Capsule Deep Generative Model That Forms Parse Trees

Yifeng Li
Department of Computer Science
Brock University
St. Catharines, Canada
yli2@brocku.ca

Xiaodan Zhu
Department of Electrical and Computer Engineering
Queen's University
Kingston, Canada
xiaodan.zhu@queensu.ca

Richard Naud
Brain and Mind Research Institute
University of Ottawa
Ottawa, Canada
rnaud@uottawa.ca

Pengcheng Xi
Digital Technologies Research Centre
National Research Council Canada
Ottawa, Canada
pengcheng.xi@nrc-cnrc.gc.ca

***Abstract*—Supervised capsule networks are theoretically advantageous over convolutional neural networks, because they aim to model a range of transformations of local physical or abstract objects and part-whole relationships among them. However, it remains unclear how to use the concept of capsules in deep generative models. In this study, to address this challenge, we present a statistical modelling of capsules in deep generative models where distributions are formulated in the exponential family. The major contribution of this unsupervised method is that parse trees as representations of part-whole relationships can be dynamically learned from the data.**

***Index Terms*—deep learning, generative model, capsule net, parse tree, exponential family**

I. Introduction

Deep neural networks and learning algorithms have achieved grand successes in a variety of areas where data are complicated and plenty [1]–[3]. Behind these achievements, the spirit is distributed representation learning [4], which says, any discrete object, symbols or sequence in our world can be represented by a vector of continuous values. This philosophy is strongly supported by its modern implementations in natural language processing through word embedding [5], [6], and computer vision through convolutional neural networks [7], [8].

However, distributed representation does not mean that structured latent representations are not important in neural networks. Actually, it is an open and foundational question. At least, the huge adoption of convolutions in various types of data beyond images indicates that sophisticated structures often help achieve state of the art performance. A common struggle that current supervised and unsupervised neural networks are facing is that complicated symbolic manipulations and reasonable combinations of local patterns can not be well handled. The introduction of capsule networks stirs new thoughts on how local parts are learned and combined to form a cohesive role, and how features are disentangled within capsules. In the supervised setting, a capsule is a set of neural units that are activated or depressed together [9], [10]. While a single unit is barely useful in representing comprehensive information, a set of units is. The introduction of capsules goes far beyond an improvement of convolutions. It advocates a family of models that carves the flow of information within neural networks. Interestingly, the dynamic routing of capsules in these deterministic capsule networks are realized using generative techniques such as expectation maximization (EM) [10], because it is in fact an inference problem. This naturally inspires the studies of capsules in generative models. In the simplest case, exponential family restricted Boltzmann machine (exp-RBM) and Helmholtz machine (exp-HM) are extended to capsule RBM (cap-RBM) and HM (cap-HM) by replacing unit latent variables with stochastic capsules [11]. Although capsules in these models can be contextually activated, part-whole trees, i.e. parse trees, are not formed in the latent space.

To push forward the front line of research in this direction, we propose a new exponential family deep generative model in this paper to automatically and dynamically build parse tree among stochastic capsules for a given sample. The paper is organized as follows. Insights into closely related work is provided in Section II. The proposed model and learning rules are described in Section III. Preliminary experiments are reported in Section IV, followed by a discussion (V).

II. Related Work

A. Transforming Auto-Encoder

The idea of capsule neural networks dates back to the preliminary introduction of transforming auto-encoder (TAE) in 2011 [12], which however did not capture much attention at the time. A schematic example of the TAE network is displayed in Fig. 1. The TAE network only has one layer of capsules. Each capsule consists of two recognition hidden layers and one generative hidden layer. The output of each

This research is supported by the New Beginning Ideation Fund from the National Research Council Canada.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

recognition component is a 3 by 3 pose matrix $\boldsymbol{M}$, and an activation probability a. The pose matrix is then multiplied by the affine or 3D transformation matrix $\boldsymbol{W}$. The transformed matrix is then fed into the generative component to produce a patch within an reconstructed image. The patch is multiplied by the activation probability a. A pose matrix or vector may potentially represent any properties of a visual entity. A vote matrix is transformed from the pose matrix. TAE can be viewed as a prototype of generative-like architecture with capsules for image generation. However, the 3 by 3 transformation matrix is not learned from data, but instead is predefined when making the input-output pair of each training image. Thus, the real application of TAE is very limited.

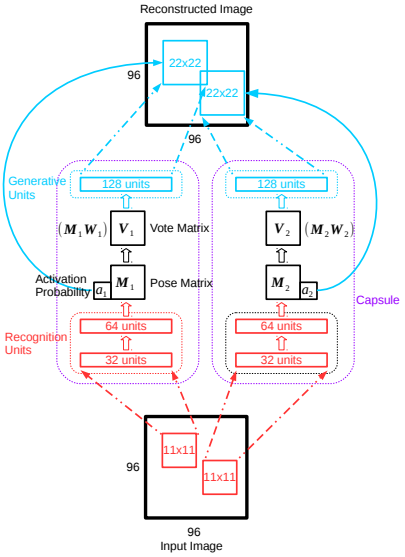


Fig. 1: Example of transforming autoencoder presented in [12].

