divergence and obtain a higher training accuracy at the cost of a short computation time [21].

At present, the training of RBMs is primarily based on Gibbs sampling. Effective improvement in the training of RBMs has proved challenging and has been an active focus area of algorithmic improvements. To solve this problem, the fundamentals of RBMs and classification accuracies of traditional algorithms are introduced in Section 2. In Section 3, theoretical analysis is performed and the important relationship between the update of network weights and mixing rate of Gibbs sampling chain is determined based on Gibbs sampling theory. Two important conclusions are deduced according to Gibbs sampling theorem. In Section 4, we first focus on the relationship between the weights of the classical algorithm and mixing rate based on Gibbs sampling theory while attempting to improve Gibbs mixing rate. The accelerated weights and adjustment coefficient are then added based on the original weight, and the fast Gibbs sampling (FGS) algorithm is proposed, with the advantages of the improved algorithm proved theoretically. Finally, in Section 5, the results of numerous simulation experiments are analyzed in detail. The experimental results indicate that the proposed fast Gibbs sampling algorithm achieves a good stationary distribution of the RBM model in a short time and accelerates the mixing rate of Gibbs sampling chain in the sampling process, thus significantly improving the classification accuracy of the RBM. The results of this study provide the theoretical guidance for improving the RBM training algorithm in future studies.

2. Preliminaries

2.1. RBM Model and Training Strategy. RBM is a typical Markov random field model, which has a two-layer network structure. The network consists of visible units and hidden units, which correspond to visible variables (data samples) and hidden variables, respectively [22]. Binary RBM is a type of RBM wherein the variables of the visible and hidden layers are binary variables. In this paper, we will introduce and employ the binary RBM as an example. The entire RBM network is a bipartite graph, in which connections only exist between the visible and hidden units. There is no connection within the visible units and hidden units. Owing to this condition, RBM has the following properties: given the state of the visible layer neurons, the activation conditions of the neurons of each hidden layer are independent, and vice versa. The structure of an RBM is shown in Figure 1.

The training process of RBMs [23] involves the following steps:

**Step 1** (modeling). Modeling includes the energy function and maximum probability hypothesis. RBMs are energy-based models, with maximum likelihood as the learning goal. The energy of the joint configuration of its visible variables $v$ and hidden variables $h$ is given by

$$E(v, h; \theta) = -\sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j.$$  \hspace{1cm} (1)

$\theta$ represents the parameter $\{W, a, b\}$. The joint probability of $v$ and $h$ can be obtained by

$$P_{\theta}(v, h) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta)).$$  \hspace{1cm} (2)

Here, $Z(\theta)$ is called the partition function. The above equation can also be written as

$$P_{\theta}(v, h) = \frac{1}{Z(\theta)} \exp \left( \sum_{i=1}^{D} \sum_{j=1}^{F} W_{ij} v_i h_j + \sum_{i=1}^{D} v_i b_i + \sum_{j=1}^{F} h_j a_j \right).$$  \hspace{1cm} (3)

The objective is to maximize the likelihood function $P(v)$. $P(v)$ can be calculated by the edge distribution of $P(v, h)$:

$$P_{\theta}(v) = \frac{1}{Z(\theta)} \sum_h \exp \left[ v^T Wh + a^T h + b^T v \right].$$  \hspace{1cm} (4)

**Step 2** (parameter estimation). The maximum likelihood estimate (MLE) method is used for parameter estimation. $P(v)$ is maximized to obtain the RBM parameters. To maximize $P(v)$, we maximize $\log(P(v)) = L(\theta)$:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_{\theta}(v^{(n)}).$$  \hspace{1cm} (5)

**Step 3** (extreme value solution). The gradient descent algorithm is mainly used to find the extreme value solution.
First, $L(\theta)$ is maximized via stochastic gradient descent. Subsequently, the $L(\theta)'$ derivative for $W$ is estimated:

$$
\frac{\partial L(\theta)}{\partial W_{ij}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial W_{ij}} \log \left( \sum_h \exp \left[ (v^{(n)})^T Wh + (a^T h + b^T v^{(n)}) \right] \right) \\
- \frac{\partial}{\partial W_{ij}} \log Z(\theta) \\
= E_{P_{\text{data}}}[v|h] - E_{P_{\theta}[v|h]}.
$$

The first half of the formula can be calculated easily. The average values of $v_i$ and $h_i$ from all of the datasets are calculated. However, the latter half of the equation involves all $2^{n+h}$ combinations of $v$ and $h$, which is difficult to solve owing to the computational complexity. The latter half of the formula is equal to

$$
\sum_{v_i h_i} v_i h_i P_{\theta}(v, h).
$$

As the analytical solution of the gradient cannot be computed, the existing training algorithms are mainly based on the sampling method. First, a Markov chain with a stationary distribution $P(v)$ is constructed to obtain the samples satisfying the distribution. The gradient is then approximated via Monte Carlo simulations:

$$
\Delta a_i = v_i^{(0)} - v_i^{(k)}, \\
\Delta b_i = P(h_j = 1 | v^{(0)}) - P(h_j = 1 | v^{(k)}), \\
\Delta W_{ij} = P(h_j = 1 | v^{(0)})v_i^{(0)} - P(h_j = 1 | v^{(k)})v_i^{(k)},
$$

where $v_i^{(0)}$ is the sample value and $v_i^{(k)}$ is the sample that satisfies the distribution $P(v)$ obtained by sampling. Finally, the parameter update equation is given by

$$
a_i = a_i + \Delta a_i, \\
b_j = b_j + \Delta b_j, \\
W_{ij} = W_{ij} + \Delta W_{ij}.
$$

2.2. Classification Accuracy. Many indicators are available for evaluating the performance of RBM training algorithms. Different performance indicators are selected according to different requirements: indicators can represent (1) convergence errors and convergence accuracy based on the convergence characteristics of the Gibbs sampling chain, (2) computation time and algorithm efficiency based on the algorithm complexity, and (3) classification accuracy compared with the actual application of the RBM.

