A Novel Inference of a Restricted Boltzmann Machine

Masayuki Tanaka
Tokyo Institute of Technology
Email: mtanaka@ctrl.titech.ac.jp

Masatoshi Okutomi
Tokyo Institute of Technology
Email: mxo@ctrl.titech.ac.jp

***Abstract*—A deep neural network (DNN) pre-trained via stacking restricted Boltzmann machines (RBMs) demonstrates high performance. The binary RBM is usually used to construct the DNN. However, a continuous probability of each node is used as real value state, although the state of the binary RBM's node should be represented by a random binary variable. One of main reasons of this abuse is that it works. One of others is to reduce a computational cost. In this paper, we propose a novel inference of the RBM, considering that the input of the RBM is the random binary variable. Straight forward derivation of the proposed inference is intractable. Then, we also propose the closed-form approximation of it. We convince that the proposed inference is more reasonable than a conventional algorithm of the RBM. Experimental comparisons demonstrate that the proposed inference improves the performance of the DNN.**

I. Introduction

A deep neural network (DNN) is a feed-forward, artificial neural network, which usually has multi-layer. Recently, the DNN with a new training algorithms has been known as a powerful modeling tool in various fields [1]. Techniques to improve generalization ability have been also proposed. Lee et al has introduced a sparseness constrain training [2]. They have reported sparse activation improves the generalization ability. A dropout training is a technique to improve the generalization ability by dropping several hidden nodes during the training process [3]. Similarly, a drop-connection drops several connections among nodes [3]. These techniques are effective to improve generalization ability. However, a huge computational cost is required. Then, the fast algorithm of the dropout training has been reported [4].

One of keys components of the DNN training is a restricted Boltzmann machine (RBM). The DNN is initialized via stacking RBMs. The training scheme of the DNN can be summarized as follows [1], [8], [5]. First, each layer is independently trained as a restricted Boltzmann machine (RBM). The RBM is trained with an unsupervised way by a contrastive divergence (CD) algorithm [9], [10]. The states of hidden nodes inferred by the trained RBM are used as inputs of the next layer. A sequence of RBMs are trained with the same manner. A deep belief network (DBN) is formed by stacking the trained RBMs. This DBN is a directed generative model [1]. Then, the DNN is initialized with the parameters of the trained DBN. The DNN is constructed by adding a final layer. The constructed network is sometime called the DBN. However, the constructed network is not the DBN. At lease the constructed network is not used as the DBN. In this paper, we call the constructed network the DBN-DNN as same as in [1]. The constructed DBN-DNN is fine-tuned with a supervised manner. For this fine tuning, a classical back-propagation algorithm is usually applied [11]. The unsupervised independent training of the RBMs are called pre-training and the supervised training of the DBN-DNN by the back-propagation algorithm is called fine-tuning. Although the initialization of the DBN-DNN by the parameters of the DBN was heuristically proposed, a theoretical justification is discussed in [12].

The training algorithms mainly focus on the RBM with a Bernoulli distribution, or a binary distribution, although there are several researches on the RBM assuming a Gaussian distribution [5], [6], [7]. In this paper, we also focus on the Bernoulli RBM.

In the pre-training phase, the inferred states of the hidden units of the RBM trained on the data are usually used as data for training a next layer of the RBM. The inferred states of the hidden units are represented by conditional probabilities of hidden unit states given visible unit states. Then, people simply use these conditional probabilities as input data of the next layer of the RBM for the unsupervised training. However, this usage is not theoretically rigorous, because the RBM is a model for the random binary variables. Recall we only focus on the Bernoulli RBM in this paper. The conditional probabilities are used as the input of the RBM, while the conditional probabilities are not random binary variables but real values whose range is zero to one. It is a contradiction in the conventional algorithm.