B. Vector Capsule Net

Fig. 2 shows an example of vector capsule net which is proposed in [9]. Each capsule here takes all transformed output of lower capsules as input, and output a vector. The capsule is expected to encapsulate pose information (such as position, orientation, scaling, and skewness) and instantiation parameters (such as color and texture). Formally, the j-th capsule at the $(l+1)$-th layer (denoted by $\boldsymbol{h}_j^{(l+1)}$) is computed using the following equations

$$\boldsymbol{h}_{i,j}^{(l)} = \boldsymbol{W}_{i,j}\boldsymbol{h}_i^{(l)} \quad \forall i \text{ in } \{1, \cdots, K_l\} \tag{1}$$

$$\boldsymbol{s}_j^{(l+1)} = \sum_{i=1}^{K_l} c_{i,j}\boldsymbol{h}_{i,j}^{(l)} \tag{2}$$

$$\boldsymbol{h}_j^{(l+1)} = \frac{\|\boldsymbol{s}_j^{(l+1)}\|^2}{1+\|\boldsymbol{s}_j^{(l+1)}\|^2}\frac{\boldsymbol{s}_j^{(l+1)}}{\|\boldsymbol{s}_j^{(l+1)}\|}, \tag{3}$$

where K_l is the total number of capsules in the l-th layer, $\boldsymbol{W}_{i,j}$ is a transformation matrix that is specific between capsules $\boldsymbol{h}_i^{(l)}$ and $\boldsymbol{h}_j^{(l+1)}$ and is learned using back-propagation, $c_{i,j}$ is used as a coupling coefficient and is computed using the softmax function

$$c_{i,j} = \frac{e^{b_{i,j}}}{\sum_{j'=1}^{K_{l+1}} e^{b_{ij'}}}, \tag{4}$$

where $b_{i,j}$ is determined using a dynamic routing algorithm based on inner product (cosine) $\boldsymbol{h}_j^{(l+1)^{\mathrm{T}}}\boldsymbol{h}_{i,j}^{(l)}$. Length of a capsule serves as activation probability of the capsule. Thus, there is no explicit activation unit for this method. The objective function in the vector capsule network is defined by a margin loss which uses the l_2 norm as activation probability of a digit capsule:

$$L = \sum_{c=1}^{C} L_c \tag{5}$$

where

$$\begin{aligned} L_c = {} & \delta(y-c)\max(0, m_+ - \|\boldsymbol{h}_c\|_2)^2 \\ & + \lambda(1-\delta(y-c))\max(0, \|\boldsymbol{h}_c\|_2 - m_-)^2, \end{aligned} \tag{6}$$

where y is the real class label, $\boldsymbol{h}_c$ is a digital capsule, $\delta(y-c) = 1$ if and only if $y = c$, $m_+ = 0.9$ and $m_- = 0.1$, and $\lambda = 0.5$. In Fig. 2, dynamic routing is used between the primary capsule layer and the digital capsule layer. Thus, the $6 \times 6 \times 32$ lower-level capsules dynamically connect to the 10 higher-level capsules.

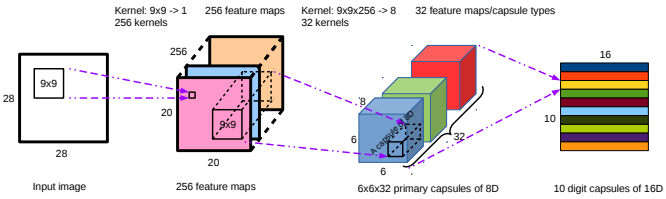


Fig. 2: Schematic example of vector capsule net [9].

This simple capsule network is the first practical method that outperforms convolutional network with the advantage of learning a range of transformations. Even though it is a supervised feedforward neural network, a decoder network is also used in the structure to obtain the reconstruction error which serve as a regularization term in the supervised objective function.

C. Matrix Capsule Net

In the matrix capsule network proposed in [10], a (matrix) capsule takes the vote matrices (transformed from pose matrices) and activation probabilities of the lower-level capsules as input, and outputs a pose matrix and an activation probability. An example of the matrix capsule network is given in Fig. 3. EM-routing is used to control interactions between two capsule layers. Since convolution is used between two capsule layers, a lower-level capsule can only see at most $K \times K$ higher-level capsules. A higher-level capsule can only see lower-level capsules from its receptive field. The transformation matrices to a feature map are shared. Between the last convolutional

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

capsule layer and class capsules, a class capsule can see all lower-level capsules. Transformation matrices from the same feature map are shared to the class capsules.

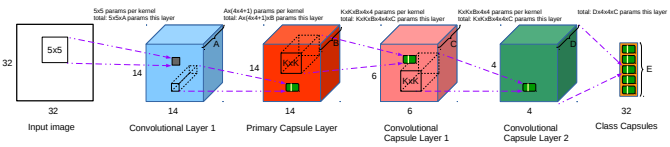


Fig. 3: Schematic example of matrix capsule net [10].

Suppose capsule C_j is the j-th capsule in the $l+1$ layer, and capsule C_i is the i-th capsule in the l-th layer. The connections from C_i to C_j is represented by transformation matrix $\boldsymbol{W}_{ij}$, which transforms the pose matrix $\boldsymbol{H}_i$ to vote matrix $\boldsymbol{V}_{ij}$ using $\boldsymbol{V}_{ij} = \boldsymbol{H}_i\boldsymbol{W}_{ij}$. Each capsule has two bias parameters: β_a and β_u, which combine with the vote matrices from the lower-level capsules to determine the activation probability of the capsule. The biases β_a and β_u can be respectively the same for all capsules. This process is visualized in Fig. 4. Thus, the parameters of a capsule network include transformation matrices on the connections and biases on the capsules.