The convergence characteristics of the Gibbs sampling chain can represent the degree of convergence of the training algorithm to a certain extent. It can only be used to assess the training algorithm qualitatively, but it is not an accurate indicator for comparing the training algorithm [22]. The training duration is directly proportional to the complexity of the training algorithm. The simpler the training algorithm is, the shorter the training duration would be. Although the training time is short, a high classification accuracy is not achieved in the final application. Therefore, the training time can serve as an auxiliary index to evaluate the training algorithm. This implies that indicators that are more convincing are necessary. As a composite index, the algorithm efficiency depends on the negative value of the algorithm running time and algorithm training error product [24]. Decreasing either the running time or training error can improve the algorithm efficiency. The algorithm efficiency can also be used as an auxiliary indicator to assess the quality of the algorithm. The classification accuracy is the most intuitive and convincing indicator for evaluating the RBM training algorithm. The RBM trained by the algorithm is introduced into the actual dataset for testing. The accuracy of the test results represents the number of successfully classified data divided by the total number of data.

In this paper, four traditional RBM training algorithms were simulated using the standard MNIST handwritten dataset. The four algorithms are CD with classic single Gibbs chain sampling, PCD that improves the initial value of each Gibbs chain sampling on the CD algorithm, and PT (PT5 and PT10) for processing multiple Gibbs chain samples with multimode problems. The classification accuracy of the four algorithms is shown in Table 1.

As shown in Table 1, the CD algorithm has a relatively low classification accuracy, while that of PCD is higher by approximately 2%, leading to improved classification. However, the PT algorithm, which addresses multimode problems better, has approximately the same classification accuracy as the CD algorithm when evaluating handwritten datasets. As the bias vectors of both the visible and hidden layers have minimum values in the RBM network, their update relative to the update of the network weights is negligible. Therefore, only the influence of the network weight $W$ is studied when evaluating the network parameters. Figure 2 shows the updating of the network weights of the four traditional RBM algorithms.

Figure 2 clearly shows that the PCD algorithm follows the process of increasing and attenuating the weights after changing the initial value of each Gibbs sampling chain.
Table 1: Classification accuracies of four traditional RBM training algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>CD</th>
<th>PCD</th>
<th>PT5</th>
<th>PT10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification accuracy (%)</td>
<td>85.80</td>
<td>87.46</td>
<td>84.10</td>
<td>85.82</td>
</tr>
</tbody>
</table>

Figure 2: Updating the network weights of the four traditional RBM algorithms.

3. Convergence Theorem for Gibbs Sampling

It is well established that the classification accuracy index can directly reveal the performance of the RBM training algorithm. The performance of the RBM training algorithm based on Gibbs chain sampling depends on the sampling property of Gibbs chain, i.e., the mixing rate of the Gibbs sampling chain. In this paper, we analyze the relationship between the update of the network weight and mixing rate of Gibbs sampling chain. Furthermore, this indirectly analyzes the relationship between the mixing update of the RBM network weight and performance of the RBM training algorithm, to assess the final classification accuracy.

Theorem 1. The mixing rate of Gibbs sampling chain gradually decreases as the randomness of the transfer operator decreases [25].

The RBM network represents a typical two-layer network structure with no connections within the layer and full connections between the layers. Owing to its special network structure, the conditional independence within the layers of the RBM indicates that the output layer node value satisfies the following conditional probability when the input layer data are provided:

$$P(h_k = 1 | v) = \frac{1}{1 + \exp(-b_j - \sum_{i=1}^{m} W_{ij} v_i)}$$

$$= \text{Sigmoid}\left( b_j + \sum_{i=1}^{m} W_{ij} v_i \right).$$

(10)

Correspondingly, when the output layer data are determined, the conditional probability of the input layer node value is

$$P(v_k = 1 | h) = \frac{1}{1 + \exp(-a_i - \sum_{j=1}^{n} W_{ij} h_j)}$$

$$= \text{Sigmoid}\left( a_i + \sum_{j=1}^{n} W_{ij} h_j \right).$$

(11)

The RBM network is based on Gibbs iteration of the above two equations to update the network parameters. As a transfer operator function of mixing with Gibbs sampling, the randomness of the sigmoid function will affect the mixing rate of the entire Gibbs sampling chain. To study the randomness of the transfer operator, sigmoid curves under different weights are provided in Figure 3. In this paper, any change in the weight generally refers to the absolute change in the weight.

The sigmoid curves become steeper with an increase in the network weight. The following two conclusions can be drawn:

Corollary 1. When the variation range of $X$ approaches infinity, the sigmoid function gradually becomes less random with an increase in the weights.

Derivation: the sigmoid function diagram shows that when the variation range of $X$ is large and approaches infinity, a larger weight results in a steeper curve. If the curve is sampled during this interval, the value of most sampling areas transferred by the transfer operator will be either 0 or 1, and the modified region, i.e., the adjustable region of the curve in the plot, is very small. Thus, there is a very small change in state before and after sampling, indicating that the randomness of the transfer operator is very small.