It is one of open issues to use the probability or the sampled binary sates. Actually, in [13], Hinton recommends to use the probabilities for some purposes and to use the sampled binary sates for other purposes. In this paper, we propose a novel inference of the RBM. The probabilities of the Bernoulli distribution can be considered as expectation of the random variables. It means that the calculation with the probabilities takes the expectation before applying the activation function. The key of the proposed inference is to take the expectation after applying the activation function. However,

a straight forward implementation of the proposed inference is intractable. We also propose the closed form approximation of the proposed inference. In order to approximate the expectation after applying the activation function, an input distribution of the activation function is approximated by Gaussian distribution. Then, the expectation can be calculated by the closed form approximation as same as [4]. The proposed inference can be used for the pre-training by the CD algorithm and for the forward inference of the DBN-DNN. The back-propagation of the DBN-DNN with the proposed inference can be derived by same manner as the DBN-DNN with the classical inference which passes the probabilities to the next layer.

We compare the performance with evaluation datasets. Experimental comparisons demonstrate that the proposed algorithm outperforms the conventional algorithm.

II. Deep Neural Network (DNN)

A. Notations

First, we introduce several notations for discussion. A lowercase p with the vector argument represents a joint probability as

$$p(\boldsymbol{v} = \mathbf{1}) \quad = \quad \prod_i p(v_i = 1)\,. \tag{1}$$

A bold-uppercase P with the vector argument represents the probabilities of each element as

$$\boldsymbol{P}(\boldsymbol{v} = \mathbf{1}) \quad = \quad \begin{pmatrix} p(v_1 = 1) \\ p(v_2 = 1) \\ \vdots \end{pmatrix}. \tag{2}$$

In this paper, we use the sigmoid function, σ, for an activation function defined by

$$\sigma(x) \quad = \quad \frac{1}{1 + e^{-x}}\,. \tag{3}$$

A bold-type $\boldsymbol{\sigma}$ represents element-wise sigmoid function as

$$\boldsymbol{\sigma}(\boldsymbol{x}) \quad = \quad \begin{pmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \vdots \end{pmatrix}. \tag{4}$$

B. The DBN-DNN construction via the stacked RBM

Recently, the DNN have became a hot topic again. However, the DNN is not new. The supervised training of the DNN by an error back propagation is well studied [11]. The error back propagation is steepest descent algorithm which requires a initial guess. The parameter learning of the DNN is strongly non-linear problem and there are many local optimums which are far from the global optimal. The initial setting of the DNN was one of challenging problems until a pre-training via a stacked RBM was proposed [5].

As mentioned above, in this paper the DNN constructed via the stacked RBM is called the DBN-DNN. The construction process of the DBN-DNN is well described in [1]. Here, we quickly review the construction process. Figure 1 shows the outline of the DBN-DNN construction. First, the RBM is pre-trained with training data by the CD training algorithm. Then, the states of the binary hidden units of the trained RBM are used as the training data of the next layer of the RBM. This process is repeated to many layers. A deep belief network (DBN) is formed by stacking the trained RBMs. Adding a decision layer to the stacked RBM, the DBN-DNN is constructed. Once the DBN-DNN is initialized, we can apply classical error back propagation algorithm, which is the supervised learning. This supervised learning process is also called the fine-tuning. In the following sections, we describe each process.

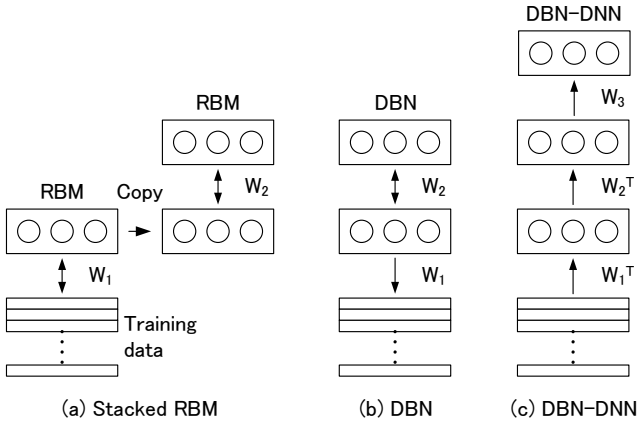


Fig. 1. The overview of the DBN-DNN construction.