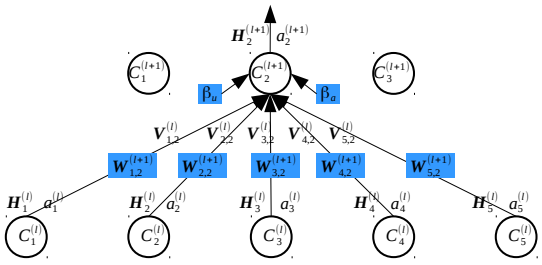


Fig. 4: Input and output of a matrix capsule. Each node represents a matrix capsule.

A capsule network is composed of multiple capsule layers. The outputs of capsules in layer $l+1$ is based on the output of capsules in layer l. The model parameters are learned using back-propagation. A cost function is defined by comparing the actual class label with the activations of the class capsules. The spread loss function used in [10] is defined as

$$L = \sum_{c=1, c\neq t}^{C} \Big(\max\big(0, m - (a_t - a_c)\big)\Big)^2, \tag{7}$$

where a_t is the activation of the capsule corresponding to the actual class t, a_c's are activations of other classes, and m is a margin which increases from 0.2 to 0.9. Cross-entropy loss can also be used even though $\sum_{c=1}^{C} a_c \neq 1$. To build the part-whole relationship among capsules, EM routing is proposed to dynamically activate parent capsules.

D. Generative Capsule Nets

The statistical modelling of capsules has been explored based on the exponential family RBMs and HMs [13]–[15]. The main idea of the capsule RBM (cap-RBM) proposed in [11] is to replace individual latent variables with stochastic capsules, each of which is composed of a vector ($\boldsymbol{h}_k$) of hidden random variables following exponential family distributions, and a Bernoulli variable (z_k) as activity indicator of the capsule. An example of such network is displayed in Fig. 5a. Assume the vector of variables $\boldsymbol{x}$ has M units, and the hidden layer has K capsules, each of which has J variables and one switch variable. The base distributions of capsule RBM are defined in natural form of exponential family as

$$p(\boldsymbol{x}) = \prod_{m=1}^{M} \exp\big(\boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{s}_m + \log f(x_m) - A(\boldsymbol{a}_m)\big) \tag{8}$$

$$p(\boldsymbol{h}) = \prod_{k=1}^{K}\prod_{j=1}^{J} \exp\big(\boldsymbol{b}_{k,j}^{\mathrm{T}}\boldsymbol{t}_{k,j} + \log g(h_{k,j}) - B(\boldsymbol{b}_{k,j})\big) \tag{9}$$

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \exp\big(c_k z_k - C(z_k)\big), \tag{10}$$

The joint distribution is formulated as an energy-based distribution: $p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z}) = \frac{1}{Z}\exp\big(-E(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z})\big)$, where Z is the partition function, and $E(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z})$ is the energy function defined as

$$\begin{aligned} E(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z}) = &-\sum_{m=1}^{M}\big(\boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{s}_m + \log f(x_m)\big) \\ &-\sum_{k=1}^{K}\sum_{j=1}^{J}\big(\boldsymbol{b}_{k,j}^{\mathrm{T}}\boldsymbol{t}_{k,j} + \log g(h_{k,j})\big) - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{z} \\ &-\sum_{k=1}^{K} z_k(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}_k\boldsymbol{h}_k). \end{aligned} \tag{11}$$

Inheriting the decomposibility of exp-RBMs, the conditionals of cap-RBM can be obtained in natural forms:

$$p(\boldsymbol{x}|\boldsymbol{h},\boldsymbol{z}) = \prod_{m=1}^{M} p\big(x_m|\eta(\hat{\boldsymbol{a}}_m)\big), \tag{12}$$

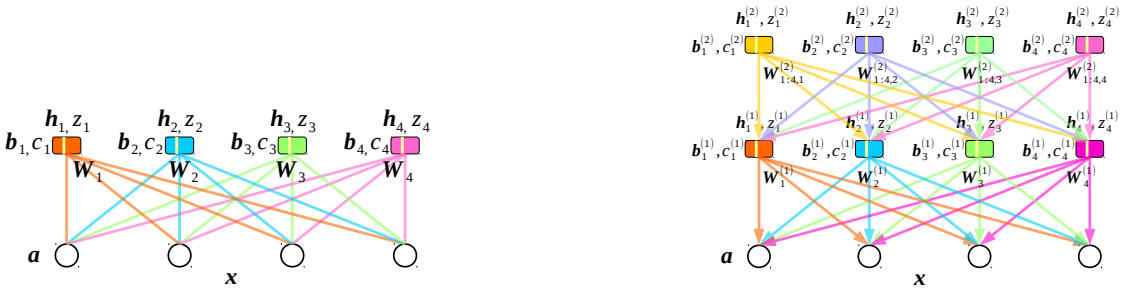
$$p(\boldsymbol{h}|\boldsymbol{x},\boldsymbol{z}) = \prod_{k=1}^{K}\prod_{j=1}^{J} p\big(h_{k,j}|\eta(\hat{\boldsymbol{b}}_{k,j})\big), \tag{13}$$

$$p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{h}) = \prod_{k=1}^{K} \mathcal{BE}\big(z_k|\eta(\hat{c}_k)\big), \tag{14}$$

where function $\eta(\cdot)$ maps the natural posterior parameters to the standard forms.