Corollary 2. When the variation range of $X$ approaches zero, the sigmoid function gradually becomes increasingly random with an increase in the weight.
Derivation: the schematic of the sigmoid function shows that when the variation range of $X$ is small and approaches 0, a smaller weight results in a flatter curve. Furthermore, the variable sampling area is very small, which results in a very small change in state before and after sampling. The curve becomes increasingly steeper with increase in the absolute value of the weights. Therefore, the variation range of the transition after sampling is larger, and there is a larger change in state before and after sampling, indicating that the randomness of the transfer operator is also very large.

4. Fast Gibbs Sampling Algorithm

4.1. Analysis Based on Gibbs Sampling Theory. In this section, we will analyze the classical Gibbs sampling algorithm based on Gibbs sampling theory to identify the relationship between the mixing rate of the Gibbs sampling chain and network weights. Unless otherwise specified, all weights referred to in this paper are the mean values of the network weights during each iteration.

4.1.1. Gibbs Chain Mixing Rate Analysis of the CD Algorithm. Figure 2 shows that the absolute value of the weight of the CD algorithm increases with the number of iterations. The weights corresponding to different iterations were equally spaced, and a schematic of the sigmoid function was drawn corresponding to the different weight parameters, as shown in Figure 4. To show the transfer randomness of the sigmoid function under different weights clearly, the image in Figure 4 was enlarged, as shown in Figure 5. Figure 2 shows that the absolute value of the weight of the CD algorithm increases with an increase in the number of iterations when training the RBM. Figure 5 and Corollary 2 indicate that the randomness of the sigmoid function continually increases, while the randomness of the transfer operator corresponds to the mixing rate of the Gibbs chain. However, at the end of training (i.e., when the number of iterations $n = 1000$), the transfer randomness of the sigmoid function is only $\sim 0.72$. The following conclusion can be drawn considering the classification accuracy of the CD algorithm presented in Table 1 (Section 2.2). Although the absolute value of the weight of the CD algorithm increases continuously with the network training, the randomness of the sigmoid transfer operator continually increases, as the maximum transfer probability of the transfer operator is only $\sim 0.72$, which limits the Gibbs sampling chain. The mixing rate eventually achieves a classification accuracy of only $\sim 85\%$ for the CD algorithm.
4.1.2. Gibbs Chain Mixing Rate Analysis of the PCD Algorithm. The PCD algorithm changes the initial value of the Gibbs chain for each sample based on the CD algorithm. Figure 2 shows that the weight of the PCD algorithm is significantly different from that of the CD algorithm. First, the absolute value of the weight of the PCD algorithm initially increases and then slowly decreases until the end. In this paper, the weights corresponding to different iteration times were equally spaced, and the corresponding sigmoid function for different weights was plotted, as shown in Figure 6. A part of Figure 6 was enlarged to clearly show the transition probability of the sigmoid function under different weights (Figure 7). By comparing the transfer operators of the sigmoid function corresponding to different weights and iterations (Figure 7), it can be observed that the randomness of the transfer operator of the PCD algorithm increases rapidly with the training, with the probability increasing rapidly from 0.79 to 0.94. The randomness then slowly decreases and remains at \( \sim 0.92 \). Combined with the classification accuracy of the PCD algorithm (Table 1, Section 2.2), the following conclusions can be drawn: compared with the CD algorithm, the absolute value of the weight of the PCD algorithm increases rapidly at the beginning of training, then slowly decreases, and remains stable in the later stages, accompanied by an initial increase in the transition probability. The rate of increase is very fast, although it can remain at \( \sim 0.92 \) in the later stage, indicating that the PCD algorithm is more random than the transfer operator of the CD algorithm is. The mixing ratio of the Gibbs chain is also higher than that of the CD algorithm. Therefore, the classification accuracy of the PCD algorithm (87.46%) is also considerably higher than that of the CD algorithm (85.80%).

4.1.3. Gibbs Chain Mixing Rate Analysis of the PT Algorithm. In addition to the single-chain Gibbs sampling algorithm, there are some typical multichain Gibbs sampling algorithms, such as PT5 and PT10. In this paper, the PT5 algorithm is introduced as an example. The change in the weight of the multichain Gibbs sampling algorithm has an important impact on the Gibbs mixing rate. Figure 2 shows that the absolute weights of the PT algorithms (PT5 and PT10) increase slowly with the number of iterations and then slowly decrease. The weights corresponding to different iterations were selected according to the same spacing, and the sigmoid function under the corresponding weights was plotted, as shown in Figure 8. A part of Figure 8 was enlarged to show the transition probability of the sigmoid function under different weights clearly (Figure 9). Based on Figure 9 and Corollary 2, it can be seen that the randomness of the sigmoid function of the PT algorithm first increases and subsequently decreases. The change in the randomness of the transfer operator corresponds to the mixing rate of the Gibbs chain that first increases and then decreases. At the end of training, the transfer probability of the sigmoid function is stable at \( \sim 0.34 \). Considering the classification accuracy of PT (Table 1, Section 2.2), the following conclusions can be drawn: although the PT algorithm can address multimode problems better than the CD algorithm, the randomness of the sigmoid transfer operator remains low due to less changes in its weight with network training, which highly limits the mixing rate of the Gibbs sampling chain. According to Theorem 1, the classification accuracy of the PT5 algorithm should be considerably lower than that of the CD algorithm (85.80%). However, as the PT algorithm involves a temperature mixing chain exchange to improve the mixing rate of the multichain Gibbs sampler [17], the classification accuracy of the PT5 algorithm is \( \sim 84.10\% \), which is slightly lower than that of the CD algorithm.
4.1.4. Summary. Based on the above theoretical analysis, it can be seen that the core factor of the RBM training algorithm based on Gibbs sampling is the convergence property of the Gibbs sampling chain, i.e., the sample mixing rate. The change in network weights is an important factor affecting the mixing rate of Gibbs chain sampling. From Corollary 2, it can be seen that the mixing ratio of the PCD algorithm improves owing to an increase in the network weights; however, the improved weighing of the PCD algorithm depends on the increase in weights proposed in Corollary 2, with the probability of the transfer operator not reaching 1. In this paper, the aim was to further improve the mixing rate of the algorithm through Corollary 1, to improve the classification accuracy.

