Adversarial Training Methods for Boltzmann Machines

JIAN ZHANG, SHIFEI DING, NAN ZHANG, AND WEIKUAN JIA

1 School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
2 Mine Digitization Engineering Research Center, Ministry of Education, China, Xuzhou 221116, China
3 School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

Corresponding author: Shifei Ding (dingsf@cumt.edu.cn)

This work was supported by the National Natural Science Foundations of China under Grant 61672522 and Grant 61976216.

ABSTRACT A Restricted Boltzmann Machines (RBM) is a generative Neural Net that is typically trained to minimize KL divergence between data distribution $P_{data}$ and its model distribution $P_{RBM}$. However, minimizing this KL divergence does not sufficiently penalize an RBM that place a high probability in regions where the data distribution has a low density, and therefore, RBMs always generate blurry images. In order to solve this problem, this paper extends the loss function of RBMs from KL divergence to adversarial loss and proposes an Adversarial Restricted Boltzmann Machine (ARBM) and an Adversarial Deep Boltzmann Machine (ADBM). Different from the other RBMs, an ARBM minimizes its adversarial loss between the data distribution and its model distribution without explicit gradients. Different from traditional DBMs, an ADBM minimizes its adversarial loss without a layer-by-layer pre-training. In order to generate high-quality color images, this paper proposes an Adversarial Hybrid Deep Generative Net (AHDGN) based on an ADBM. The experiments verify that the adversarial loss can be minimized in our proposed models, and the generated images are comparable with the current state-of-the-art results.

INDEX TERMS Restricted Boltzmann machine, generative model, adversarial generative net, Gibbs sampling, neural net.

I. INTRODUCTION

The last decade has witnessed revolutionary advances in machine learning, mainly due to the progress in training neural nets. In generative models, neural generative models such as Restricted Boltzmann Machines (RBMs) [1]–[3], Variational Autoencoders (VAEs) [4], Generative Flow models (Glows) [5], and Generative Adversarial Networks (GANs) [6] have demonstrated promising results on image synthesis. GANs, in particular, are generally regarded as the current state-of-the-art [7]. The popularity of RBM-based generative models, including Deep Belief Nets and Deep Boltzmann Machines, has faded in recent years. The charge is that other approaches, especially GANs, simply work better in practice. Actually, RBMs and derived models generally have sufficient representational power to learn essentially any distribution [8], the difficulties must arise during training.

Traditional RBM training algorithms require explicit gradients of their objective functions. In RBMs, although explicit gradients are intractable owing to complex partition functions, they can be approximated by using sampling-based algorithms, such as Gibbs sampling [9], [10]. Based on Gibbs sampling, an RBM is trained to minimize KL divergence between data distribution $P_{data}$ and its model distribution $P_{RBM}$, and this divergence is also called forward KL divergence, conversely, the KL divergence between $P_{RBM}$ and $P_{data}$ is called reverse KL divergence [11]. But, in practical terms, minimizing forward KL divergence does not sufficiently penalize RBMs that place a high probability in regions where the data distribution has a low density, and as a result, a standard RBM always generates blurry images. Moreover, in RBMs, gradients of the reverse KL divergence and the adversarial loss such as JS divergence or Wasserstein distance are hardly expressed explicitly, so RBMs cannot be trained to minimize these objective functions by using traditional training methods. Therefore, a major difficulty in traditional RBM training algorithms is that if gradients of the loss function are not explicit, RBM training is intractable, and this difficulty seriously limits the research and application of RBMs.

The main work of this paper is to alleviate the above difficulty. We propose a novel RBM training algorithm and show that the novel RBM can be trained to minimize an adversarial...
loss by using the proposed algorithm even if gradients of the objective function are not explicit. We call the resulting model an Adversarial Restricted Boltzmann Machine (ARBM). In an ARBM, the energy function and conditional probabilities can be expressed explicitly, and although gradients of the objective function are not explicit, by using special activation probabilities and re-parameterized tricks, they can be calculated in a frame of generative adversarial nets [11], [12]. However, if data lie on a low-dimensional Manifold of a high-dimensional space, most generative adversarial nets face a Gradient Vanishing problem [13]–[15]. Fortunately, we can alleviate this problem by using the hidden layer of an ARBM to learn an intrinsic Manifold embedding of data. Specifically, an ARBM is a Generative Neural Net, and the conditional probabilities in ARBMs follow Truncated Gaussian distribution, which can be effectively sampled by using re-parameterized tricks. Based on re-parameterized tricks, a step of Gibbs sampling can be realized as a Forward-propagation process of a 2-layer Neural Net, and an RBM can be trained end-to-end to minimize the adversarial loss and a graph similarity loss. In order to generate more high-quality images, this paper proposes an Adversarial Deep Boltzmann Machine (ADBM) and an Adversarial Hybrid Deep Generative Net (AHDGN) based on the proposed adversarial training methods. The experiments verify that the adversarial loss can be minimized in our proposed models, and the generated images are comparable with the current state-of-the-art results.

Our main contributions can be summarized as follows:

1) Traditional RBM training algorithms require explicit gradients of the objective function. We relax this limitation and show that an ARBM can be trained without explicit gradients. Without the limitation of explicit gradients, the objective function of RBM can be extended from the forward KL divergence to any adversarial loss.

2) In order to capture the similarity between pixels and alleviate the Gradient Vanishing problem in adversarial training algorithms, a graph regularization term is introduced into the hidden layer, and an additional discriminator is trained to minimize an adversarial loss of implicit expressions in the ARBM. We also theoretically analyze the optimal solution of the proposed ARBM.

3) A traditional Deep Boltzmann Machine (DBM) is trained by a layer-by-layer pre-training and a fine-tuning training process. Different from traditional DBMs, this paper designs an ADBM and a corresponding convolutional hybrid generative net, which can be directly trained without complex pre-training processes.

This paper is structured as follows: We begin with a brief review of RBMs, then discuss the problems in RBM training and go on to define and describe ARBMs and the corresponding deep generative neural nets. Finally, we present the experimental results.