C. Definition of Restricted Boltzmann Machine

Before discussion of the pre-training, we describe the definition of the RBM. The joint probabilities of visible units $\boldsymbol{v}$ and hidden units $\boldsymbol{h}$ of the RBM is defined by

$$p(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}) \quad = \quad \frac{1}{Z} e^{E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})}\,, \tag{5}$$

where Z is the partition function, E is the energy function, and $\boldsymbol{W}$, $\boldsymbol{a}$, and $\boldsymbol{b}$ are model parameters. The partition function Z is defined by

$$Z \quad = \quad \sum_{\boldsymbol{v}', \boldsymbol{h}'} e^{E(\boldsymbol{v}', \boldsymbol{h}'; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})}\,. \tag{6}$$

The energy function E is

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}) \quad = \quad -\boldsymbol{h}^T \boldsymbol{W} \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h} - \boldsymbol{c}^T \boldsymbol{v}\,, \tag{7}$$

where T represents the transpose operator.

The conditional probabilities of the hidden state given the visible state $\boldsymbol{v}$ are expressed by

$$\boldsymbol{P}(\boldsymbol{h} = \mathbf{1} | \boldsymbol{v}) \quad = \quad \boldsymbol{\sigma}(\boldsymbol{W}^T \boldsymbol{v} + \boldsymbol{b})\,. \tag{8}$$

Note that the given visible state is random binary state. The conditional probabilities of the visible state given the hidden

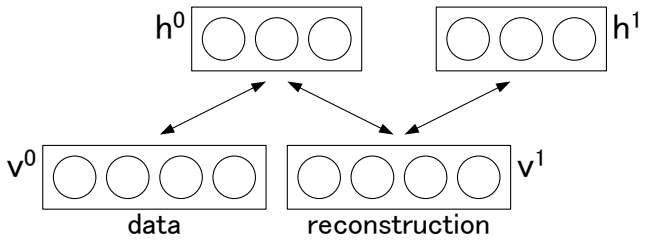


Fig. 2. Data and reconstruction in the contrastive divergence training.

state are similarly expressed by

$$\boldsymbol{P}(\boldsymbol{v}=\boldsymbol{1}|\boldsymbol{h}) \quad = \quad \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{h}+\boldsymbol{c})\,. \tag{9}$$

Again, the given hidden state is the random binary variable.

D. Contrastive Divergence Training

The parameters of the RBM, $\{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}\}$, can be trained for a given training data to maximize the log-likelihood. Here, we only describe the training of parameter $\boldsymbol{W}$. The other parameters, $\boldsymbol{b}, \boldsymbol{c}$, can be trained with the same manner. Hinton proposed a fast learning procedure called contrastive divergence (CD) training [9]. The CD training is performed with the stochastic steepest ascent. The change of the parameters in the CD training is given by

$$\Delta w_{ij} \quad = \quad \epsilon(< v_i^0 h_j^0 > - < v_i^1 h_j^1 >)\,, \tag{10}$$

where ϵ is the training rate, the angle brackets represents the average of the training data, and the superscripts of 0 and 1 are defined as shown in Fig. 2. Starting from the training data $\boldsymbol{v}^0$, the hidden state $\boldsymbol{h}^0$ is inferred by Eq. (8). Then, the training data is reconstructed by Eq. (9) with the hidden state $\boldsymbol{h}^0$. The hidden state $\boldsymbol{h}^1$ is inferred with the reconstruction $\boldsymbol{v}^1$.

Eq. (10) is one of the approximated derivatives of the log-likelihood. It is heuristically determined whether we use the visible and hidden state as binary variables sampled with the probabilities or the probabilities themselves [13]. However, the RBM is defined to model the random binary variables as described above. In this sense, we should use the binary visible and hidden states.