4.2. Accelerated Weight and Adjustment Coefficient. Based on the theoretical deduction of the above analysis, it can be established that the classical algorithm can be improved by improving the weight of convergence theorem 2 of Gibbs sampling. Although there is significant improvement in the mixing rate of the Gibbs sampling chain of the improved algorithm and in the classification accuracy by 2%, we found that the probability of the transfer operator of the improved PCD algorithm was only 0.94. Therefore, the aim was to modify the weight update based on the classical CD algorithm to improve the mixing rate of the Gibbs sampler. The weight update should satisfy the following two criteria:

(1) The update of weights must increase rapidly in the initial stage. According to Corollary 2 derived from the convergence theorem of Gibbs sampling, a rapid increase in the weights at the beginning is beneficial for increasing the randomness of the transfer operator, leading to a rapid increase in the mixing rate of the Gibbs chain with the update of the weights.

(2) The update of weights should show a decreasing trend in the mid and late stages of the training period. As the weight increases, the corresponding X interval of the transfer operator will increase accordingly. According to Corollary 1 derived from the convergence theorem of Gibbs sampling, when the transfer probability of the transfer operator is 1, the adjustable domain of the transfer operator continually decreases if the absolute value of the weight increases upon sampling (Figure 3). This implies that the mixing rate of Gibbs chain decreases. Therefore, if the absolute value of the weight reduces after the transition probability of the transfer operator reaches 1, the weighting of the transfer operator will increase its unable domain. This will further improve the mixing rate of the Gibbs chain, thereby improving the performance of the algorithm.

Based on the above analysis, this paper proposes a fast Gibbs sampling algorithm that introduces an accelerated weight fastW and adjustment coefficient $\xi$. The accelerated weight introduced in this paper is updated in the same way as the traditional weights. The accelerated weight update also depends on the difference between the expectation of data and the expectation of model. However, the weight of the entire network is the sum of the accelerated and traditional weights, which effectively ensures that the weight update increases rapidly in the early stage of training. The adjustment coefficient added in this study can effectively change the accelerated weight update rate. The adjustment coefficient can be varied in the range of 0 to 1, which can effectively decrease the trend of accelerated weight updating in the mid and later stages of training. By introducing the accelerated weight and adjustment coefficient, the proposed FGS algorithm can improve the transition probability of the
transfer operators and mixing rate of the Gibbs chains, thus improving the classification accuracy of the algorithm.

The equation used for updating the weight is given as follows.

Similar to the traditional RBM training algorithm, the input data \( v \) is sampled to obtain the hidden layer data. Here, \( v^+ = v^- \), both of which represent the input data:

\[
\begin{align*}
  h^+ &= P(h | v^+, W), \\
  h^- &= P(h | v^-, W + \text{fast}W).
\end{align*}
\]

After calculating the hidden layer data, the positive and negative gradients are updated:

\[
\begin{align*}
  W^+ &= v^+ h^+, \\
  W^- &= v^- h^-.
\end{align*}
\]

Then, the new input layer data can be calculated by the hidden layer data. The weights are the sum of the traditional and accelerated weights:

\[
v = P(v | h^+, W + \text{fast}W).
\]

The gradient for the weight updates is composed of two parts: traditional weight update and accelerated weight update:

\[
\Delta W = W^+ + W^-.
\]

The traditional and accelerated weight updates are governed by the equations shown below. The accelerated weight update in this paper is controlled by the adjustment coefficient \( \xi \), and the trend change can be effectively controlled by the adjustment.

\[
\begin{align*}
  W &= W + \Delta W, \\
  \text{fast}W &= \xi \ast \text{fast}W + \Delta W.
\end{align*}
\]

Based on the above equations and algorithm description, the pseudocode of the FGS algorithm proposed in this paper is provided in Algorithm 1.

5. Simulations and Analysis

The simulations and analysis were performed in two steps. The first step involved the design of the experiment, which includes the design details of each group of experiments. The second step involved the display and analysis of the simulation results.

5.1. Experimental Design. The experimental design mainly involves the following three steps: (1) details of each dataset used in the experiments, (2) setup of the RBM network structure and parameters in the simulation experiment, and (3) the design of the comparative experiment.

5.1.1. Benchmark Dataset. Five standard datasets were used to test our algorithm, namely, MNIST (Figure 10), MNORB (Figure 11), CIFAR10 (Figure 12), CIFAR100 (Figure 13), and USPS (Figure 14).

5.1.2. Network Structure and Parameters. The RBM network structure designed in this study and initializations of each network parameter in different datasets are shown in Tables 2–6. Here, \( S \) represents the RBM structure in different datasets, \( a \) is the bias of the visible layer, \( b \) is the bias of the hidden layer, \( \lambda \) is the learning rate, \( W \) represents the traditional weight, and \( \text{fast}W \) represents the proposed fast weight.