II. RESTRICTED BOLTZMANN MACHINES

An RBM is an energy-based model with two layers of neurons. Visible units \( x \) describe the data and hidden units \( h \) capture interactions between the visible units. The joint distribution

\[
p(x, h) = Z^{-1}e^{-E(x, h)}
\]

is defined by an energy function:

\[
E(x, h) = \sum_i a_i(x_i) + \sum_j b_j(h_j) - \sum_{i,j} W_{ij} \frac{x_i}{\sigma_i^2} h_j \epsilon_j
\]

with a partition function \( Z = \sum_{x,h} e^{-E(x, h)} \), where, \( a, b \) are generic functions, \( \epsilon \) and \( \sigma \) are scale parameters. This formulation provides a flexible way of writing RBMs that encompass common models such as Bernoulli RBMs, Gaussian RBMs, and Truncated Gaussian Graphical Models (TGGMs). In an RBM, the objective is to maximize the log-likelihood \( L = E_{\text{data}} [\log p(x, h)] \), and maximizing the log-likelihood function is equivalent to minimizing the forward KL divergence between data distribution \( P_{\text{data}} \) and the model distribution \( P_{\text{RBM}} \). For the convenience of expression, take a dataset, which has a single sample as an example, the objective function of an RBM can be expressed as follow:

\[
L_s = \log(p(x)) = \log\left(\sum_h p(x, h)\right)
\]

\[
= \log\left(\sum_h e^{-E(x, h)}\right) - \log(Z)
\]

Therefore, gradients of the log-likelihood and forward KL divergence with respect to RBM parameters take an same explicit form:

\[
\frac{\partial L_s}{\partial \theta} = -\sum_h p(h|x) \frac{\partial E(x, h)}{\partial \theta} + \sum_{x,h} p(x, h) \frac{\partial E(x, h)}{\partial \theta}
\]

\[
= -E_{\text{data}} \left[ \frac{\partial E(x, h)}{\partial \theta} \right] + E_{\text{RBM}} \left[ \frac{\partial E(x, h)}{\partial \theta} \right]
\]

However, this explicit gradient is intractable owing to the complex partition function. Fortunately, the key feature of an RBM is the conditional independence of layers, which allows the RBM to sample from its distribution by using block Gibbs sampling. In order to obtain effective samples form \( P_{\text{RBM}} \), some approximate algorithms are proposed based on Gibbs sampling such as Contrastive Divergence and Persistent Contrastive Divergence (PCD) [10], [11]. The two averages in Eq. (2) are computed using examples from the data set and samples drawn from the model by Gibbs sampling, respectively. In RBM training, if \((\tilde{x}, \tilde{h})\) is a pair of sample drawn from Gibbs sampling, then \((\tilde{x})\) is used as an approximated sample from \( P_{\text{RBM}} (x) \). A diagram of Gibbs sampling in RBM training is shown in Fig. 1.

When the hidden units and visible units are binary, and the parameter \( \theta \) contains \( W, a, \) and \( b \). Based on CD algorithm,
The gradients with respect to RBM parameters are expressed as follow:

\[
\frac{\partial \ln P(x)}{\partial W_{ij}} \approx P(h_i = 1|x^{(0)}) - P(h_i = 1|x^{(k)})
\]

\[
\frac{\partial \ln P(x)}{\partial a_i} \approx x_i^{(0)} - x_i^{(k)}
\]

\[
\frac{\partial \ln P(x)}{\partial b_i} \approx P(h_i = 1|x^{(0)}) - P(h_i = 1|x^{(k)})
\]

where, \(x^{(k)}\) is the data, and \(x^{(k)}\) is the state of visible units at \(k^{th}\) alternative Gibbs sampling between \(p(h|x)\) and \(p(x|h)\) by using Contrastive Divergence algorithm (CD-K). The RBM training algorithm is shown in Algorithm 1.

Based on Gibbs sampling, minimizing the forward KL divergence strongly punishes RBMs that underestimate the probability of data, whereas the situation that RBMs overestimate the probability of data will not lead to enough gradients. As a result, a conventional RBM always try to cover all the modes of data and generates blurry images [13].

**Algorithm 1. Minibatch Training of Binary RBMs**

for number of training iterations do:

Sample minibatch of \(N\) data samples \(X_{train}\) from training dataset.

# get \(x^{(k)}\) from \(k\) steps of Gibbs sampling

for \(t\) in \(k\) steps do:

(1) Sample \(h^{(t+1)}\) from \(p(h|x)\),

(2) Sample \(x^{(t+1)}\) from \(p(x|h)\)

end for

Calculate explicit gradients:

\[\frac{\partial L}{\partial W_{ij}} = -E_{p_{data}}[\frac{\partial \ln E(x, h)}{\partial W_{ij}}] + E_{p_{model}}[\frac{\partial \ln E(x, h)}{\partial W_{ij}}]\]

Update the parameters in RBMs:

\[\Delta W_{ij} \approx \eta [p(h_i = 1|x^{(0)})x^{(0)} - p(h_i = 1|x^{(k)})x^{(k)}]\]

\[\Delta a_i \approx \eta [x_i^{(0)} - x_i^{(k)}]\]

\[\Delta b_i \approx \eta [p(h_i = 1|x^{(0)}) - p(h_i = 1|x^{(k)})]\]

where, \(\eta\) are learning rates.

end for

the gradients with respect to RBM parameters are expressed as follow:

where, \(\hat{P}(x)\) is the parameterized MCMC density, and \(\hat{p}_{model}\) is the model distribution. The RBM training algorithm is shown in Algorithm 1.

Based on Gibbs sampling, minimizing the forward KL divergence strongly punishes RBMs that underestimate the probability of data, whereas the situation that RBMs overestimate the probability of data will not lead to enough gradients. As a result, a conventional RBM always try to cover all the modes of data and generates blurry images [13].

**III. THE PROPOSED ARBMS AND THE CORRESPONDING DEEP GENERATIVE MODELS**

An RBM is trained to minimize the forward KL divergence based on Gibbs sampling. Although gradients have explicit expressions in RBMs, they are intractable, and some sampling-based methods can be used to obtain approximated gradients based on the explicit expressions. Because gradients of adversarial loss are hardly to be explicitly expressed, minimizing the adversarial loss such as JS divergence or Wasserstein distance in RBMs is difficult. In this paper, we relax this limitation and show that an ARBM can be trained without explicit gradients. Without this limitation, the objective function of an ARBM can be extended from forward KL divergence to any adversarial loss. Moreover, we design an ADBM and a corresponding convolutional hybrid generative net, which can be directly trained without complex pre-training processes.