E. Inference of the Conventional Algorithm

Once we construct the DBN, we can easily initialize the DBN-DNN as shown in Fig. 1, because the mathematical expression of the forward inference in the DBN-DNN is same as that in the DBN. Even though theoretical justification for this initialization has been reported [12], each layer of the DBN-DNN is not the RBM. However, here we reconsider each layer of the DBN-DNN from the RBM's point of view, because the each layer is pre-trained as the RBM.

In the DNN, the output of the $(\ell+1)$-th layer's nodes are calculated by

$$\boldsymbol{q}_{\ell+1} \quad = \quad \boldsymbol{\sigma}(\boldsymbol{W}_\ell^T \boldsymbol{q}_\ell + \boldsymbol{b}_\ell)\,, \tag{11}$$

where $\boldsymbol{q}_\ell$ represents the output of the ℓ-th layer's nodes.

From the RBM's point of view, the state of each node is represented by the probabilities of the random binary variable. In the conventional DBN-DNN, the probabilities or the expectation of the random binary variables are used as the output of the nodes. Namely,

$$\boldsymbol{q}_\ell \quad = \quad \boldsymbol{P}(\boldsymbol{\xi}_\ell = \boldsymbol{1}) \;=\; E_{\boldsymbol{P}(\boldsymbol{\xi}_\ell)}[\boldsymbol{\xi}_\ell]\,, \tag{12}$$

where $\boldsymbol{\xi}_\ell$ represents the associated random binary variables. In this sense, we can consider that the probabilities of the $(\ell+1)$-th node's state are inferred by

$$\boldsymbol{P}(\boldsymbol{\xi}_{\ell+1} = \boldsymbol{1}) \quad = \quad \boldsymbol{\sigma}(\boldsymbol{W}_\ell^T E_{\boldsymbol{P}(\boldsymbol{\xi}_\ell)}[\boldsymbol{\xi}_\ell] + \boldsymbol{b}_\ell)\,. \tag{13}$$

Here, we discuss the inference of the DBN-DNN layer. But, we meet the same calculation in copying hidden states as shown in Fig. 1-(a) and the reconstruction in the CD training as shown in Fig. 2.

III. Proposed Inference of the RBM

We propose the novel inference of the RBM. As mention above, the inference of the RBM is one of keys in the DBN-DNN. First, we describe the proposed inference considering the probabilistic properties of the RBM. However, the rigorous calculation of the proposed inference is intractable. Therefore, the closed form approximation is also discussed. Then, we show the CD training and the DBN-DNN with the proposed inference.

A. Proposed Inference

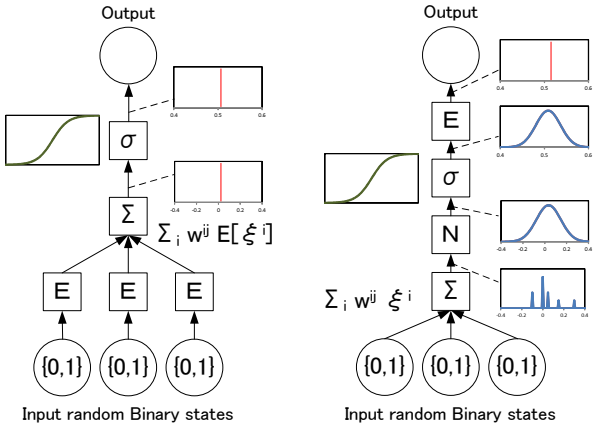
It is generally better to take expectation as late as possible. Therefore, we propose the inference of the RBM taking expectation after the activation function. The proposed inference is expressed by

$$\boldsymbol{P}(\boldsymbol{\xi}_{\ell+1} = \boldsymbol{1}) \quad = \quad E_{\boldsymbol{P}(\boldsymbol{\xi}_\ell)}\left[\boldsymbol{\sigma}(\boldsymbol{W}_\ell^T \boldsymbol{\xi}_\ell + \boldsymbol{b}_\ell)\right]\,. \tag{14}$$

The difference between the proposed inference in Eq. (14) and the conventional inference in Eq. (13) is place of taking the expectation. In the proposed inference, conditional probabilities of the $(\ell+1)$-th node's state given every possible combination of binary states of the ℓ-th node are evaluated. Then, the expectation of those evaluated conditional probabilities is evaluated to infer the probabilities of the $(\ell+1)$-th node's state.