5.1.3. Design of the Comparative Experiment. To comprehensively compare the FGS algorithm with other traditional RBM training algorithms, the CD and PCD algorithms were selected owing to their better sampling. We also compared FGS with the PT5 and PT10 algorithms with respect to the multimode distribution problem. The experimental parameters of the six algorithms are shown in Table 7. Here, \( M \) represents number of iterations of the PT algorithm and \( K \) is the number of Gibbs sampling iterations, Batch represents the training sample size of a single batch, and Iter represents the number of network iterations.

(1) Experiment 1: Mixing Rate of Gibbs Chain. As the focus of this paper was to improve the RBM training algorithm based on Gibbs chain sampling, the convergence property of Gibbs sampling chain is a crucial indicator of the improved performance of the algorithm. The convergence property of Gibbs sampling chain depends on its mixing rate. The mixing rate of the Gibbs chain can be measured by the randomness of the transfer operator using Theorem 1. Here, MNIST was used as the test dataset and the randomness of the transfer operator of the FGS algorithm was evaluated. The experimental results and analysis are presented in 5.2.1.

(2) Experiment 2: Effect of Sampling. Considering the MNIST dataset as an example, the algorithm based on Gibbs sampling was used to reconstruct the data after sampling [22]. The reconstructed graphs of each training algorithm were saved and compared with the original image to observe the performance of each algorithm intuitively. The experimental results and analysis are presented in 5.2.2.

(3) Experiment 3: Classification Accuracy. A key assessment of the effectiveness of an RBM training algorithm involves the comparison of the classification accuracy of the network. Considering the MNIST dataset as an example, the softmax classifier (Figure 15) was added to the output layer of the trained RBM network, and the data of the hidden layer of the RBM network was used as the input to train the classifier. The output of the top layer ranged from 0 to 9, covering ten categories. The following process was adopted: the corresponding binary output was obtained via the trained network parameters following the input of a new set of binary data. These binary outputs obtained through the RBM served as the inputs of the classifier, and they were trained in accordance with the labels of each set of data. Subsequently, the test data were inputted to the network and the classification accuracy was determined. Softmax is a generalization of the logistic regression model.
Input: RBM \((v, h)\), training batch \(S\)

Output: gradient approximation \(\Delta W, \Delta a_i, \Delta b_j\) for \(i = 1, \ldots, m, j = 1, \ldots, n\)

1. Init \(\Delta W = \Delta a_i = \Delta b_j = 0\)
2. Init \(W\) to small regular values, init fast\(W\) to all zeros
3. For all \(v \in S\) do
   4. \(v_0 \leftarrow v\)
   5. for \(t = 0, \ldots, k - 1\) do
      6. for \(j = 1, \ldots, n\) do sample
         7. \(h_t^{(j)} \sim p(h_t | v_0^{(j)}, W)\), calculate: \(W^t = v_0^{(j)} h_t^{(j)}\)
      8. end
      9. for \(j = 1, \ldots, n\) do sample
         10. \(h_t^{(j)} \sim p(h_t | v_0^{(j)}, W + \text{fast}\!W),\) calculate: \(W^t = v_0^{(j)} h_t^{(j)}\)
   11. end
   12. for \(i = 1, \ldots, m\) do sample
      13. \(v_{t+1} (i) \sim p(v_{t+1} | h_t(i), W + \text{fast}\!W)\)
      14. end
   15. for \(j = 1, \ldots, n, i = 1, \ldots, m\) do
      16. \(\Delta W \leftarrow \Delta W + p(h_i = 1 | v_0(i)) * v_i^{(0)} - p(h_i = 1 | v_i^{(k)}) * v_i^{(k)}\)
      17. \(W = W + \Delta W\)
      18. fast\!W = \(\xi * \text{fast}\!W + \Delta W\)
   19. end
20. end
21. for \(i = 1, \ldots, m\) do
   22. \(\Delta a_i \leftarrow \Delta a_i + v_0^{(i)} - v_i^{(k)}\)
   23. end
24. for \(j = 1, \ldots, n\) do
   25. \(\Delta b_j \leftarrow \Delta b_j + p(h_j = 1 | v_0^{(b)}) - p(h_j = 1 | v_i^{(k)})\)
   26. end
27. end

\(m\) and \(n\) represent the number of neurons in the visible and hidden layers, respectively; \(v\) and \(h\) represent the state vectors of the visible and hidden layers, respectively; \(a\) and \(b\) represent the bias vectors of the visible and hidden layers, respectively; \(W\) represents the weight matrix between the hidden and visible layers; fast\!W represents the accelerated weight; \(\xi\) is the adjustment coefficient; and \(\Delta W, \Delta a_i, \Delta b_j\) represent the update gradients of the weights and biases.

**Algorithm 1:** Pseudocode of the proposed FGS algorithm.

**Figure 10:** The MNIST dataset, which is a standard handwritten dataset used for training the RBM [26]. It contains 60,000 training samples and 10,000 test samples, wherein each sample is a grayscale image with \(28 \times 28\) pixels.

**Figure 11:** The MNORB dataset. This dataset contains 19,440 training sets and 24,300 test sets [27], wherein each sample is a grayscale image with \(32 \times 32\) pixels.

(4) **Experiment 4: Training Time.** The training time is another important criterion used to assess the training algorithm. In general, the network training time increases with increasing complexity of the algorithm. In cases where similar classification accuracies are achieved, the performance of an algorithm decreases with increase in the training time.

for multiclassification problems and is an important classification model in supervised learning. The detailed experimental results and analysis of the classification accuracy are presented in Section 5.2.3.
Figure 12: The CIFAR10 dataset, which contains color maps with $32 \times 32 \times 3$ pixels. We used 10,000 training samples and 10,000 test samples of this dataset [28].