**A. ADVERSARIAL RESTRICTED BOLTZMANN MACHINES**

Monte Carlo (MC) methods are key approaches to deal with complex probability distributions in machine learning, and they approximate a complex distribution by using a small number of typical states, obtained by sampling ancestrally from a proposal distribution or iteratively by using a suitable Markov chain (Markov Chain Monte Carlo, or MCMC). If MCMC kernels are parameterized by neural nets, the resulting Markov Chain can be viewed as an implicit generative model, and the JS divergence or a Wasserstein distance between data distribution \(P_{data}\) and the model distribution \(P_{model}\) can be minimized. This parameterized MCMC model is called an Adversarial Markov Chain (MGAN) [11], [12]. However, the transition probabilities in MGANs are implicit, and MGANs also suffer from the Gradient Vanishing problem.

The proposed ARBMs are based on two situations. The first one is that if a MCMC transition kernel can be expressed as an explicit probability, and this explicit probability can be decomposed into alternative Gibbs sampling probabilities of \(p(x|h)\) and \(p(h|x)\) in ARBMs, ARBMs can be trained based on Assumption (1).

**Assumption 1:** If a Markov Chain can reach its detailed balance state after \(K\) steps of Gibbs sampling, and then a sample drawn from \((K+1)\)th step Gibbs sampling is an effective sample of \(P(x, h)\). Just like the conventional training algorithms in RBMs, \(\hat{x}^{(K+1)}\) is used as an approximated sample from \(P_{model}(x)\), because \(P_{model}(x)\) cannot be sampled directly.
Based on Assumption (1), $\hat{x}^{(K+1)}$ can be treated as an approximated sample of the model distribution $P_{\text{ARBMS}}(x)$, and the MCMC transition kernel can be explicitly expressed by an alternative Gibbs sampling process of $p(x|h)$ and $p(h|x)$. In order to obtain effective gradients, the sampling process of ARBMs should be differentiable. Therefore, the second situation ARBMs being based on is that Gibbs sampling of the proposed ARBM can be re-parameterized so that this sampling process can be realized with a 2-layer Neural Net. Specifically, an ARBM is a special form of RBMs, and the conditional probabilities in ARBMs follow Truncated Gaussian Distribution [16], which can be effectively sampled by using re-parameterized tricks. Based on re-parameterized tricks [17], a step of Gibbs sampling can be realized with a 2-layer Neural Net. And the energy function and the probabilities of ARBMs can be expressed as follows:

$$E(x, h) = \frac{1}{2}(x^T \text{diag}(a)x + h^T \text{diag}(d)h)$$

$$p(x, h) = \frac{1}{Z} e^{-E(x,h)}I(x \geq 0)I(y \geq 0)$$

$$p(x|h) = \prod_{i=1}^{m} N_T(x_i) \left[ \frac{1}{d_i} - Wh + b \right] \left[ \frac{1}{d_i} - a_i \right]$$

$$p(h|x) = \prod_{i=1}^{m} N_T(h_i) \left[ \frac{1}{d_j} - Wh^T + c \right] \left[ \frac{1}{d_j} - a_i \right]$$

where, $N_T(.)$ denotes the Truncated Gaussian distribution, and $I(.)$ is the indicator function.

When the covariance matrix of Gaussian distribution is diagonal, the sampling process can be expressed by a re-parameterized trick:

$$x \sim N\left(u, \sigma^2\right) \rightarrow x = u + \sigma \ast \varepsilon$$

where, $\varepsilon \sim N(0, 1)$

Based on the re-parameterized trick, Eq. (5) and Eq. (6) of ARBMs can be expressed in Eq. (7):

$$h_i = \frac{1}{d_j} - \text{Wh} + c_i + \frac{1}{d_j} \ast \varepsilon$$

$$x_i = \frac{1}{a_i} - \text{Wh} + b_i + \frac{1}{a_i} \ast \varepsilon$$

where, $\varepsilon \sim N(0, 1)$

Therefore, based on $\epsilon$, the sampling process in ARBMs is parameter-free, and the hidden representation $h$ can be calculated by a forward-propagation layer that is parameterized by $W$, $c$, and $d$. The reverse sampling of $x$ based on $h$ can be calculated by another forward-propagation layer that is parameterized by $W$, $b$, and $a$. Just like traditional RBMs, the weights $W$ is shared in the 2 forward-propagation layers. Therefore, a full step of Gibbs sampling can be expressed by a 2-layer Neural Net in Fig. 2.

As Fig. 2 shows, based on the re-parameterized trick, a full step of Gibbs sampling can be expressed by a 2-layer Neural Net, and by using this Neural Net as an explicit transition kernel, an ARBM can be trained to minimize an adversarial loss, such as $JS$ divergence or Wasserstein distance. However, gradients of the adversarial loss may not be stable, when data lie on a low-dimensional Manifold, gradients may vanish. In order to alleviate this problem, this paper embeds the data to a low-dimensional representation, which aims to capture the low-dimensional Manifold structure. Fortunately, the hidden layer in an ARBM can be used as a low-dimensional embedding. Based on the theory of graph Laplacian, a similarity matrix of data can be defined according to a Gaussian diffusion kernel:

$$\omega(i, j) = \exp\left(\frac{-d(i, j)}{\sigma^2}\right)$$

where, $d(i, j)$ is the Euclidean distance between $x_i$ and $x_j$. Assuming that $h_i$ and $h_j$ are corresponding low-dimensional hidden representations of $x_i$ and $x_j$ in an ARBM. In order to capture the low-dimensional Manifold structure and produce an effective gradient when the gradient of the adversarial loss $Loss_{MGAN}$ between $P_{\text{data}}(x)$ and $P_{\text{model}}(x)$ in Eq. (9) vanishes, this paper introduces a Graph Similarity Loss to the loss function and minimizes both the adversarial loss of $(P_{\text{data}}(h), P_{\text{model}}(h))$. The loss function of ARBMs can be expressed as Eq. (9) - Eq. (11), and the Graph Similarity Loss is expressed as Eq. (12):

$$L(G, D_1, D_2) = \alpha Loss_{MGAN} + (1 - \alpha) Loss_{Hidden}$$

$$Loss_{MGAN} = E_{x \sim P_{\text{data}}} \left[ \log D_1(x) \right]$$

$$+ \beta Loss_{graph}$$

$$+ \lambda E_{x \sim N(0, 1), \hat{x} \sim T^m(x)} \left[ \log(1 - D_1(\hat{x})) \right]$$

