MODELING DEEP BIDIRECTIONAL RELATIONSHIPS FOR IMAGE CLASSIFICATION AND GENERATION

Toru Nakashika

The Univeristy of Electro-Communications
Graduate School of Information Systems
nakashia@uec.ac.jp

Tetsuya Takiguchi, Yasuo Ariki

Kobe University
Graduate School of System Informatics
takigu@kobe-u.ac.jp, ariki@kobe-u.ac.jp

ABSTRACT

This paper presents a novel probabilistic model that represents a joint probability of two visible variables with a deep architecture, called a deep relational model (DRM). The model stacks several layers from one visible layer on to another visible layer, sandwiching hidden layers between them. As with restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs), all connections (weights) between two adjacent layers are undirected. During the maximum-likelihood (ML)-based training, the network attempts to capture latent complex relationships between two visible variables (e.g., an image showing a certain number and its corresponding label) thanks to its deep architecture. Unlike deep neural networks, 1) the proposed DRM is a totally generative model, and 2) the weights can be optimized in a probabilistic manner. This paper presents and discusses the experiments conduced to evaluate our DRM's performance in recognition and generation tasks.

Index Terms— Image classification, image generation, deep learning, generative model

1. INTRODUCTION

Since Hinton *et al.* introduced an effective pre-training algorithm for deep neural networks[1] (DNNs) using deep belief networks (DBNs) in 2006 [1], the use of deep learning has rapidly spread in the field of machine learning, artificial intelligence, signal processing, etc. A DBN is a graphical model that stacks restricted Boltzmann machines (RBMs) [2] layer-by-layer, each of which represents the probability distribution of visible variables with hidden variables. The effectiveness of using DBNs (or RBMs) has been proved especially in discriminative or deterministic tasks, such as handwritten character recognition [1], 3-D object recognition [3], machine transliteration [4], speech recognition [5], and voice conversion [6]. The discriminative tasks are generally achieved by setting the initial values of weights of a DNN as the trained weights of a DBN, and running back-propagation to fine-tune the DNN weights. This can be done due to the ability of deep learning that captures high-level abstractions at higher layers.

When it comes to the use of deep learning for generation tasks, we can find various models, such as a deep Boltzmann machines (DBM) [7], a denoising auto-encoder (DAE) [8], a shape Boltzmann machine (ShapeBM) [9], and a sum-product network (SPN) [10]. These models were mainly introduced to capture high-order abstractions for good representation of the observations, rather than for discriminative goal. Once obtaining high-level abstractions, we can, for instance, remove some noise on the observations, or restore missing parts in the observations.

Most of the existing deep-learning approaches focus on extracting high-order abstractions from one variable. In this paper, we try to capture the high-order relationships between two different types of variables based on deep learning. For that, we define a new probabilistic model called a deep relational model (DRM). A DRM is similar to an RBM and a DBM, each of which is a probabilistic model based on an energy function. The model sandwiches several hidden layers[2] between two visible layers and defines a joint probability for the two visible variables. Every two adjacent layers are connected with undirected weights, which are estimated so as to maximize the likelihood of the two visible variables. Interestingly, since the DRM is a totally generative model, it allows us not only to apply it to recognition tasks, but to also generate samples of one variable from the other variable. For example, considering that we have two kinds of variables for a hand-written digit image and a one-hot vector of the labels, we can estimate the label by inferring mean-field posteriors given an image (recognition task). On the other hand, by inferring posteriors given a label, we could obtain a generated image corresponding to the label (generation task).

2. RELATED WORK

In this section, we compare our proposed model, a deep relational model (DRM), with other related models: a restricted Boltzmann machine (RBM), a deep belief network (DBN) [1], a deep Boltzmann machine (DBM) [7], a deep energy model (DEM) [11], and a deep neural network (DNN). As shown in Figure 1, each model except an RBM has a deep architecture by stacking a visible layer $\boldsymbol{x}$ and multiple hidden layers $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots$ layer-by-layer with having connections between adjacent two layers, which has the capability of representing more complex data than an RBM that stacks a single hidden layer. A DNN and the proposed model further stack another visible variable $\boldsymbol{y}$ on the top. Therefore, these two models try to capture latent relationships between $\boldsymbol{x}$ and $\boldsymbol{y}$, while the other models just discover latent features or representation from $\boldsymbol{x}$.