Figure 13: The CIFAR100 dataset, which contains color maps with $32 \times 32 \times 3$ pixels. The CIFAR100 dataset used in this paper consists of 10,000 training sets and 10,000 test sets [28].

Figure 14: The USPS handwritten dataset derived from the US Postal Service handwritten numeral recognition library. It contains grayscale images of “0” through “9,” where each image is $16 \times 16$ pixels. This dataset contains 1,100 separate examples for each category.

Table 2: Structure and parameters of MNIST.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Str</td>
<td>$784 \times 500$</td>
</tr>
<tr>
<td>a</td>
<td>zeros(1, 784)</td>
</tr>
<tr>
<td>b</td>
<td>zeros(1, 500)</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.1</td>
</tr>
<tr>
<td>W</td>
<td>$0.1 \times \text{rand}(784, 500)$</td>
</tr>
<tr>
<td>fastW</td>
<td>zeros(784, 500)</td>
</tr>
</tbody>
</table>

Table 3: Structure and parameters of MNORB.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Str</td>
<td>$1024 \times 700$</td>
</tr>
<tr>
<td>a</td>
<td>zeros(1, 1024)</td>
</tr>
<tr>
<td>b</td>
<td>zeros(1, 700)</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.001</td>
</tr>
<tr>
<td>W</td>
<td>$0.1 \times \text{rand}(1024, 700)$</td>
</tr>
<tr>
<td>fastW</td>
<td>zeros(1024, 700)</td>
</tr>
</tbody>
</table>

Table 4: Structure and parameters of CIFAR10.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Str</td>
<td>$3072 \times 2000$</td>
</tr>
<tr>
<td>a</td>
<td>zeros(1, 3072)</td>
</tr>
<tr>
<td>b</td>
<td>zeros(1, 2000)</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.01</td>
</tr>
<tr>
<td>W</td>
<td>$0.1 \times \text{rand}(3072, 2000)$</td>
</tr>
<tr>
<td>fastW</td>
<td>zeros(3072, 2000)</td>
</tr>
</tbody>
</table>

Table 5: Structure and parameters of CIFAR100.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Str</td>
<td>$3072 \times 2000$</td>
</tr>
<tr>
<td>a</td>
<td>zeros(1, 3072)</td>
</tr>
<tr>
<td>b</td>
<td>zeros(1, 2000)</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.001</td>
</tr>
<tr>
<td>W</td>
<td>$0.1 \times \text{rand}(3072, 2000)$</td>
</tr>
<tr>
<td>fastW</td>
<td>zeros(3072, 2000)</td>
</tr>
</tbody>
</table>

Table 6: Structure and parameters of USPS.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Str</td>
<td>$256 \times 200$</td>
</tr>
<tr>
<td>a</td>
<td>zeros(1, 256)</td>
</tr>
<tr>
<td>b</td>
<td>zeros(1, 200)</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.1</td>
</tr>
<tr>
<td>W</td>
<td>$0.1 \times \text{rand}(256, 200)$</td>
</tr>
<tr>
<td>fastW</td>
<td>zeros(256, 200)</td>
</tr>
</tbody>
</table>

Table 7: Parameter settings of the comparative experiment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>CD</th>
<th>PCD</th>
<th>FGS</th>
<th>PT5</th>
<th>PT10</th>
<th>DGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>5</td>
<td>10</td>
<td>—</td>
</tr>
<tr>
<td>$K$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1,5,10</td>
</tr>
<tr>
<td>Batch</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Iter</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>
Detailed experimental results and analysis of the training duration are presented in Section 5.2.4.

(5) Experiment 5: Adjustment Coefficient. In this paper, the adjustment coefficient $\xi$ was added to the designed FGS algorithm to adjust the fast weight. The adjustment coefficient was varied to analyze its effect on the whole network. In this experiment, MNIST was considered as the example dataset. The adjustment coefficient values are shown in Table 8, and the corresponding experimental results and analysis are detailed in 5.2.5.

5.2. Experimental Results and Analysis. According to the comparative experiment designed in Section 5.1.3, the following section presents the corresponding experimental results and analysis based on the experimental design.

5.2.1. Mixing Rate of Gibbs Chain. Figure 16 shows the network weights of the FGS algorithm proposed in this paper. The results showed that the fast weight and adjustment coefficient added in this study could help analyze the weights detailed in Section 4 of this paper. In the early stage of training, the absolute value of the weight of the FGS algorithm increased rapidly, while it decreased gradually in the mid-later stages of training.

The weight parameters corresponding to different iterations were selected based on an equal interval, with the corresponding sigmoid function shown in Figure 17. To show the transition probability of the sigmoid function clearly, a part of the image in Figure 17 was enlarged, as shown in Figure 18. Figure 17 shows that, in the initial stage of network training, the absolute value of the weight increased rapidly, which caused the transition probability of the transfer operator to increase rapidly from 0.8 at 200 steps to 1 at 400 iterations, resulting in a high probability value. According to Corollary 1 of the convergence theorem for Gibbs sampling, the randomness of the transfer operator tends to decrease if the absolute value of the weight continues to increase during this interval. However, after comprehensively considering the relationship between the weight and randomness of the transfer operator, the FGS algorithm proposed in this paper can solve the above-mentioned problem by adding the adjustment coefficient of the fast weight. Figure 16 shows that the absolute weight of the FGS algorithm decreased in the mid to later stages of training, synonymous with conclusions 1

![Table 8: Value of the adjustment coefficient.](image)