$$+ \left( (1 - \lambda) E_{x \sim P_{\text{data}}, \hat{x} \sim T^m(x)} \right)$$

$$\times \left[ \log(1 - D_1(\hat{x})) \right]$$
Therefore, when the Markov Chain reaches its detailed balance, and the stationary distribution is denoted as \( \pi \), it shows that, given a theoretical analysis of the proposed ARBMs, that essentially enforces that, given a fixed transition kernel, a Markov Chain 2-tuple \((T, h)\) can be regarded as a function of the random variable \( D \). Note that this is a fixed point for the transition operator. We assume that the Markov chain reaches the detailed balance state after the generator would run \((b + m)/2\) steps on average, and \( b \) and \( m \) are hyper-parameters in this paper. Therefore, for every step of gradient update, we assume that the Markov Chain has reached the detailed balance state. \( D_1 \) and \( D_2 \) are discriminators, \( D_1 \) is a Convolutional Neural Net, and \( D_2 \) is a Full-connection Neural Net. Now, we provide a theoretical analysis of the proposed ARBMs, that essentially shows that, given \( G, D_1 \) and \( D_2 \), at the optimal points, \( G \) can recover the data distributions by minimizing both a JS divergence of \((\pi, \pi_h)\) and a JS divergence of \((\pi, \pi_model)\). Firstly, consider the optimization problem with respect to discriminators given a fixed generator.

**Proposition 1:** Given a fixed \( G \), maximizing \( L(G, D_1, D_2) \) about \( D_1 \) and \( D_2 \) in Eq. (9) yields to the following closed-form optimal discriminators \( D_1^*, D_2^* \), which are independent of parameter \( \alpha \):

\[
D_1^* = \frac{\pi \pi(x)}{\pi \pi(x) + \pi_model(x)}
\]

\[
D_2^* = \frac{\pi \pi(h)}{\pi \pi(h) + \pi_model(h)}
\]

**Proof:** According to the theory of Markov Chain and our assumption, given a fixed transition kernel, a Markov Chain can reach the detailed balance state after \((b + m)/2\) steps on average, and the stationary distribution is denoted as \( \pi_model \). Therefore, when the Markov Chain reaches its detailed balance state, Eq. (10) and Eq. (11) can be expressed as:

\[
\begin{align*}
\text{LOSSMGAN} &= E_{x \sim \pi} \left[ \log D_1(x) \right] \\
\text{LOSShidden} &= E_{h \sim \pi} \left[ \log D_2(h) \right]
\end{align*}
\]

Therefore, the loss function can be written as:

\[
L = \alpha \int x \, E_1(D_1(x)) + E_2(D_2(x)) \, dx \\
+ (1 - \alpha) \int h \, E_1(D_2(h)) + E_2(D_2(h)) \, dh \\
+ \beta \text{LOSSgraph}
\]

where, 
\[
\begin{align*}
E_1(D_1(x)) &= \pi \pi(data(x)) \log \pi \pi(data(x)) \\
E_2(D_1(x)) &= \pi \pi_model(x) \log (1 - D_1(x)) \\
E_1(D_2(h)) &= \pi \pi(data(h)) \log D_2(h) \\
E_2(D_2(h)) &= \pi \pi_model(h) \log (1 - D_2(h))
\end{align*}
\]

Considering the function inside the integral, given \( x \), we maximize this Eq. (16) w.r.t the \( D_1 \) to find \( D_1^* \). Meanwhile, \( h \) can be calculated by using Eq. (7), given \( h \), Eq. (15) w.r.t the \( D_2 \) can be maximized to find \( D_2^* \). The outputs of \( D_1 \) and \( D_2 \) are in the Interval \((0, 1)\). Based on Eq. (14), \( D_1 \) and \( D_2 \) can be treated as 2 variables, and the dependent variable \( \text{LOSSMGAN} \) can be regarded as a function of the random variable \( D_1 \). Based on Eq. (15), the dependent variable \( \text{LOSShidden} \) can be regarded as a function of the variable \( D_2 \). Setting the derivatives w.r.t variables \( D_1 \) and \( D_2 \) to 0, we gain:

\[
\alpha \frac{\pi \pi(data(x))}{(1 - D_1(x))^2} + \frac{\pi \pi(data(x))}{(1 - D_1(x))^2} = 0
\]

\[
(1 - \alpha) \frac{\pi \pi_model(h)}{(1 - D_1(h))^2} + \frac{\pi \pi(data(h))}{(D_1(h))^2} = 0
\]

Based on Eq. (17), the 2nd Derivatives are:

\[
\alpha \frac{\pi \pi(data(x))}{(1 - D_1(x))^2} + \frac{\pi \pi(data(x))}{(1 - D_1(x))^2} \\
\times (1 - \alpha) \frac{\pi \pi_model(h)}{(1 - D_1(h))^2} + \frac{\pi \pi(data(h))}{(D_1(h))^2}
\]

which are non-negative. Therefore, the extreme point of Eq. (16) is unique w.r.t \( D_1 \) or \( D_2 \), and the **Proposition 1** is proved.

Next, we fix \( D_1 = D_1^*, D_2 = D_2^* \) and calculate the optimal \( G^* \) for the generator \( G \).