One advantage of cap-RBM is that the activity of capsules are dynamically inferred. Different samples activate different subset of capsules. Innovation is required to generalize the concept of cap-RBM to deep generative models. Capsule HM (cap-HM) is the most straightforward generalization of cap-RBM for directed multi-layer capsule models. In cap-HM, the joint distribution of visible variables and L layers of latent vectors is factorized as:

$$\begin{aligned} p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z}) = &p(\boldsymbol{x}|\boldsymbol{h}^{(1)},\boldsymbol{z}^{(1)})\Big(\prod_{l=1}^{L-1} p(\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}|\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)})\Big) \\ &p(\boldsymbol{h}^{(L)},\boldsymbol{z}^{(L)}). \end{aligned} \tag{15}$$



(a) Capsule RBM. (b) Capsule HM.

Fig. 5: Examples of capsule RBM and capsule HM. The meanings of the variables and parameters in capsule RBM are explained as follows. $\boldsymbol{a}$: bias parameters on the visible units. $\boldsymbol{b}_k$: bias parameters on $\boldsymbol{h}_k$. c_k: bias parameter on z_k. $\boldsymbol{W}_k$: interaction matrix between the k-th capsule ($\boldsymbol{h}_k$ and z_k) and $\boldsymbol{x}$. Similar meanings apply to capsule HM.

In this factorization,

$$p(\boldsymbol{x}|\boldsymbol{h}^{(1)}, \boldsymbol{z}^{(1)}) = \prod_{m=1}^{M} p(x_m|\eta(\hat{\boldsymbol{a}}_m)), \tag{16}$$

is the same as in cap-RBM, the immediate component

$$p(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}|\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}) = \frac{1}{Z_l} \exp\big(-E(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}|\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)})\big), \tag{17}$$

is actually an energy-based distribution, and

$$p\big(\boldsymbol{h}^{(L)}, \boldsymbol{z}^{(L)}\big) = p\big(\boldsymbol{h}^{(L)}\big)p\big(\boldsymbol{h}^{(Z)}\big). \tag{18}$$

A nice benefit of using cap-HM is that a sample will only activate a subset of capsules in each layer, offering a novel way to visualize the activity spectrum in latent space. However, it does not construct a parse tree which is essential to represent the part-whole relationship. In another word, each capsule at layer l can be connected to multiple capsules at layer $l+1$ in cap-HM. Our work presented below addresses this challenge.

III. Method

Our goal in this paper is to design a capsule DGM (cap-DGM), in the exponential family, that can form parse trees. An example of the desired capsule DGM is illustrated in Fig. 6.

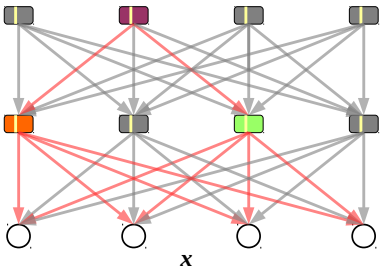


Fig. 6: An schematic example of capsule DGM that forms a parse-tree in latent space.

This model has four set of variables. The visible variable is denoted by $\boldsymbol{x}$. The capsules in all hidden layers are denoted by $\boldsymbol{h} = \{\boldsymbol{h}^{(1)}, \cdots, \boldsymbol{h}^{(L)}\}$ where $\boldsymbol{h}_k^{(l)}$ represents the k-th capsule in the l-th layer and its corresponding activation variable is denoted by $z_k^{(l)}$. The parse tree structure in the network is realized by $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}^{(1)}, \cdots, \boldsymbol{\alpha}^{(L-1)}\}$ where $\boldsymbol{\alpha}^{(l)} = \{\boldsymbol{\alpha}_1^{(l)}, \cdots, \boldsymbol{\alpha}_{K_l}^{(l)}\}$ where $\boldsymbol{\alpha}_k^{(l)}$ is a vector variable of length K_{l+1} (following Multinoulli distribution) indicating which parent capsule at upper layer should capsule $C_k^{(l)}$ be assigned to. That is, if $\alpha_{k_1,k_2}^{(l)} = 1$, then capsule $C_{k_1}^{(l)}$ at the l-th layer is assigned to parent capsule $C_{k_2}^{(l+1)}$ at layer $l+1$. According to the model's graphical structure, the joint distribution of this model can be factorized into product of three components:

$$p(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{z}, \boldsymbol{\alpha}) = p(\boldsymbol{x}|\boldsymbol{h}^{(1)}, \boldsymbol{z}^{(1)}) \cdot \prod_{l=1}^{L-1} p(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}, \boldsymbol{\alpha}^{(l)}|\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}) \cdot p(\boldsymbol{h}^{(L)}, \boldsymbol{z}^{(L)}) \tag{19}$$

These components are respectively explained below.