An important factor in distinguishing each model is the direction of the connections between two adjacent layers. For example, a DBN has undirected connections at the top two layers, which form an RBM, and directed connections to the lower layers. A general DNN is a feedforward model; every two adjacent layers have deterministic weights in the direction from the source to the target variables. Meanwhile, the proposed DGM has totally bidirectional con-

[1]The term "neural networks" usually refers to a feedforward (directed) type of neural networks, and we also follow this here.

[2]When we give one hidden layer for our model, it is equivalent to an RBM with a concatenated vector of two visible variables. This will be discussed later.

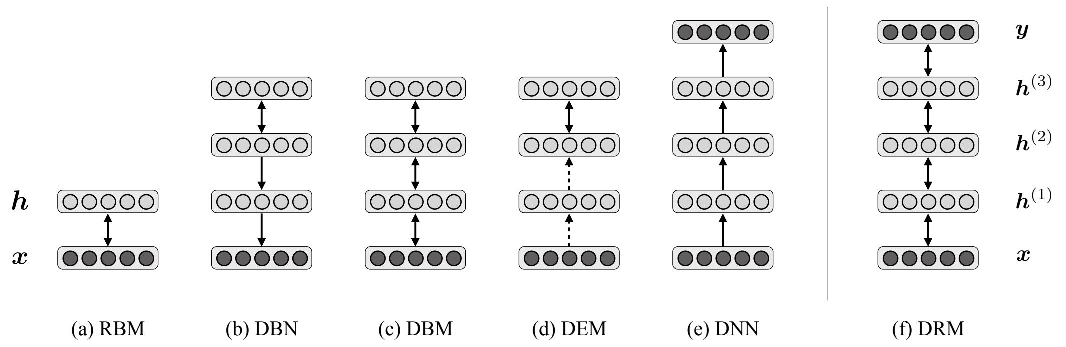


Fig. 1. Graphical representation of (a) a restricted Boltzmann machine, (b) a deep belief network, (c) a deep Boltzmann machine, (d) a deep energy model, (e) a deep neural network, and (f) the proposed model, a deep relational model. Two-way arrows and one-way arrows indicate undirected weights and directed weights, respectively. Dotted arrows represent deterministic relationships.

nections through all layers, just like a DBM does. This leads to the propagatation of information from the bottom up and from the top down in the network, while a DNN only infers from bottom to top. Assuming $\boldsymbol{x}$ and $\boldsymbol{y}$ indicate a vectorized image and a one-hot vector of the labels, a DGM allows us not only to estimate the label vector given an image, but also to generate an image from given a label vector.

Another aspect is the way parameters are estimated. Energy-based models, which include RBMs, DBMs, DEMs, and DGMs, are stochastic models in which the parameters are estimated so as to maximize the likelihood of observations[3]. On the other hand, the parameters of a DNN are optimized using back-propagation, where the errors between the output of the network and the target vector are minimized and propagated back to the lower layers. Since a stochastic model, such as a DGM, optimizes the parameters in a probabilistic framework, we can further extend the parameter estimation method to using maximum a posteriori (MAP), Bayesian inference, and so on.

Usually, deep-learning methods, such as a DBN, a DBM and a DEM, are used for the pre-training of a DNN. As reported in [1], a pre-trained DNN dramatically outperformed a randomly-initialized DNN. Generally speaking, in a deep network, error signals get weaker as they are back-propagated to the lower layer, which causes difficulties in estimating the parameters of the lower layer. Therefore, the pre-training approaches are considered to be effective in compensating for the *thin* gradients of the parameters. However, these approaches learn high-order representation in an unsupervised manner without knowing the existence of the target features. Therefore, it could be said that the learned weights are not necessarily appropriate for the initial values of a DNN that takes the target features into account. Our model, in contrast, connects with a visible layer for the target features and optimizes the parameters jointly, which may lead to better results compared with the above methods, even in a recognition task. Furthermore, our model is not adversely affected by the problems associated with DBMs. During the training of a DBM, it is difficult to estimate the weight parameters at the higher layers due to the fading gradients far from the visible layer [12]. On the contrary, our model sandwiches hidden layers with two visible layers at the opposite sides, and hence it propagates gradients more clearly top-to-bottom and bottom-to-top.

As for a DEM, Ngiam *et al.* also proposed a discriminative extension that considers target features in the model [11]. The model

[3]In practice, an approximation method is used.