**Proposition 2:** Given \( D_1^*, D_2^* \), at the Nash equilibrium point \((G^*, D_1^*, D_2^*)\), minimizing \( L(G, D_1, D_2) \) in Eq. (9) yields to the following solutions:

\[
L(G^*, D_1^*, D_2^*) = -\log(D_1^*)
\]

\[
D_1^* = \frac{1}{2}, \quad \text{and} \quad D_2^* = \frac{1}{2}
\]

\[
\forall x, h \at \pi_mod = \pi_data
\]

**Proof:** Given \( D_1 = D_1^*, D_2 = D_2^* \), the objective function can be written as:

\[
L(G, D_1^*, D_2^*)
\]

\[
= \beta \text{LOSSgraph} + \alpha \int x \pi \pi(data(x)) \log \pi \pi(data(x)) + \pi \pi_model(x) \, dx \\
+ \alpha \int x \pi \pi_model(x) \log \pi \pi(data(x)) + \pi \pi_model(x) \, dx \\
\times (1 - \alpha) \int h \pi \pi(data(h)) \log \pi \pi(data(h)) + \pi \pi_model(h) \, dh \\
+ (1 - \alpha) \int h \pi \pi_model(h) \log \pi \pi(data(h)) + \pi \pi_model(h) \, dh
\]

\[
= \alpha (-\log 4 + 2 \text{JSD} (\pi \pi(data(x) || \pi \pi_model(x))) \\
+ (1 - \alpha) \text{JSD} (\pi \pi(data(h) || \pi \pi_model(h))) \\
- (1 - \alpha) \log 4 + \beta \text{LOSSgraph}
\]
Algorithm 2 Minibatch Stochastic Gradient Descent Training of ARBMs. The Number of Steps to Apply to the ARBMs, \( m \) and \( k \), Are Hyperparameters

**for** number of training iterations do:
(1) Sample minibatch of \( N \) noise samples \( z \) from noise prior \( N(0, 1) \).
(2) Sample minibatch of \( N \) data samples \( X_{\text{train}} \) from training dataset.
# generate samples from ARBMs.
for \( t \) in \( m \) steps do:
# get \( \tilde{x}^{(m)} \) from \( m \) steps of Gibbs sampling based
# on the noise samples \( z \) in form of Neural Nets:
(1) \( \tilde{h}^{(t+1)}_j = \frac{1}{\sigma_j} [W^T \tilde{x}^{(t)} + c]_j + \frac{1}{\sigma_j} * \epsilon \),
(2) \( \tilde{x}^{(t+1)}_i = \frac{1}{\sigma_i} [\tilde{h}^{(t+1)} + b]_i + \frac{1}{\sigma_i} * \epsilon \),
where, \( x^{(0)} = z \), \( \epsilon \sim N(0, 1) \)
end for for \( t \) in \( k \) steps do:
# get \( \tilde{x}^{(k)} \) from \( k \) steps of Gibbs sampling based
# on the data samples \( x \) in form of a Neural Net:
(1) \( \tilde{h}^{(t+1)}_j = \frac{1}{\sigma_j} [W^T \tilde{x}^{(t)} + c]_j + \frac{1}{\sigma_j} * \epsilon \),
(2) \( \tilde{x}^{(t+1)}_i = \frac{1}{\sigma_i} [\tilde{h}^{(t+1)} + b]_i + \frac{1}{\sigma_i} * \epsilon \),
where, \( x^{(0)} = x_{\text{train}} \), \( \epsilon \sim N(0, 1) \)
end for

Update the discriminator by assigning its stochastic gradient:
\[ \nabla_{\theta_{\text{ARBMs}}} \left[ \frac{1}{N} \sum_{i=1}^{N} \text{Loss}_{\text{MGAN}}(x_i) + \text{Loss}_{\text{hidden}}(h_i) \right] \]
Update the parameters in ARBMs by assigning its stochastic gradient:
\[ \nabla_{\theta_{\text{ARBMs}}} \left[ \frac{1}{N} \sum_{i=1}^{N} \text{Loss}_{\text{MGAN}}(x_i) \right] + \text{Loss}_{\text{hidden}}(h_i) + \text{Loss}_{\text{graph}} \]
end for

Therefore, the objective function in Eq. (20) can be treated as a minimizing process of a JS divergence of \( P_{\text{data}}(x) \) and \( P_{\text{model}}(x) \) and a JS divergence of \( (P_{\text{data}}(h), P_{\text{model}}(h)) \) with an additional graph Laplacian Norm \( \text{Loss}_{\text{graph}} \). When \( P_{\text{data}} \) and \( P_{\text{model}} \) are close enough, \( \text{Loss}_{\text{graph}} \) trends to 0 ideally. Because the JS divergence between two distributions is always non-negative and zero only if they are equal, the global minimum of \( L(G, D_1^*, D_2^*) \) in Eq. (20) is \(-\log 4\).

In the frame of MGANs, if the data lie on a low-dimensional Manifold, gradients of JS divergence between \( P_{\text{data}}(x) \) and \( P_{\text{model}}(x) \) may vanish, which results in unstable gradients. In ARBMs, although gradients of JS divergence of \( (P_{\text{data}}(x), P_{\text{model}}(x)) \) may vanish, the JS divergence of \( (P_{\text{data}}(h), P_{\text{model}}(h)) \) can be used to produce gradients until the embedding representations of \( P_{\text{data}} \) and \( P_{\text{model}} \) are indistinguishable. The ARBM algorithm is shown in Algorithm 2.

The main difference between an RBM and an ARBM is that an ARBM implicitly calculates gradients by using the 2-layer Neural Net. Meanwhile, Based on CD algorithm, an RBM has an explicit gradient and the gradient update can be denoted explicitly in Algorithm 1.

B. ADVERSARIAL DEEP BOLTZMANN MACHINES AND ADVERSARIAL HYBRID DEEP GENERATIVE NETS

DBM training is difficult because a DBM usually consists of several layers, and the DBM training contains two stages, in the first stage, every two adjacent layers in a DBM are treated as an RBM, and the RBMs should be trained layer-by-layer; The second stage is a global training based on MCMC [18], [19]. In order to simplify the training process of DBMs, this paper proposes an adversarial training method for a 2-hidden-layer Deep Boltzmann Machine. We call the resulting model an Adversarial Deep Boltzmann Machine (ADBM).

As a generative model, an ADBM can be directly trained without a pre-training process. In order to generate color images and decrease the training complexity, we introduce convolutional layers to the generative model and construct an Adversarial Hybrid Deep Generative Net (AHDGN).