First, distribution $p(\boldsymbol{x}|\boldsymbol{h}^{(1)}, \boldsymbol{z}^{(1)})$ is the same as in capsule RBM. The exponential family form of this distribution can be written as

$$p(\boldsymbol{x}|\boldsymbol{h}^{(1)}, \boldsymbol{z}^{(1)}) = \prod_{m=1}^{M} p\big(x_m|\eta(\hat{\boldsymbol{a}}_m)\big), \tag{20}$$

where M is the number of dimensions in the visible variable, $\hat{\boldsymbol{a}}_m = \{\hat{a}_m^{(1)}, \cdots, \hat{a}_m^{(R)}\}$ are the posterior natural parameters corresponding to sufficient statistics of x_m, $r =\in \{1, \cdots, R\}$, and R is the number of sufficient statistics. The natural parameter corresponding to the r-th sufficient statistic of vector $\boldsymbol{x}$ is calculated as

$$\hat{\boldsymbol{a}}^{(r)} = \boldsymbol{a}^{(r)} + \delta(r=1)\big(\sum_{k_1=1}^{K_1} z_{k_1}^{(1)} \boldsymbol{W}_{k_1}^{(1)} \boldsymbol{h}_{k_1}^{(1)}\big), \tag{21}$$

where with delta function $\delta(r = 1)$ we assume the first sufficient statistic is $\boldsymbol{x}$.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Second, the conditional distribution in an intermediate layer $p(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}, \boldsymbol{\alpha}^{(l)} | \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)})$ is actually an energy-based model:

$$p(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}, \boldsymbol{\alpha}^{(l)} | \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}) = \frac{1}{Z_l} e^{-E(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}, \boldsymbol{\alpha}^{(l)})}, \tag{22}$$

where Z_l is the partition function, and in the language of exponential family the energy function can be defined as

$$\begin{aligned} E(\boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}, \boldsymbol{\alpha}^{(l)}) = &- \sum_{k_1}^{K_l} \sum_{j=1}^{J} \left(\boldsymbol{b}_{k_1,j}^{(l)\,\mathrm{T}} \boldsymbol{t}_{k_1,j}^{(l)} + \log g(h_{k_1,j}^{(l)}) \right) \\ &- \boldsymbol{c}^{(l)\,\mathrm{T}} \boldsymbol{z}^{(l)} - \sum_{k_1=1}^{K_l} \boldsymbol{d}_{k_1}^{(l)\,\mathrm{T}} \boldsymbol{\alpha}_{k_1}^{(l)} \\ &- \sum_{k_1=1}^{K_l} \sum_{k_2=1}^{K_{l+1}} \alpha_{k_1,k_2}^{(l)} z_{k_1}^{(l)} z_{k_2}^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\,\mathrm{T}} \boldsymbol{W}_{k_1,k_2}^{(l+1)} \boldsymbol{h}_{k_2}^{(l+1)}, \end{aligned} \tag{23}$$

where J is the length of a capsule $\boldsymbol{h}_{k_1}^{(l)}$, $\boldsymbol{t}_{k_1}^{(l)}$ is the corresponding sufficient statistics, $\boldsymbol{b}_{k_1}^{(l)}$ is the corresponding natural parameter, $c^{(l)}$ is the natural parameter of Bernoulli vector $\boldsymbol{z}^{(l)}$, $\boldsymbol{d}_{k_1}^{(l)}$ is the natural parameter of Multinoulli distribution of vector $\boldsymbol{\alpha}_{k_1}^{(l)}$, and matrix $\boldsymbol{W}_{k_1,k_2}^{(l+1)}$ is the transformation matrix between capsule $C_{k_1}^{(l)}$ and $C_{k_2}^{(l+1)}$.

From this energy-based distribution, we can derive conditionals of each individual variable, as below:

$$\begin{aligned} &p(\boldsymbol{h}^{(l)} | \boldsymbol{z}^{(l)}, \boldsymbol{\alpha}^{(l)}, \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}) \\ &= \prod_{k_1=1}^{K_l} p(\boldsymbol{h}_{k_1}^{(1)} | z_{k_1}^{(l)}, \boldsymbol{\alpha}_{k_1}^{(l)}, \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}) \\ &= \prod_{k_1=1}^{K_l} p\big(\boldsymbol{h}_{k_1}^{(l)} | \eta(\hat{\boldsymbol{b}}_{k_1}^{(l)})\big), \end{aligned} \tag{24}$$

where

$$\hat{\boldsymbol{b}}_{k_1}^{(l,u)} = \boldsymbol{b}_{k_1}^{(l,u)} + \delta(u=1) \sum_{k_2=1}^{K_{l+1}} \alpha_{k_1,k_2}^{(1)} z_{k_1}^{(l)} z_{k_2}^{(l+1)} \boldsymbol{W}_{k_1,k_2}^{(l+1)} \boldsymbol{h}_{k_2}^{(l+1)}; \tag{25}$$

$$\begin{aligned} &p\big(\boldsymbol{z}^{(l)} | \boldsymbol{h}^{(l)}, \boldsymbol{\alpha}^{(l)}, \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}\big) \\ &= \prod_{k_1=1}^{K_l} p\big(z_{k_1}^{(l)} | \boldsymbol{h}_{k_1}^{(l)}, \boldsymbol{\alpha}_{k_1}^{(l)}, \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}\big) \\ &= \prod_{k_1=1}^{K_l} \mathcal{BE}\big(z_{k_1}^{(l)} | \eta(\hat{c}_{k_1}^{(l)})\big), \end{aligned} \tag{26}$$

where $z_{k_1}^{(l)}$ is a Bernoulli variable and its posterior natural parameter is