As mentioned above, in the conventional inference in the CD training and the DBN-DNN, the real value of the probability is used as the input of the RBM, even though the RBM is developed only to model the random binary variables [1]. In the contrast, the proposed algorithm calculates the expectation of the probabilities after taking the activation function. It means that the conditional probabilities are evaluated with

[1]The RBM for the real values has been proposed. However, in this paper, we focus only the RBM for the random binary variables



(a) Conventional inference (b) Proposed inference

Fig. 3. Comparison of the inference processes.

the given random binary variables. This is mathematically rigorous. The proposed inference follows the assumption that the inputs of the RBM are the random binary variables, while the conventional inference violates that assumption. In this sense, the output of the proposed inference means the average probability, while the conventional inference does not.

However, the rigorous evaluation of the proposed inference is intractable, because the possible combination of the binary state is too huge. For the tractable evaluation, we follow the same manner of the closed form approximation of the fast drop-out training [4]. First, we introduce a random variable, $S^j_{\ell+1}$, associated to the input of the activation function of the j-th node in the $\ell+1$-th layer:

$$S^j_{\ell+1} \quad = \quad \boldsymbol{w}^{j\,T}_\ell \boldsymbol{\xi}_\ell + b^j_\ell \,, \tag{15}$$

where $\boldsymbol{w}^j_\ell$ is the j-th column vector of the matrix $\boldsymbol{W}_\ell$, and b^j_ℓ is the j-th element of the vector $\boldsymbol{b}_\ell$. Then, the density distribution of this random variable is approximated by Gaussian distribution

$$S^j_{\ell+1} \quad \sim \quad \mathcal{N}(S^j_{\ell+1}|\mu,\rho^2)\,, \tag{16}$$

where $\mathcal{N}(S^j_{\ell+1}|\mu,\rho^2)$ represents the Gaussian distribution whose mean is μ and variance is ρ^2. The mean and the variance are given by

$$\begin{aligned} \mu \quad &= \quad E[S^j_{\ell+1}] \;=\; E[\boldsymbol{w}^j_\ell \boldsymbol{\xi}_\ell + b^j_\ell] \\ &= \quad \sum_i w^{ij}_\ell p(\xi^i_\ell = 1) + b^j_\ell \,, \end{aligned} \tag{17}$$

$$\begin{aligned} \rho^2 \quad &= \quad V[S^j_{\ell+1}] \;=\; V[\boldsymbol{w}^j_\ell \boldsymbol{\xi}_\ell + b^j_\ell] \\ &= \quad \sum_i {w^{ij}_\ell}^2 \, p(\xi^i_\ell = 1)\{1 - p(\xi^i_\ell = 1)\}\,, \end{aligned} \tag{18}$$

where $V[x]$ represents the variance of the variable x, w^{ij}_ℓ is the (i,j)-th element of the matrix $\boldsymbol{W}_\ell$, and ξ^i_ℓ is the i-th element of the vector $\boldsymbol{\xi}_\ell$. Once we approximate the distribution of the input of the activation function by the Gaussian distribution, the expectation of the output of the activation function can be approximated by the following closed form calculation [4]:

$$\int_{-\infty}^{\infty} \sigma(x)\mathcal{N}(x|\mu,\rho^2)\,dx \quad \simeq \quad \sigma\left(\frac{\mu}{\sqrt{1+\pi\rho^2/8}}\right) \tag{19}$$

The resultant closed form approximation of the proposed inference in Eq. (14) can be represented by

$$\begin{aligned} p(\xi^j_{\ell+1} = 1) \quad &= \quad E[\sigma(\boldsymbol{w}^j_\ell \boldsymbol{\xi}_\ell + b^j_\ell)] \\ &\simeq \quad \sigma\left(\frac{\mu}{\sqrt{1+\pi\rho^2/8}}\right)\,, \end{aligned} \tag{20}$$

where μ and ρ^2 are presented in Eqs. (17) and (18), respectively.