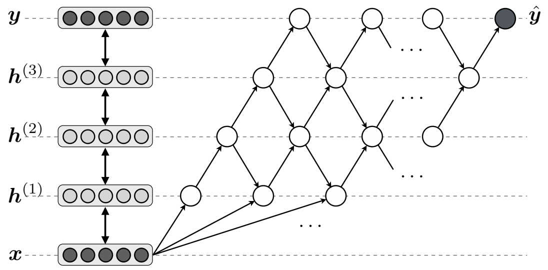


Fig. 2. Generating $\hat{\boldsymbol{y}}$ from $\boldsymbol{x}$ by repeating mean-field updates.

is, however, still discriminative; it does not have an ability to generate the source features from the target features. Furthermore, what the weights at the lower layers are trained without knowing about the target features also applies to this model.

3. DEEP RELATIONAL MODEL

Considering a dataset of images and its the labels, the labels should have been intentionally-, carefully-, and manually-assigned. As a result, there must be a strong correlation between an image and the assigned label. To capture latent, complicated, high-order relationships between two observable variables, such as an image and a one-hot vector of the label, we introduce a deep stochastic network called a deep relational model (DRM). The DRM will be defined as an energy-based model, which resembles a restricted Boltzmann machine (RBM) and a deep Boltzmann machine (DBM).

As shown in Figure 1 (f), a DRM is a deep network that sandwiches multiple hidden layers with two visible layers. As an energy-based model, a DRM defines a joint probability distribution of one (first) visible variables $\boldsymbol{x} \in \{0,1\}^I$ and the other (second) visible variables $\boldsymbol{y} \in \{0,1\}^K$ along with hidden variables $\boldsymbol{h}^{(l)} \in \{0,1\}^{J_l} (l = 1, \cdots, L)$, where L is the number of hidden layers. Similarly to an RBM and a DRM, each unit is only connected to the units at the adjacent layers, and is not connected to the units at the same layer. We define the joint probability distribution using a DRM as follows:

$$p(\boldsymbol{x}, \boldsymbol{y}; \theta) = \sum_{\forall \boldsymbol{h}^{(l)}} p(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)}; \theta) \tag{1}$$

$$p(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)}; \theta) = \frac{1}{Z(\theta)} e^{-E(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)}; \theta)}, \tag{2}$$

where the energy function E is defined as:

$$E(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)}; \theta) = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} - \sum_{l=1}^{L} {\boldsymbol{c}^{(l)}}^{\mathrm{T}} \boldsymbol{h}^{(l)} - \boldsymbol{d}^{\mathrm{T}}\boldsymbol{y}$$

$$- \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)} - \sum_{l=2}^{L} {\boldsymbol{h}^{(l-1)}}^{\mathrm{T}}\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l)} - {\boldsymbol{h}^{(L)}}^{\mathrm{T}}\boldsymbol{W}^{(L+1)}\boldsymbol{y}.$$

In addition to the previously-defined parameters $\boldsymbol{b}$, $\boldsymbol{c}^{(l)}$, and $\boldsymbol{W}^{(l)}$, the bias parameters for the second visible variables $\boldsymbol{d} \in \mathbb{R}^{K}$ are used. $\boldsymbol{W}^{(L+1)} \in \mathbb{R}^{J_l \times K}$ is the connection weights between the highest hidden layer and the second visible layer.

Each conditional distributions given the units at the adjacent layers can be computed as:

$$p(x_i = 1 | \boldsymbol{h}^{(1)}) = \sigma(b_i + \boldsymbol{W}_{i:}^{(1)}\boldsymbol{h}^{(1)}) \tag{3}$$

$$p(h_j^{(l)} = 1 | \boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l+1)}) = \sigma(c_j^{(l)} + {\boldsymbol{W}_{:j}^{(l)}}^{\mathrm{T}}\boldsymbol{h}^{(l-1)} + \boldsymbol{W}_{j:}^{(l+1)}\boldsymbol{h}^{(l+1)}) \tag{4}$$

$$p(y_k = 1 | \boldsymbol{h}^{(L)}) = \sigma(d_k + {\boldsymbol{W}_{:k}^{(L)}}^{\mathrm{T}}\boldsymbol{h}^{(L)}). \tag{5}$$