1) ADVERSARIAL DEEP BOLTZMANN MACHINES

An ADBM consists of two hidden layers and the energy function can be expressed as:
\[ E(x, h_1, h_2) = \frac{1}{2} (x^T \text{diag}(\alpha) x + h_1^T \text{diag}(\beta_1) h_1 + h_2^T \text{diag}(\beta_2) h_2) \]
\[ -x^T W_1 h_1 - h_1^T W_2 h_2 - a^T x - b^T h_1 - c^T h_2 \] \( (21) \)

Based on the re-parameterized tricks, Eq. (22) can be can be expressed as Eq. (23):
\[ x_i = \frac{1}{\alpha_i} [W_1 h_1 + a]_i + \frac{1}{\alpha_i} * \epsilon \]
\[ h_{1,j} = \frac{1}{\beta_{1,j}} [W_1^T x + W_2 h_2]_j + \frac{1}{\beta_{1,j}} * \epsilon \]
\[ h_{2,j} = \frac{1}{\beta_{2,j}} [W_1^T h_1 + c]_j + \frac{1}{\beta_{2,j}} * \epsilon \]
\( \text{where, } \epsilon \sim N(0, 1) \) \( (23) \)

The Gibbs sampling process can be expressed as follows:
\[ P(h_1 | x, h_2) \rightarrow P(h_2 | h_1) \rightarrow P(x | h_1) \] \( (24) \)

Based on Eq. (23) and Eq. (24). A step of transition kernel in an ADBM can be expressed by a Neural Net in Fig. 3.

In ADBMs, the \( \text{Loss}_{\text{graph}} \) term is only added to the first hidden layer, because the dimension of the second hidden layer is similar to that of the first hidden layer. The training algorithm of an ADBM is shown in Algorithm 3.
2) ADVERSARIAL HYBRID DEEP GENERATIVE NETS

ADBMs are effective in generating real-valued gray images, but the color images generated by ADBMs are unsatisfactory, because a deeper ADBM is difficult to be expressed as a Neural Net, and ADBMs are hard to be extended to Convolutional deep Neural Nets [20]–[22]. Therefore, Adversarial Hybrid Deep Generative Nets (AHDGNs) are proposed based on ADBMs. In AHDGNs, images are passed into a convolutional layer, and the output of this convolutional layer are used as the input of an ADBM. The structure of an AHDGN is shown in Fig. 4.

Hybrid generative models such as Discrete Variational Autoencoders are usually difficult for training [23]. In this paper, owing to the proposed adversarial training algorithm, the hybrid model can be trained in a unified frame of a Neural Net by using BP algorithm. The training algorithm of AHDGNs is shown in Algorithm 4.

IV. EXPERIMENTS

In the previous section, we analyze and demonstrate the optimal solution of the proposed ARBM theoretically. In this section, we show empirical results of the proposed models by using some datasets that are commonly used for generative models. Because the JS divergence can be treated as a function of forward KL divergence and reverse KL divergence, we can indirectly reflect JS divergence by monitoring both the forward KL divergence and reverse KL divergence. We aim to demonstrate 3 key results:

1) During ARBM training, both forward KL divergence and reverse KL divergence are decreasing with iterations. Quantitatively, compared with RBMs and other commonly used generative models, ARBMs and ADBMs achieve ideal minimums for both forward KL divergence and reverse KL divergence.

2) By using Gibbs Sampling, ARBMs can learn the Manifold structure of input data and generate sharp images.
Algorithm 4 Minibatch Stochastic Gradient Descent Training of AHDGNs

for number of training iterations do:

(1) Sample minibatch of $N$ noise samples $z$ from noise prior $N(0, 1)$.
(2) Sample minibatch of $N$ data samples $X_{\text{train}}$ from training dataset.

# generate samples from AHDGNs.

for $t$ in $m$ steps do:

(1) get $h_i^{(t+1)}$ from $z$ though the convolutional layer.
(2) Use $h_i^{(t+1)}$ as the input of ADBM, and get the generated samples $\tilde{h}_i^{(t+1)}$ from ADBM with $p$ step Gibbs sampling.

(3) Get $\hat{x}^{(t+1)}$ from $\tilde{h}_i^{(t+1)}$ by deconvolutional layer.

end for

for $t$ in $k$ steps do:

(1) get $h_i^{(t+1)}$ from $X_{\text{train}}$ though the convolutional layer.
(2) Use $h_i^{(t+1)}$ as the input of ADBM, and get the generated samples $\tilde{h}_i^{(t+1)}$ from ADBM with $p$ step Gibbs sampling.

(3) Get $\hat{x}^{(t+1)}$ from $\tilde{h}_i^{(t+1)}$ by deconvolutional layer.

end for

Update the discriminator by gradients:

$\nabla \theta_\text{D} = \frac{1}{N} \sum_{i=1}^{N} \text{Loss}_{\text{D}}(x_i) + \text{Loss}_{\text{D}}(\hat{x}_i)$

Update the parameters in AHDGNs:

$\nabla \theta_{\text{AHDGN}} = \frac{1}{N} \sum_{i=1}^{N} \text{Loss}_{\text{AHDGN}}(x_i) + \text{Loss}_{\text{AHDGN}}(\hat{x}_i)$

where, $\text{Loss}_{\text{AHDGN}}$ is built by the first hidden layer
 # in the ADBM, and $h_i^{(1)}$ denotes the first hidden layer.

end for

The gradient-based updates can use any gradient-based learning rule. We use momentum in our experiments.

Moreover, after the Markov Chain of an ARBM reaches its detailed balance state, along with every step of state transition, an ARBM can realize a state transition between multi-modes of the data distribution starting from a random noise.

3) ADBMs and AHDGNs can realize state transition between multi-modes of the data distribution starting from a random noise, and the generated images are comparable to the current state-of-art results.

The MNIST dataset of handwritten images is one of the most widely used benchmarks in machine learning. Following [7], this paper monitors the forward $KL$ divergence and reverse $KL$ divergence. Let $\{x_i\}_{i=1}^{n}$ and $\{y_i\}_{i=1}^{m}$ be samples drawn from distribution $p$ and $q$. Let $p_{\text{h}}(i)$ be the distance from $X_i$ to its neighbor in $\{X_j\}_{j \neq i}$, and $v_{\text{m}}(i)$ be the distance from $X_i$ to its neighbor in $Y_i$. Then,

$$KL[p||q] \approx \frac{d}{n} \sum_{i=1}^{n} \frac{v_{\text{m}}(i)}{p_{\text{h}}(i)} + \log \frac{m}{n-1}$$

where, $d$ is the dimension of the space. The reverse $KL$ divergence can be approximated by reversing the identities of $X$ and $Y$ [24]. In experiments, we compute the forward $KL$ divergence and reverse $KL$ divergence on minibatches of the MNIST, and show that training an ARBM decreases both the forward $KL$ divergence and reverse $KL$ divergence in Fig. 5.