The conventional inference in Eq. (13) with the same notation can be expressed by

$$p(\xi^j_{\ell+1} = 1) \quad = \quad \sigma\left(\mu\right)\,. \tag{21}$$

The conventional and proposed inference processes are summarized in Fig. 3. Comparing Eq. (20) and Fig. 3-(b) with Eq. (21) and Fig. 3-(a), one can find that the conventional inference only takes account of the mean or the expectation while the proposed inference involves the mean and the variance as the distribution.

B. The CD Training of the RBM

In order to apply the CD training, we need to calculate the hidden state $\boldsymbol{h}^0$, the reconstruction $\boldsymbol{v}^1$, and the associated hidden state $\boldsymbol{h}^1$ from the training data $\boldsymbol{v}^0$. In the proposed inference, we assume that the inputs are the random binary variables sampled with the given probabilities. For the first layer of the RBM, the training data is instance. In this sense, the probabilities associated to the input are not explicitly given. However, we can interpret the instance of the training data as the probability. For instance, if the training data is 1, the associated probability can be considered as $p(v=1)=1$. With the same manner, if the training data is 0, the associated probability can be considered as $p(v=1)=0$. Using this interpretation, we can evaluate the hidden state $\boldsymbol{h}^0$ with the proposed inference. In the proposed inference, the output is the average probability. Then, the reconstruction $\boldsymbol{v}^1$ and the associated hidden state $\boldsymbol{h}^1$ can be evaluated by applying the proposed inference step-by-step. For the second and subsequent layers, the hidden state inferred by the proposed algorithm can be used as probability of the input random binary state.

The hidden state inferred by the proposed algorithm is the average of the probability and we assume the input is the random binary variable sampled with the given probability. Therefore, the copy process in Fig. 1 is reasonable with the proposed inference. In contrast, the copy process with the conventional inference requires theoretical justification, because the hidden state inferred by the existing algorithm is the probability and the input is assumed as the random binary variable.

C. Back-propagation Learning of the DBN-DNN

The forward inference of the DBN-DNN with the proposed inference is straight forward. One can simply apply the proposed inference to the DBN-DNN. In the normal back-propagation for the DBN-DNN with the conventional inference, one only needs to back-propagate the derivatives for each hidden unit with input. For the fine-tuning of the DBN-DNN with the proposed inference, we need to back-propagate two sets of partial derivatives as mentioned in [4]. However, the back-propagation process can be derived with the same manner as the normal back-propagation.

The derivatives for the final layer can be expressed as follows:

$$\frac{\partial L}{\partial w_{ij}} = \delta_j \frac{\partial \mu_j}{\partial w_{ij}} + \tau_j \frac{\partial \rho_j^2}{\partial w_{ij}}, \tag{22}$$

$$\delta_j = \frac{\partial L}{\partial \mu_j} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial n_j}\frac{\partial n_j}{\partial \mu_j}, \tag{23}$$

$$\tau_j = \frac{\partial L}{\partial \rho_j^2} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial n_j}\frac{\partial n_j}{\partial \rho_j^2}, \tag{24}$$

where L represents the loss function, o_j is the output of j-th node, n_j is the input of the activation function of j-th node, and suffix i and j identify the input and the output nodes. A square difference or a cross entropy between the ground truth output and the inferred output are usually used for the loss function. In the proposed algorithm, the input of the activation function is

$$n = \frac{\mu}{\sqrt{1+\rho^2\pi/8}}. \tag{25}$$

For the hidden layer, the derivatives can be expressed as follows:

$$\frac{\partial L}{\partial w_{ij}} = \delta_j \frac{\partial \mu_j}{\partial w_{ij}} + \tau_j \frac{\partial \rho_j^2}{\partial w_{ij}}, \tag{26}$$