$$\hat{c}_{k_1}^{(l)} = c_{k_1}^{(l)} + \sum_{k_2=1}^{K_{l+1}} \alpha_{k_1,k_2}^{(1)} z_{k_2}^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\,\mathrm{T}} \boldsymbol{W}_{k_1,k_2}^{(l+1)} \boldsymbol{h}_{k_2}^{(l+1)}; \tag{27}$$

$$\begin{aligned} &p\big(\boldsymbol{\alpha}^{(l)} | \boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}, \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}\big) \\ &= \prod_{k_1=1}^{K_l} p\big(\boldsymbol{\alpha}_{k_1}^{(l)} | \boldsymbol{h}_{k_1}^{(l)}, z_{k_1}^{(l)}, \boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}\big) \\ &= \prod_{k_1=1}^{K_l} \mathcal{MU}\big(\boldsymbol{\alpha}_{k_1}^{(l)} | \eta(\hat{\boldsymbol{d}}_{k_1}^{(l)})\big), \end{aligned} \tag{28}$$

where $\boldsymbol{\alpha}_{k_1}^{(l)}$ is a Multinoulli variable of length K_{l+1} and its posterior parameter is

$$\hat{\boldsymbol{d}}_{k_1}^{(l)} = \boldsymbol{d}_{k_1}^{(l)} + \begin{bmatrix} z_{k_1}^{(l)} z_1^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\,\mathrm{T}} \boldsymbol{W}_{k_1,1}^{(l+1)} \boldsymbol{h}_1^{(l+1)} \\ z_{k_1}^{(l)} z_2^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\,\mathrm{T}} \boldsymbol{W}_{k_1,2}^{(l+1)} \boldsymbol{h}_2^{(l+1)} \\ \vdots \\ z_{k_1}^{(l)} z_{K_{l+1}}^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\,\mathrm{T}} \boldsymbol{W}_{k_1,K_{l+1}}^{(l+1)} \boldsymbol{h}_{K_{l+1}}^{(l+1)} \end{bmatrix}. \tag{29}$$

For the top hidden layer, $p(\boldsymbol{h}^{(L)}, \boldsymbol{z}^{(L)})$ can be simply factorized as

$$p(\boldsymbol{h}^{(L)}, \boldsymbol{z}^{(L)}) = p(\boldsymbol{z}) p(\boldsymbol{h} | \boldsymbol{z}), \tag{30}$$

where $p(\boldsymbol{z})$ follows a Multinoulli distribution, and $p(\boldsymbol{h}|\boldsymbol{z})$ follows a distribution from the exponential family.

Similarly, the joint distribution of the inference component is written as

$$\begin{aligned} q(\boldsymbol{h}, \boldsymbol{z}, \boldsymbol{\alpha} | \boldsymbol{x}) &= q(\boldsymbol{h}^{(1)}, \boldsymbol{z}^{(1)} | \boldsymbol{x}) \\ &\cdot \prod_{l=1}^{L-1} q(\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}, \boldsymbol{\alpha}^{(l)} | \boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}). \end{aligned} \tag{31}$$

First, $q(\boldsymbol{h}^{(1)}, \boldsymbol{z}^{(1)} | \boldsymbol{x})$ is the same as in capsule RBM.

For $1 \leq l \leq L-2$, $q(\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}, \boldsymbol{\alpha}^{(l)} | \boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)})$ is an energy-based submodel

$$q(\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}, \boldsymbol{\alpha}^{(l)} | \boldsymbol{h}^{(l)}, \boldsymbol{z}^{(l)}) = \frac{1}{Z_{l+1}^{(R)}} e^{-E(\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}, \boldsymbol{\alpha}^{(l)})}, \tag{32}$$

where

$$\begin{aligned} &E(\boldsymbol{h}^{(l+1)}, \boldsymbol{z}^{(l+1)}, \boldsymbol{\alpha}^{(l)}) \\ &= - \sum_{k_2=1}^{K_{l+1}} \sum_{j=1}^{J} \left(\boldsymbol{b}_{k_2,j}^{(R,l+1)\,\mathrm{T}} \boldsymbol{t}_{k_2,j}^{(l+1)} + \log g(h_{k_2,j}^{(l+1)}) \right) \\ &- \boldsymbol{c}^{(R,l+1)\,\mathrm{T}} \boldsymbol{z}^{(l+1)} \\ &- \sum_{k_1=1}^{K_l} \boldsymbol{d}_{k_1}^{(R,l)\,\mathrm{T}} \boldsymbol{\alpha}_{k_1}^{(l)} \\ &- \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_{l+1}} \alpha_{k_1,k_2}^{(l)} z_{k_1}^{(l)} z_{k_2}^{(l+1)} \boldsymbol{h}_{k_1}^{(l+1)\,\mathrm{T}} \boldsymbol{W}_{k_1,k_2}^{(R,l+1)} \boldsymbol{h}_{k_2}^{(l+1)}. \end{aligned} \tag{33}$$