<table>
<thead>
<tr>
<th>Parameter range</th>
<th>$\xi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \rightarrow 0.01$</td>
<td>0.001</td>
</tr>
<tr>
<td>$0.01 \rightarrow 0.1$</td>
<td>0.01, 0.05</td>
</tr>
<tr>
<td>$0.1 \rightarrow 0.5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$0.5 \rightarrow 0.9$</td>
<td>0.5</td>
</tr>
<tr>
<td>$0.9 \rightarrow 1$</td>
<td>0.9, 0.95</td>
</tr>
<tr>
<td>$1$</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 16: Comparison between the weights of the CD and FGS algorithms.](image)

![Figure 15: Classification process of the RBM with softmax.](image)
and Figure 17 of the Gibbs sampling convergence theorem. In the mid and later stages of training, although the probability of the transfer operator reached the highest value of 1, as the absolute value of the weight decreased, the curve of the transfer operator became relatively flat and its adjustable domain increased. Therefore, the randomness of the transfer operator continued to increase between 400 and 1000 iterations. Thus, FGS algorithm has a significantly higher Gibbs mixing rate compared with the CD algorithm.

5.2.2. Effect of Sampling. A more intuitive criterion for evaluating the quality of RBM training involves the image reconstructed by the sampling algorithm. Figure 19 shows the original image, while Figures 20–25 illustrate the reconstructed images obtained using the six sampling
algorithms. The figures show that the network images reconstructed using the FGS algorithm proposed in this paper exhibited less noise and high reconstruction precision, indicating the better training effect of FGS compared with the other algorithms.

5.2.3. Classification Accuracy. One of the most important criteria for assessing the effectiveness of RBM training is the classification accuracy with respect to the input dataset. Table 9 shows a comparison between the classification accuracies of the six algorithms for the five datasets.

(1) Analysis of the Classification Accuracy. The FGS algorithm proposed in this paper performed very well on the five commonly used datasets, namely, MNIST, MNORB, CIFAR10, CAFAR100, and USPS. Using the same RBM network structure and experimental parameters, FGS achieved the highest classification accuracy of 90.02% on the MINIST dataset, demonstrating that the improved algorithm could accurately identify the handwritten dataset. Second, the traditional RBM training algorithms, CIFAR10 and CIFAR100, did not effectively recognize the datasets with color images, as feature extraction is relatively difficult owing to these images having many pixels. The improved FGS algorithm achieved good recognition accuracy for these two colorful datasets. Subsequently, the classification accuracy was compared for the grayscale images in the MNORB dataset. The FGS algorithm achieved the best classification results in this case under the same experimental conditions. Finally, all algorithms were tested on the USPS dataset, which resulted in the proposed FGS achieving the highest classification accuracy. In summary, under the same experimental conditions and RBM network structure, the designed FGS algorithm designed could achieve an improved classification accuracy.

5.2.4. Training Time. Comparison between the training times of the six groups of experiments for the MNIST (Figure 26), MNORB (Figure 27), CIFAR 10 (Figure 28), CIFAR 100 (Figure 29), and USPS (Figure 30) datasets is shown in the following figures. It can be clearly seen that the training time of PT5 and PT10 was considerably longer than that of the other four algorithms. To compare the training times of the other four algorithms, comparison graphs of the CD, PCD, FGS, and DGS algorithms for the MNIST (Figure 31), MNORB (Figure 32), CIFAR 10 (Figure 33), CIFAR 100 (Figure 34), and USPS (Figure 35) datasets are illustrated.

(2) Analysis of the Training Time Comparisons. Two graphs were constructed to compare the training times for each dataset. The plots on the left compare the training times of all six algorithms for each dataset. It can be clearly seen that the training time of the PT5 and PT10 algorithms was longer than that of the other four algorithms. This is because PT algorithms have multiple temperature chains, with complex and frequent interactions increasing the time. Therefore, a comparison between the training times of the remaining four algorithms is shown in the figures on the right. Based on these figures and Table 10, the following conclusions can be drawn: under the same experimental conditions, the CD algorithm can be trained in the shortest time, as it is the simplest implementation of Gibbs sampling. As the PCD algorithm only changes the initial value of each iteration of Gibbs sampling, it mostly requires a similar time to train as CD. Although the FGS algorithm adds the accelerated weight and adjustment coefficient compared with the CD algorithm, it is significantly less complex than the multichain PT algorithm. Therefore, this indicates that the FGS algorithm has a shorter training time compared with the PT5 and PT10 algorithms. The DGS algorithm is a phased dynamic Gibbs sampling algorithm. As the training progressed, the number of Gibbs sampling steps increased to ten. Multiple iterations of a single sampling chain improved the recognition accuracy at the cost of increased time. FGS required
less time than DGS in most cases, despite being approximately 0.3 s slower with respect to the USPS dataset. Based on the results presented in 5.2.3, the accuracy of the FGS algorithm was considerably higher than that of the CD algorithm. This indicates that the improved FGS algorithm proposed in this paper can achieve a significant improvement in the accuracy relative to the CD algorithm at the cost of sacrificing a small amount of time, thus demonstrating a
significant improvement in the mixing rate of Gibbs sampling.