$$\delta_j = \alpha_j \frac{\partial n_j}{\partial \mu_j}, \tag{27}$$

$$\tau_j = \alpha_j \frac{\partial n_j}{\partial \rho_j^2}, \tag{28}$$

$$\alpha_j = \left[\sum_k \left\{\delta_k \frac{\partial \mu_k}{\partial o_j} + \tau_k \frac{\partial \rho_k^2}{\partial o_j}\right\}\right] \frac{\partial o_j}{\partial n_j}, \tag{29}$$

where δ_k and τ_k are propagated derivatives from the previous layer, and the suffix k identify the node of the previous layer. The derivatives of the bias term $\partial L/\partial b_j$ can be obtained similar manner. The parameters of the DBN-DNN with the proposed inference can be updated by iteratively back-propagating two set of derivatives δ and τ, as same as the normal back-propagation algorithm.

IV. Experimental Validation

We evaluate the proposed inference with two evaluation datasets; CIFAR-10 [14] and MNIST [15]. We apply the proposed inference in the pre-training by the CD training and the fine-tuning by the error back-propagation. The forward inference of the DBN-DNN is also used the proposed inference. The proposed algorithm is compared with the existing approach which consists of the pre-training by the CD training and the fine-tuning by the error back-propagation with the conventional inference. We use the cross entropy for the loss function of the fine-tuning. The matlab code of the proposed algorithm is available online [2].

The performance is evaluated with two metrics:

$$\text{rmse} = \sqrt{\frac{1}{MN}\sum_{k=1}^{N} ||\boldsymbol{y}_k - \boldsymbol{F}(\boldsymbol{x}_k)||_2^2}, \tag{30}$$

$$\text{err} = \frac{N_{inc}}{N}, \tag{31}$$

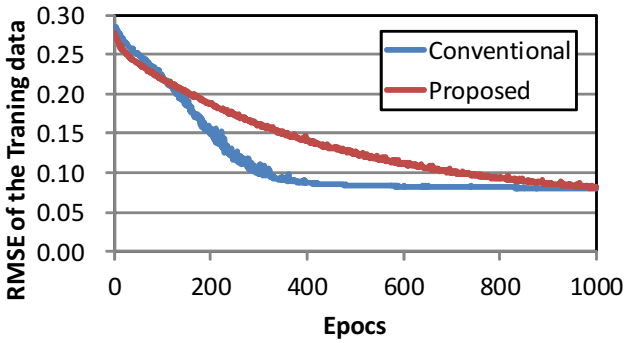
where M is the number of output classes, N is the number of data, N_{inc} is the number of incorrect inference, $\boldsymbol{x}_k$ represents the k-th input data, $\boldsymbol{y}_k$ represents the ground truth output, and $\boldsymbol{F}$ represents the inference of the trained DBN-DNN. The ground truth output is set as the only element associated to the true class is one and the other elements are zero. If the node which has the maximum output matches the true class, we count as correct.

A. Experiments on CIFAR-10

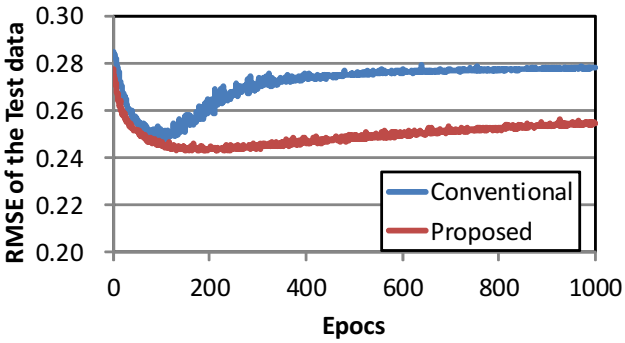
The tiny images dataset contains 80 million 32×32 color images collected from the web [16]. The CIFAR-10 dataset is a subset of the tiny images dataset which contains 60,000 images divided among ten classes [14]. Each class contains 5,000 training images and 1,000 testing images. Each image has $3,072 (= 32 \times 32 \times 3)$ dimensional data which represents three channels of intensity at each pixel. Those data are not binary data, but we use those data as probabilities after normalization to the range of $[0, 1]$ as same as in [4]. We build two-hidden-layer neural network with 3072-1000-1000-10 nodes. The RMSEs and the errors for the training data and the test data are summarized in Table I.