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

The conditional distributions of this energy-based submodel can be obtained as below:

$$\begin{aligned} &q(\boldsymbol{h}^{(l+1)}|\boldsymbol{z}^{(l+1)},\boldsymbol{\alpha}^{(l)},\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}) \\ &= \prod_{k_2=1}^{K_{l+1}} q(\boldsymbol{h}_{k_2}^{(l+1)}|z_{k_2}^{(l+1)},\boldsymbol{\alpha}^{(l)},\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}) \\ &= \prod_{k_2=1}^{K_{l+1}} q\big(\boldsymbol{h}_{k_2}^{(l+1)}|\eta(\hat{\boldsymbol{b}}_{k_2}^{(R,l+1)})\big), \end{aligned} \tag{34}$$

where

$$\begin{aligned} \hat{\boldsymbol{b}}_{k_2}^{(R,l+1,u)} &= \boldsymbol{b}_{k_2}^{(R,l+1,u)} + \delta(u=1) \\ &\cdot \sum_{k_1=1}^{K_l} \alpha_{k_1,k_2}^{(l)} z_{k_1}^{(l)} z_{k_2}^{(l+1)} \boldsymbol{h}_{k_1}^{(l+1)\mathrm{T}} \boldsymbol{W}_{k_1,k_2}^{(R,l+1)} \boldsymbol{h}_{k_2}^{(l+1)}; \end{aligned} \tag{35}$$

$$\begin{aligned} &q(\boldsymbol{z}^{(l+1)}|\boldsymbol{h}^{(l+1)},\boldsymbol{\alpha}^{(l)},\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}) \\ &= \prod_{k_2=1}^{K_{l+1}} q\big(z_{k_2}^{(l+1)}|\boldsymbol{h}_{k_2}^{(l+1)},\boldsymbol{\alpha}^{(l)},\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}\big) \\ &= \prod_{k_2=1}^{K_{l+1}} \mathcal{BE}\big(z_{k_2}^{(l+1)}|\eta(\hat{c}_{k_2}^{(R,l+1)})\big), \end{aligned} \tag{36}$$

where

$$\hat{c}_{k_2}^{(R,l+1)} = c_{k_2}^{(R,l+1)} + \sum_{k_1=1}^{K_l} \alpha_{k_1,k_2}^{(l)} z_{k_1}^{(l)} \boldsymbol{h}_{k_1}^{(l+1)\mathrm{T}} \boldsymbol{W}_{k_1,k_2}^{(R,l+1)} \boldsymbol{h}_{k_2}^{(l+1)}; \tag{37}$$

$$\begin{aligned} &q(\boldsymbol{\alpha}^{(l)}|\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)},\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}) \\ &= \prod_{k_1=1}^{K_l} q(\boldsymbol{\alpha}_{k_1}^{(l)}|\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)},z_{k_1}^{(l)}) \\ &= \prod_{k_1=1}^{K_l} \mathcal{MU}\big(\eta(\hat{\boldsymbol{d}}_{k_1}^{(R,l)})\big), \end{aligned} \tag{38}$$

where

$$\hat{\boldsymbol{d}}_{k_1}^{(R,l)} = \boldsymbol{d}_{k_1}^{(R,l)} + \begin{bmatrix} z_{k_1}^{(l)} z_1^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\mathrm{T}} \boldsymbol{W}_{k_1,1}^{(R,l+1)} \boldsymbol{h}_1^{(l+1)} \\ z_{k_1}^{(l)} z_2^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\mathrm{T}} \boldsymbol{W}_{k_1,2}^{(R,l+1)} \boldsymbol{h}_2^{(l+1)} \\ \vdots \\ z_{k_1}^{(l)} z_{K_{l+1}}^{(l+1)} \boldsymbol{h}_{k_1}^{(l)\mathrm{T}} \boldsymbol{W}_{k_1,K_{l+1}}^{(R,l+1)} \boldsymbol{h}_{K_{l+1}}^{(l+1)} \end{bmatrix}. \tag{39}$$

We assume $\boldsymbol{z}^{(L)}$ is Multinoulli in $q(\boldsymbol{h}^{(L)},\boldsymbol{z}^{(L)},\boldsymbol{\alpha}^{(L-1)}|\boldsymbol{h}^{(L-1)},\boldsymbol{z}^{(L-1)})$.

A. Model Learning

Obviously, the generative parameters include $\theta = \{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},\boldsymbol{W}\}$. The recognition parameters include $\phi = \{\boldsymbol{b}^{(R)},\boldsymbol{c}^{(R)},\boldsymbol{d}^{(R)},\boldsymbol{W}^{(R)}\}$. Similar to the parameter learning in capsule HM, we can derive learning rules for capsule DGM:

$$\Delta_{\boldsymbol{\theta}} = \frac{\partial j(\boldsymbol{x})}{\partial \boldsymbol{\theta}} = -\mathrm{E}_{q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})}\Big[\frac{\partial \log p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})}{\partial \boldsymbol{\theta}}\Big]. \tag{40}$$

Now, let us start with the gradient for $\boldsymbol{W}_{k_l,k_{l+1}}^{(l+1)}$ below.