Note that the conditional probabilities of $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(L)}$ can be calculated from Eq. (4) by regarding as $\boldsymbol{h}^{(0)} = \boldsymbol{x}$ and $\boldsymbol{h}^{(L+1)} = \boldsymbol{y}$, respectively. Although the joint configuration of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined in a DRM, the first variable $\boldsymbol{x}$ is not directly connected to the second variable $\boldsymbol{y}$, and $\boldsymbol{y}$ is not required to infer $\boldsymbol{x}$, as Eq. (3) indicates. Through hidden layers, $\boldsymbol{x}$ and $\boldsymbol{y}$ propagate their information to each other layer-by-layer. Therefore, the network models deep latent correlations between $\boldsymbol{x}$ and $\boldsymbol{y}$. That means the trained network has the ability to estimate one variable given the other variable. To estimate variable $\hat{\boldsymbol{y}}$ given $\boldsymbol{x}$, for example, we use an iterative mean-field update approach, as shown in Figure 2. In this procedure, we first compute the expectations (mean-field approximation) for each hidden layer's unit from bottom to top, as in Eq. (4), regarding all the values of the units at the upper layer as zero. Then, we calculate the expectations of hidden units using the previously-calculated values for $\boldsymbol{h}^{(l-1)}$ and $\boldsymbol{h}^{(l)}$ in Eq. (4). We iterate this procedure T times with clamping the values of $\boldsymbol{x}$ (in our experiments, we used $T = 10$). Finally, we obtain the expected values of $\boldsymbol{y}$ by calculating $\mathbb{E}[\boldsymbol{y}|\boldsymbol{x}] \approx \mathbb{E}[\boldsymbol{y}|\hat{\boldsymbol{h}^{(L)}}] = \sigma(\boldsymbol{d} + {\boldsymbol{W}^{(L)}}^{\mathrm{T}}\hat{\boldsymbol{h}^{(L)}})$, where $\hat{\boldsymbol{h}^{(L)}}$ is the lastly-updated $\boldsymbol{h}^{(L)}$ after the iteration.

For parameter estimation, the joint log-likelihood of $\boldsymbol{x}$ and $\boldsymbol{y}$ is used. Differentiating partially with respect to each parameter, we obtain:

$$\frac{\partial \log p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{\partial b_i} = \langle x_i \rangle_{\mathrm{data}} - \langle x_i \rangle_{\mathrm{model}} \tag{6}$$

$$\frac{\partial \log p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{\partial c_j^{(l)}} = \langle h_j^{(l)} \rangle_{\mathrm{data}} - \langle h_j^{(l)} \rangle_{\mathrm{model}} \tag{7}$$

$$\frac{\partial \log p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{\partial d_k} = \langle y_k \rangle_{\mathrm{data}} - \langle y_k \rangle_{\mathrm{model}} \tag{8}$$

$$\frac{\partial \log p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{\partial W_{ij}^{(l)}} = \begin{cases} \langle x_i h_j^{(1)} \rangle_{\mathrm{data}} - \langle x_i h_j^{(1)} \rangle_{\mathrm{model}} & (l = 1) \\ \langle h_i^{(l-1)} h_j^{(l)} \rangle_{\mathrm{data}} - \langle h_i^{(l-1)} h_j^{(l)} \rangle_{\mathrm{model}} & (l = 2, \cdots, L) \\ \langle h_i^{(L)} y_j \rangle_{\mathrm{data}} - \langle h_i^{(L)} y_j \rangle_{\mathrm{model}} & (l = L + 1) \end{cases}, \tag{9}$$

where $\langle\cdot\rangle_{\mathrm{data}}$ and $\langle\cdot\rangle_{\mathrm{model}}$ indicate the expectations of the empirical data and the inner model, respectively. As mentioned before, the second terms are computationally difficult. Therefore, we approximate the second terms with the expectations of the reconstracted data $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ that are sampled from the iteratively-updated inner model (Figure 3). The iterative procedure is similar to the generation scheme shown in Figure 2, but we use the empirical values of $\boldsymbol{y}$ during iteration. After updating each expectated value of hidden units T times, we sample $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ using Eqs (3)(5).

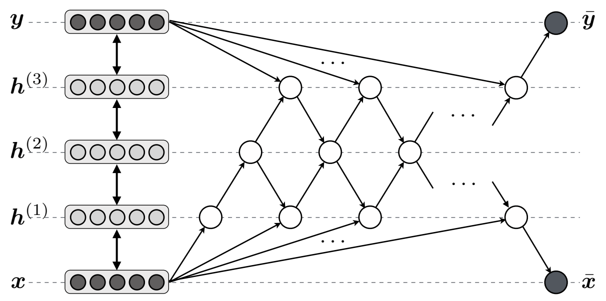


Fig. 3. Iterative inference using mean-field updates. White circles and black circles indicate mean-field inference and randomly-generated samples with their probabilities, respectively.