Figure 4 shows the convergence properties. In terms of the RMSE of the training data, the convergence of the proposed algorithm is slower than that of the existing algorithm. The convergence properties of the test data is more important than that of the training data. The convergence of the test data of the proposed algorithm is much better than that of the existing algorithm. The RMSE of the test data of the existing algorithm increases after 100 epochs. It means the existing algorithm over-fitted to the training data. After training of both DBN-DNN, we evaluate the error rate. The error rate is summarized in Table I. This comparison shows the proposed algorithm improves the performance of the DBN-DNN.

[2] We will put the URL information here for a camera ready after acceptance.



(a) Training data



(b) Test data

Fig. 4. Convergence properties of the training and the test data on CIFAR-10.

TABLE I. COMPARISONS ON CIFAR-10.

	Training data		Test data	
	RMSE	ERROR [%]	RMSE	ERROR [%]
Conventional inference	0.0806	6.35	0.278	47.5
Proposed inference	0.0813	3.14	0.255	45.6

TABLE II. COMPARISONS ON MNIST.

	Training data		Test data	
	RMSE	ERROR [%]	RMSE	ERROR [%]
Conventional inference	0.0188	0.298	0.0593	1.86
Proposed inference	0.0160	0.158	0.0547	1.76

B. *Experiments on MNIST*

The MNIST handwritten digit dataset consists of 28×28-sized black and white images, each containing a digit 0 to 9 (10-classes) [15]. It includes the 60,000 training images and the 10,000 test images. We build two-hidden-layer neural network with 784-1200-1200-10 nodes. The RMSEs and the errors for the training data and the test data are summarized in Table II. This comparison also demonstrate the proposed inference improve the performance of the DBN-DNN.

V. CONCLUSION

We have proposed the novel inference of the binary RBM. Although the input of the binary RBM is assumed to be random binary variable, the existing algorithms use continuous probability as the input of the binary RBM. The proposed inference has been derived considering the input of the binary RBM is the random binary variable. The proposed inference provides theoretically reasonable transition from the stacked RBM to the DBN-DNN, while the conventional inference cannot. We have also proposed the closed form approximation, because the straight forward derivation of the proposed algorithm is intractable. Experimental comparisons have demonstrated that the proposed inference improves the performance of the DBN-DNN. Our future works include to involve the proposed inference in a sparseness constrain training, the dropout, and the drop-connect techniques.

REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2," in *Advances in neural information processing systems*, 2007, pp. 873–880.

[3] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1058–1066.

[4] S. Wang and C. Manning, "Fast dropout training," in *International Conference on Machine Learning (ICML)*, 2013, pp. 118–126.

[5] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[6] K. Cho, A. Ilin, and T. Raiko, "Improved learning of gaussian-bernoulli restricted boltzmann machines," in *Artificial Neural Networks and Machine Learning (ICANN)*. Springer, 2011, pp. 10–17.

[7] M. Ranzato, V. Mnih, J. Susskind, and G. Hinton, "Modeling natural images using gated mrfs," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 9, pp. 2206–2222, 2013.

[8] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[9] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[10] T. Tieleman and G. Hinton, "Using fast weights to improve persistent contrastive divergence," in *International Conference on Machine Learning (ICML)*. ACM, 2009, pp. 1033–1040.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 1, p. 213, 2002.

[12] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[13] G. Hinton, "A practical guide to training restricted boltzmann machines," *UTML TR 2010-003*, 2010.

[14] A. Krizhevsky, "Learning multiple layers of featuresfrom tiny images," in *Master's thesis, Dept. of Comp. Sci., University of Toronto,*. IEEE, 2009.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] T. A., F. R., and F. W.T., "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 11, pp. 1958–1970, 2008.