As we can see in Fig. 5, both forward $KL$ divergence and reverse $KL$ divergence decrease during ARBM training. In order to make a qualitative comparison, six different models are trained to estimate the two $KL$ divergences on MNIST: A non-convolutional GAN, a non-convolutional WGAN-GP [25], a Restricted Truncated Gaussian Graph Model (RTGMM) [16], a Variational RBM [26], an ARBM, and an ADBM. The Variational RBM introduces neutral variational inference to the partition function and tracks the partition function during learning. The RTGMM introduces Truncated Gaussian distribution to an undirected Gaussian Graph model, which can be treated as a special Restricted Boltzmann Machine. The WGAN-GP aims to minimize the Wasserstein distance between the data distribution and model distribution with a gradient penalty (WGAN-GP). The estimations of forward $KL$ divergence and reverse $KL$ divergence on MNIST are shown in Table 1.

Table 1 shows that adversarial training algorithms are effective in minimizing both the forward $KL$ divergence and the reverse $KL$ divergence, and the ADBM achieves comparable results with RBM-based models and GANs.

The next experiment shows that by minimizing the two $KL$ divergences and learning the Manifold structure, an ARBM
can generate sharper images than standard RBMs. In order to show that ARBMs can generate sharp images and learn the intrinsic Manifold structure of the data, firstly we run comparative experiments on three artificial datasets, and all the RBM-based models have the exact same architecture. The experimental results are shown in Fig. 6.

Fig. 6 shows a comparison of fantasy particles from each of the generative models along with the corresponding data distributions. The first column samples are drawn from original data, the second column samples are drawn from a standard RBM, the third column samples are generated by a RTGMM, the fourth column samples are generated by a WGAN-GP, and the last column samples are generated by an ARBM. As we can see in Fig. 6, a standard RBM spreads the model density across the support of the data distribution. Meanwhile, an ARBM is able to learn the three distributions and to recognize the training data distribution that has low probability. We also offer qualitative comparisons of ARBMs with RBM-based models and the state-of-art models in image generation.

Firstly, we train a RTGMM and a Variational RBM on MNIST by using CD (or PCD) algorithms and assume that the models fit the data distribution after enough iterations. For the Variational RBM and the RTGMM, we start with a random sample and run 50 steps Gibbs sampling, and the generated images are shown in Fig. 7:

As Fig. 7 shows, every row of images is drawn from a Markov chain, which starts with a random sample from the training dataset. Although a Variational RBM and a RTGMM can generate images from their Markov Chains, the generated images fall into similar modes of the data distribution.

This experimental result denotes that the CD algorithm and the PCD algorithm limit a Markov Chain to cover the whole modes of the data distribution. Different from Variational RBMs and RTGMMs, a Markov Chain in an ARBM is assumed to be convergent. Therefore, starting from a random noise, an ARBM can realize a state transition between multi-modes of the data distribution. Fig. 8 shows the generated images from an ARBM.

Because a Markov Chain in an ARBM covers most modes in data distribution, ARBMs are more suitable for the image generation task than other RBM-based models. In order to generate sharper images and build a deep generative net, this paper proposes an ADBM. Fig. 9 shows the generated images from an ADBM on MNIST.

An ADBM can be trained without a complex pre-training process, and as we can see in Fig. 9, the generated images on MNIST by an ADBM are sharper and clearer. Next, we compare the image generation ability of ADBMs with other generative Neural Nets on gray images. NORB is a synthetic 3D object recognition dataset that contains five classes of toys (humans, animals, cars, planes, trucks) imaged by a stereo-pair camera system from different viewpoints under different lighting conditions. The stereo-pair images are subsampled from their original resolution of $108 \times 108 \times 2$ to $32 \times 32 \times 2$ to speed up experiments. Fig. 10 shows the generated images on NORB dataset.

As shown in Fig. 10, the first image that contains 10 rows and 50 columns is generated by an ADBM. In every row, the ARBM start with a random noise, and every image in this row is drawn from one step of state transition in the ADBM after the Markov Chain reaches its detailed balance state.

In the second row of Fig. 10, the first image is reconstructed by a Gaussian-binary RBM, the second image is reconstructed by a RTGMM, the third image is generated by a GAN, and the fourth image is generated by a WGAN-GP. Starting with a Gaussian Noise, an ADBM can generate sharp gray images without convolutional operations.
This experiment shows that ADBMs demonstrate a similar ability to GAN-based models on gray images for the image generation task.

However, ADBMs and non-convolutional models are not fit for generating color images, because without characters of local connection and shared weights, there are too many parameters in non-convolutional Neural Nets for modeling color images. Whereas convolution operations in RBM-based models are not reversible, so convolutional layers are difficult to be added to the undirected Probabilistic Graphical Models. In order to generate color images by using ARBM-based models, this paper introduces convolutional layers to an ADBM and proposes an AHDGN, which can be trained by our proposed algorithm. We test the image generation capability of an AHDGN on CelebA dataset. CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. Fig. 11 shows the generated images from an AHDGN on CelebA dataset.

As Fig. 11 shows, the image in the first row is generated by an AHDGN, the first image in the second row is generated by a DCGAN, the second image in the second row is generated by a WGAN, and the last image in the second row is generated by a WGAN-GP. All the models generate images starting with Gaussian noise, and generated images of the proposed AHDGN are comparable to the most commonly used GANs. Lastly, we introduce the details of our experiments. All the models are based on Tensorflow, and the Markov Chain is realized in form of RNNs [15]. The discriminator $D_2$ is a simple 2-layer Full-connection Neural Net, and the other structures and hyperparameters in our experiments are shown in Table 2.
V. CONCLUSION

This paper proposes an Adversarial RBM and two corresponding deep generative nets. ARBMs are trained to minimize two adversarial loss and learning the Manifold structure in data distribution. Experiments show that the proposed ARBM-based generative models perform better than standard RBMs and achieve comparable results with the most commonly used GAN models. This paper extends the objective functions from forward KL divergence to adversarial loss. However, there are still some problems to be solved in our future research. The partition function in an RBM (or ARBM) is still intractable, and some more effective algorithms are needed to approximate the partition function. Although minimizing JS divergence or Wasserstein distance in generative models can generate sharp images, they suffer from the problem of mode-collapse. Generating sharp images from random Gaussian noise to cover the whole data distribution is still an open problem.