5.2.5. Adjustment Coefficient. The designed FGS algorithm included the adjustment coefficient $\xi$ with fast weights. We conducted a comparative experiment on the MNIST handwritten dataset to investigate the influence of the adjustment coefficients on training the RBM network. Based on the experimental parameters presented in Table 6, different values were selected for different ranges of the adjustment coefficient, with the reconstruction error chosen as the evaluation index. Figure 36 shows a comparison between

the reconstruction errors of each group of experiments, with the corresponding recognition accuracies shown in Table 11. The reconstruction error is the difference between the original data and training samples following a Gibbs transfer through the RBM distribution [22]. Assuming that the mini-batch data size is the batch size, $v_1$ is the initial state, and $v_2$ is the reconstructed state after one sampling interval, then the equation for calculating the reconstructed error is given by

$$\text{error} = \frac{\text{sum}(\text{sum}(v_1 - v_2)^2)}{\text{batchsize}}$$

(17)

In Figure 36, it can be observed that when different values are selected, the reconstruction error gradually converges with the number of iterations, indicating that the choice of the adjustment coefficient does not affect the convergence of the entire network.
Figure 32: Comparison between the training times of four algorithms on the MNORB dataset.

Figure 33: Comparison between the training times of four algorithms on the CIFAR10 dataset.

Figure 34: Comparison between the training times of four algorithms on the CIFAR100 dataset.
Figure 35: Comparison between the training times of four algorithms on the USPS dataset.

Table 10: Comparison between the training times of the six algorithms on each dataset.

<table>
<thead>
<tr>
<th>Algorithm (unit)</th>
<th>MNIST (s)</th>
<th>MNORB (s)</th>
<th>CIFAR10 (s)</th>
<th>CIFAR100 (s)</th>
<th>USPS (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>8.345</td>
<td>7.865</td>
<td>74.006</td>
<td>75.616</td>
<td>1.416</td>
</tr>
<tr>
<td>PCD</td>
<td>19.548</td>
<td>10.881</td>
<td>30.702</td>
<td>116.394</td>
<td>1.437</td>
</tr>
<tr>
<td>PT5</td>
<td>280.185</td>
<td>304.399</td>
<td>1.282 × 10³</td>
<td>2.267 × 10³</td>
<td>66.302</td>
</tr>
<tr>
<td>PT10</td>
<td>2.986 × 10³</td>
<td>308.346</td>
<td>2.462 × 10³</td>
<td>4.453 × 10³</td>
<td>130.527</td>
</tr>
<tr>
<td>DGS</td>
<td>46.385</td>
<td>22.337</td>
<td>181.419</td>
<td>181.577</td>
<td>2.343</td>
</tr>
<tr>
<td>FGS</td>
<td>14.041</td>
<td>10.798</td>
<td>159.562</td>
<td>159.555</td>
<td>2.689</td>
</tr>
</tbody>
</table>

Figure 36: Comparison of reconstruction errors for different adjustment coefficient.

Table 11: Comparison of the classification accuracies of different adjustment coefficients.

<table>
<thead>
<tr>
<th>Adjustment coefficient $\xi$</th>
<th>0</th>
<th>0.001</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.5</th>
<th>0.9</th>
<th>0.95</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification accuracy</td>
<td>86.2</td>
<td>87.72</td>
<td>88.84</td>
<td>88.24</td>
<td>88.16</td>
<td>86.54</td>
<td><strong>90.02</strong></td>
<td>88.4</td>
<td>82.16</td>
</tr>
</tbody>
</table>
Table 11 shows that the classification accuracy of the entire algorithm is relatively high when the adjustment coefficient is approximately 0.9, indicating that the mixing rate for Gibbs sampling is the highest for the whole network. Based on the data in Table 8, it can be seen that the selection of the adjustment coefficient would affect the accuracy of the algorithm. The recognition accuracy of the whole network will be relatively high when the adjustment coefficient is in the range 0.9–0.95.

6. Conclusion

The aim of this study was to improve the classification accuracy of the RBM training algorithm by formulating an analysis based on Gibbs sampling theory. The important relationship between the update of the network weights and mixing rate of Gibbs sampling chain was established, resulting in two important conclusions based on the convergence theorem of Gibbs sampling. Experimental analysis revealed that the mixing rate of Gibbs chain can be improved for classical algorithms. In this study, the accelerated weights and adjustment coefficient of the fast Gibbs sampler were included to improve the update of the training weights. Furthermore, the proposed FGS algorithm improved the mixing rate of Gibbs sampling chain and controlled the training time. Compared with the classic RBM training algorithm, the proposed FGS algorithm achieved a higher classification accuracy. In this paper, five control trials were simulated using five experimental datasets (MNIST, CIFAR10, CIFAR100, MNORB, and USPS), which were assessed based on the mixing ratio of Gibbs chain, sampling comparison, classification accuracy, training duration, and hyperparameters. The experimental results showed that the proposed FGS algorithm had good sampling convergence properties. FGS can reduce the training time, achieve a higher classification accuracy than the classical algorithms, including CD, PCD, PT5, PT10, and DGS, and improve the mixing rate of Gibbs sampling chain. Furthermore, based on the experimental results, it can be concluded that the proposed FGS algorithm has a certain universality.

Data Availability

Previously reported data were used to support this study. These prior studies (and datasets) are cited at relevant places within the text as references. The four public datasets available are as follows: MNIST: http://yann.lecun.com/exdb/mnist/, MNORB: https://cs.nyu.edu/~ylclab/data/norb-v1.0-small/, CIFAR10: http://groups.csail.mit.edu/vision/TinyImages/, CIFAR100: http://groups.csail.mit.edu/vision/TinyImages/.

Conflicts of Interest

The authors have no conflicts of interest to declare.

Acknowledgments

The authors are grateful to Professor Gao, Dr. Wan, and Dr. Li for discussions on this topic. This study was supported by the (1) National Natural Science Foundation of China (Grant no. 61573285) and (2) Science and Technology on Avionics Integration Laboratory and Aeronautical Science Foundation of China (ASFC) (Grant no. 20175553027).

References