To boost up parameter optimization, we further use a pre-training scheme. In this scheme, before training each parameter of a DRM, we perform training of RBMs. Similar to the greedy-wise training used in a DBN [1], we first train RBMs at the lowest ($\boldsymbol{x}$ and $\boldsymbol{y}$) levels. The second RBMs use the expected values of the hidden units inferred from the first RBMs for the values of visible units. In this way, we perform pre-training from outer to inner. In the training stage of a DRM, the parameters obtained in the pre-training are set to the initial values of the training.

When we want to do image recognition tasks, we can estimate the label $\boldsymbol{y}$ given an image $\boldsymbol{x}$, as discussed in the previous subsection (see Figure 2). However, we can also employ a fine-tuning scheme. After the training of the DRM, we fine-tune each parameter using back-propagation, treating it as a discriminative DNN.

4. EXPERIMENTS

To evaluate our method and examine its potential, we conducted recognition and generation experiments using the MNIST dataset[4]. The dataset contains 60,000 training and 10,000 test images of hand-written digits (0-9) with a size of 28 × 28 pixels, along with the label data. A classification with only ten classes is not a hard task, and if we use the full training set of a large number of data, the error rate becomes very low, which may make it difficult to evaluate and compare each method. Therefore, we reduced the number of training data to 10,000 randomly-selected images. To speed-up learning, we divided the training data into mini-batches, each of which contained 20 data, and trained the model with a learning rate of 0.1 in 50 epochs.

4.1. Recognition Task

For all the experiments, we used a four-layer network: the first visible layer, two hidden layers, and the second visible layer. In our first recognition experiments, we investigated how the model performs

[4]The MNIST dataset is available at http://yann.lecun.com/exdb/mnist/index.html.

Table 1. Error rate [%] when changing the number of hidden units at the 1st and the 2nd hidden layers.

1st \ 2nd	50	100	200	300	400	500
200	11.68	10.1	8.97	8.34	10.02	6.98
300	9.12	7.74	6.43	5.95	7.30	6.59
400	7.18	6.97	5.22	4.94	**3.94**	4.75
500	8.35	7.75	5.79	4.67	4.51	4.87

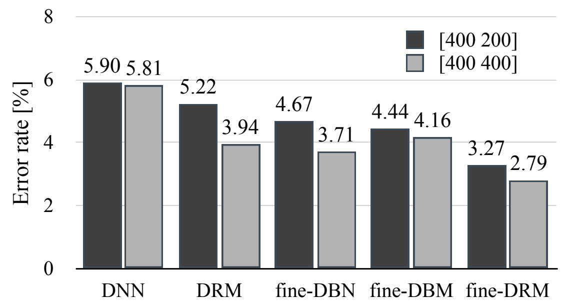
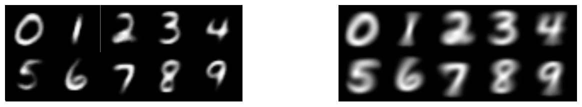


Fig. 4. Error rate [%] for the MNIST dataset obtained by each method.

when changing the number of hidden units from 200 to 500 for the first hidden layer, and from 50 to 500 for the second hidden layer. The number of first and the second visible units are $28 \times 28 = 784$ and 10, respectively. The experimental results are summarized in Table 1. As shown in Table 1, the error rate was most improved when 400 hidden units were used for both hidden layers. As indicated, the large number of hidden units did not necessarily improve the error rate. Interestingly, the networks that have more units at the first hidden layer than the second hidden layer produced better results than their counterparts that have the same number of parameters but more units at the second hidden layer than the first layer (e.g., the network of 500 1st-hidden units and 300 2nd-hidden units outperformed the network of 300 1st-hidden units and 500 2nd-hidden units, 4.67% compared to 6.59%). This is because the first visible layer has more units than the second visible layer, and if the network has fewer units at the higher layers, the information at the lower layers is gradually propagated and compressed smoothly as going up.