REFERENCES

[1] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” Neural Comput., vol. 14, no. 8, pp. 1771–1800, Aug. 2002.
[2] G. E. Hinton, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[3] S. Santurkar, D. Tsipras, and A. Ilyas, “How does batch normalization help optimization?” presented at the 31st Int. Conf. Neural Inf. Process. Syst., 2018.
[4] L. Mescheder, S. Nowozin, and A. Geiger, “Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks,” presented at the 32nd Int. Conf. Mach. Learn., 2017.
[5] A. Grover, M. Dhar, and S. Ermon, “Flow-GAN: Combining maximum likelihood and adversarial learning in generative models,” presented at the 32nd AAAI Conf. Artif. Intell., 2018.
[6] I. Goodfellow, J. Pouget-Abadie, and M. Mirza, “Generative adversarial networks,” presented at the 27th Neural Inf. Process. Syst., 2014.
[7] C. Fisher, A. Smith, and J. Walsh, “Boltzmann encoded adversarial machines,” 2018, arXiv:1804.08682. [Online]. Available: https://arxiv.org/abs/1804.08682
[8] Y. Chen, Z. Lin, and X. Zhao, “Deep learning-based classification of hyperspectral data,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 7, no. 6, pp. 2094–2107, Jun. 2014.
[9] T. Tielemans, “Training restricted Boltzmann machines using approximations to the likelihood gradient,” presented at the 23rd Int. Conf. Mach. Learn., 2008.
[10] T. Tielemans and G. Hinton, “Using fast weights to improve persistent contrastive divergence,” presented at the 24th Int. Conf. Mach. Learn., 2009.
[11] J. Song, S. Zhao, and S. Ermon, “A-NICE-MC: Adversarial training for MCMC,” presented at the 30th Int. Conf. Neural Inf. Process. Syst., 2017.
[12] J. Song, S. Zhao, and S. Ermon, “Generative adversarial training for Markov chains,” presented at the 5th Int. Conf. Learn. Represent., 2017.
[13] K. Li and J. Malik, “Implicit maximum likelihood estimation,” 2018, arXiv:1809.09087. [Online]. Available: https://arxiv.org/abs/1809.09087
[14] M. Arjovsky, S. Chintala, and I. Bottou, “Wasserstein GAN,” presented at the 32nd Int. Conf. Mach. Learn., 2017.
[15] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, “Least squares generative adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2794–2802.
[16] Q. Su, X. Liao, and C. Li, “Unsupervised learning with truncated Gaussian graphical models,” presented at the 31st AAAI Conf. Artif. Intell., 2017.
[17] D. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2014, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114
[18] R. Salakhutdinov and G. Hinton, “An efficient learning procedure for deep Boltzmann machines,” Neural Comput., vol. 24, no. 8, pp. 1967–2006, Aug. 2012.
[19] N. Zhang, S. Ding, J. Zhang, and Y. Xue, “Research on point-wise gated deep networks,” Appl. Soft Comput., vol. 52, pp. 1210–1221, Mar. 2017.
[20] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltan, A. R. Mohamed, G. Dahl, and B. Ramabhadran, “Deep convolutional neural networks for large-scale speech tasks,” Neural Netw., vol. 64, pp. 39–48, Apr. 2015.
[21] J. Masci, U. Meier, and C. Dan, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in Proc. Artif. Neural Netw. Mach. Learn., 2011.
[22] H. Lee, R. Grosse, and R. Ranganath, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” presented at the 24th Int. Conf. Mach. Learn., 2009.
[23] A. Vahdat, W. Macready, and Z. Bian, “DVAE++: Discrete variational autoencoders with overlapping transformations,” presented at the 33rd Int. Conf. Mach. Learn., 2018.
[24] Q. Wang, S. Kulkarni, and S. Verdu, “Divergence estimation for multidimensional densities via k-nearest-neighbor distances,” IEEE Trans. Inf. Theory, vol. 55, no. 5, pp. 2392–2405, 2009.
[25] I. Gulrajani, F. Ahmed, and M. Arjovsky, “Improved training of wasserstein GANs,” presented at the 30th Int. Conf. Neural Inf. Process. Syst., 2017.
[26] V. Kuleshov and S. Ermon, “Neural variational inference and learning in undirected graphical models,” presented at the 30th Int. Conf. Neural Inf. Process. Syst., 2017.

JIAN ZHANG received the B.S. degree in computer science from the Chinese University of Mining and Technology, Xuzhou, China, in 2013, where he is currently pursuing the Ph.D. degree in artificial intelligence and pattern recognition. He is a member of the Mine Digitization Engineering Research Center. His research results have expounded in over ten publications at peer-reviewed journals. His current research interests include deep learning, structure design and optimization of neural networks, and image synthesis.

SHIFEI DING was a Postdoctoral Fellow of the Chinese Academy of Sciences, Beijing, China. He is currently a Professor with the School of Computer Science and Technology, China University of Mining and Technology. He is also a member of the Mine Digitization Engineering Research Center, the Senior Member of the American Computer Association (ACM), the Director of the Chinese Computer Society (CCF), and the Director of the Chinese Artificial Intelligence Society (CAAI).

His research results have expounded in over 200 publications at peer-reviewed journals. He is the Editor-in-Chief of International Journal of Collaborative Intelligence and an Associate Editor of the Journal of Digital Contents Technology and Application. He is also a Reviewer of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE ACCESS, Information Sciences, Neurocomputing, Knowledge-Based Systems, and Pattern Recognition.

NAN ZHANG received the Ph.D. degree in artificial intelligence and pattern recognition from the Chinese University of Mining and Technology, in 2019. He is currently a Postdoctoral Fellow of the School of Computer Science and Technology and the Head of the Pattern Recognition and Machine Learning Research Group, East China Normal University. He is also a member of the Mine Digitization Engineering Research Center. His research results have expounded in over ten publications at peer-reviewed journals. His current research interests include multiview learning and deep learning.

WEIKUAN JIA received the Ph.D. degree in agricultural electrification and automation from Jiangsu University, in 2016. He is currently an Associate Professor with the School of Information Science and Engineering, Shandong Normal University. His research results have expounded in over 30 publications at peer-reviewed journals. His current research interests include artificial intelligence and agricultural information technology.