Secondly, we compared our method with three conventional methods: a DNN, a DBN, and a DBM. We used the same conditions for the conventional methods as for our model; 10,000 training data, a learning rate of 0.1, a batch-size of 20, and the number of 50 for epochs were used. Each network-based method has two hidden layers, and was compared in the case of 400 1st-hidden-layer units and 200 2nd-hidden-layer units ("[400 200]") with the case of 400 1st-hidden-layer units and 400 2nd-hidden-layer units ("[400 400]"). After training the DBN, the DBM, and the DRM, we fine-tuned their parameters using back-propagation of 10 epochs (noted as "fine-DBN","fine-DBM", and "fine-DRBM", respectively, in Figure 4), while a DNN trained the parameters starting from randomly-initialized values. Figure 4 shows the comparison results. Our model "DRM" used the mean-field-update scheme to estimate $\boldsymbol{y}$ (Figure 2), and "fine-DRM" used the fine-tuning scheme and produced the label vectors in a feedforward DNN). As shown in Figure 4, our model "fine-DRM" performed best of all in both cases



(a) DRM (b) Mean

Fig. 5. Generated images given each one-hot label using our model (left), and mean values over the training data calculated for each label (right).

of "[400 200]" and "[400 400]". This is due to the fact that a DRM models a route from an image to the label in the stochastic training, while a DBN and DBM do not. As observed, the DRM is a bidirectional generative model, and if the parameters are specialized and tuned as a directional, discriminative model (i.e., "fine-DRM"), the performance was improved. An interesting observation is that the performance of our model without fine-tuning is comparable to that of the other fine-tuned models (3.94% compared to 3.71% of a DBN and to 4.16% of a DBM), even though it is a still generative model.

4.2. Generation Task

As we generate the label given an image using a DRM, it will be possible to generate the image given a label, because a DRM models the joint distribution of the two. To examine the potential of this possibility, we conducted a generation experiment. In this experiment, we used a [400 200] architecture and generated images $\boldsymbol{x}$ given the one-hot labels $\boldsymbol{y}$ through mean-field updates in a similar manner to the generation scheme (reverse up and down in the left side of Figure 2). Essentially, this procedure generates an image from $\boldsymbol{y}$; however, the dimension of 10 for the vector $\boldsymbol{y}$ is too small to estimate the upper features of 200, 400, and 784 units properly through the iterative updates. Therefore, we gave the initial values of hidden units as the means of the hidden units calculated from the training data. After that, we obtained the images for each one-hot vector of the labels as shown in Figure 5 (a). To compare the images, we list images obtained from just calculating the means of each pixel for each label in Figure 5 (b). As shown in Figure 5, the images from Mean are blurred and obscure. On the other hand, the images from a DRM are very clear and sharpened, much like real handwritten digits.

5. CONCLUSION

In this paper, we proposed a new joint distribution model of two variables that has a hierarchical architecture to capture latent, complicated, high-order relationships between the two. The proposed model, a deep relational model (DRM), is viewed as one of the energy-based models, and the parameters are trainable using maximum likelihood with mean-field approximation. In our recognition experiments, we showed its high performance, especially when fine-tuned. In the generation experiments, we obtained interesting images generated from the trained DRM. In the future, we will further investigate its potential, focusing particularly on generative tasks.

6. REFERENCES

[1] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[2] Paul Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," *Parallel Distributed Processing*, vol. 1, 1986.

[3] Vinod Nair and Geoffrey Hinton, "3-d object recognition with deep belief nets," *Advances in Neural Information Processing Systems*, vol. 22, pp. 1339–1347, 2009.

[4] Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney, "A deep learning approach to machine transliteration," in *Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2009, pp. 233–241.

[5] Abdel-rahman Mohamed, George E. Dahl, and Geoffrey Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[6] Toru Nakashika, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki, "Voice conversion in high-order eigen space using deep belief nets.," in *INTERSPEECH*, 2013, pp. 369–372.

[7] Ruslan Salakhutdinov and Geoffrey E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[8] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent, "Generalized denoising auto-encoders as generative models," vol. 26, pp. 899–907, 2013.

[9] SM Ali Eslami, Nicolas Heess, Christopher KI Williams, and John Winn, "The shape boltzmann machine: a strong model of object shape," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 155–176, 2014.

[10] Hoifung Poon and Pedro Domingos, "Sum-product networks: A new deep architecture," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2011, pp. 689–690.

[11] Jiquan Ngiam, Zhenghao Chen, Pang W. Koh, and Andrew Y. Ng, "Learning deep energy models," in *28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1105–1112.

[12] Ruslan Salakhutdinov and Hugo Larochelle, "Efficient learning of deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 693–700.