$$\begin{aligned} &\frac{\partial \log p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(l+1)}} = \frac{\partial \log p\big(\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)},\boldsymbol{\alpha}^{(l)}|\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)}\big)}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(l+1)}} \\ &= \frac{\partial E(\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)},\boldsymbol{\alpha}^{(l)})}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(l+1)}} \\ &\quad - \mathrm{E}_{p(\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)},\boldsymbol{\alpha}^{(l)}|\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)})}\Big[\frac{\partial E(\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)},\boldsymbol{\alpha}^{(l)})}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(l+1)}}\Big] \\ &= \big(\alpha_{k_l,k_{l+1}}^{(l)} z_{k_l}^{(l)} \boldsymbol{h}_{k_l}^{(l)} - \langle \alpha_{k_l,k_{l+1}}^{(l)} z_{k_l}^{(l)} \boldsymbol{h}_{k_l}^{(l)} \rangle\big)\big(z_{k_{l+1}}^{(l+1)} \boldsymbol{h}_{k_{l+1}}^{(l+1)}\big)^{\mathrm{T}}, \end{aligned} \tag{41}$$

where $\langle\cdot\rangle$ is the shorthand for expectation w.r.t. generative distribution $p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})$, and variables outside $\langle\cdot\rangle$ are sampled from the recognition distribution $q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})$ (actually they are mean-field approximations).

Similarly, the gradients for $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ can be computed as below

$$\frac{\partial \log p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})}{\partial \boldsymbol{b}_k^{(l,u)}} = \boldsymbol{t}_k^{(l,u)} - \langle \boldsymbol{t}_k^{(l,u)} \rangle, \tag{42}$$

$$\frac{\partial \log p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})}{\partial \boldsymbol{c}^{(l)}} = \boldsymbol{z}^{(l)} - \langle \boldsymbol{z}^{(l)} \rangle, \tag{43}$$

$$\frac{\partial \log p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})}{\partial \boldsymbol{d}_{k_l,k_{l+1}}^{(l)}} = \boldsymbol{\alpha}_{k_l,k_{l+1}}^{(l)} - \langle \boldsymbol{\alpha}_{k_l,k_{l+1}}^{(l)} \rangle. \tag{44}$$

Using the same spirit, we can compute gradients for recognition parameters using the master equation below.

$$\Delta_{\boldsymbol{\phi}} = \frac{\partial j(\boldsymbol{x})}{\partial \boldsymbol{\phi}} = -\mathrm{E}_{p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})}\Big[\frac{\partial \log q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})}{\partial \boldsymbol{\phi}}\Big]. \tag{45}$$

$$\begin{aligned} &\frac{\partial q\big(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x}\big)}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(R,l+1)}} = \frac{\partial q\big(\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)},\boldsymbol{\alpha}^{(l)}|\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)}\big)}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(R,l+1)}} \\ &= \frac{\partial E(\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)},\boldsymbol{\alpha}^{(l)})}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(R,l+1)}} \\ &\quad - \mathrm{E}_{q(\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)},\boldsymbol{\alpha}^{(l)}|\boldsymbol{h}^{(l)},\boldsymbol{z}^{(l)})}\Big[\frac{\partial E(\boldsymbol{h}^{(l+1)},\boldsymbol{z}^{(l+1)},\boldsymbol{\alpha}^{(l)})}{\partial \boldsymbol{W}_{k_l,k_{l+1}}^{(R,l+1)}}\Big] \end{aligned} \tag{46}$$

$$= \big(z_{k_l}^{(l)} \boldsymbol{h}_{k_l}^{(l)}\big)\big(\alpha_{k_l,k_{l+1}}^{(l)} z_{k_{l+1}}^{(l+1)} \boldsymbol{h}_{k_{l+1}}^{(l+1)} - \langle \alpha_{k_l,k_{l+1}}^{(l)} z_{k_{l+1}}^{(l+1)} \boldsymbol{h}_{k_{l+1}}^{(l+1)} \rangle\big)^{\mathrm{T}}. \tag{47}$$

Similarly, we can have

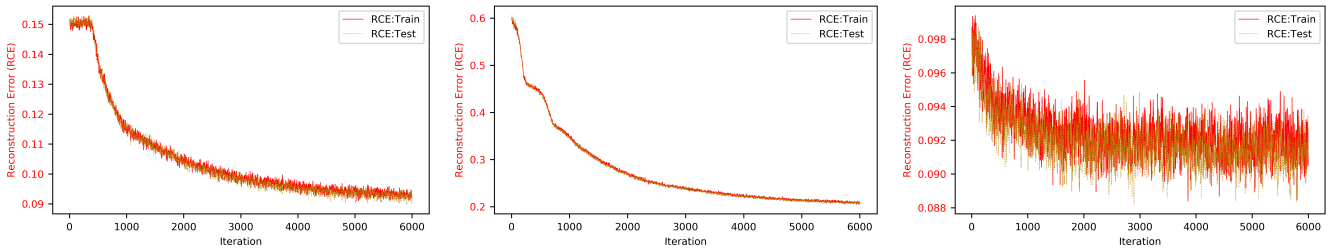
$$\frac{\partial \log q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})}{\partial \boldsymbol{b}_k^{(R,l,u)}} = \boldsymbol{t}_k^{(l,u)} - \langle \boldsymbol{t}_k^{(l,u)} \rangle, \tag{48}$$

$$\frac{\partial \log q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})}{\partial \boldsymbol{c}^{(R,l)}} = \boldsymbol{z}^{(l)} - \langle \boldsymbol{z}^{(l)} \rangle, \tag{49}$$

$$\frac{\partial \log q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})}{\partial \boldsymbol{d}_{k_l,k_{l+1}}^{(R,l)}} = \boldsymbol{\alpha}_{k_l,k_{l+1}}^{(l)} - \langle \boldsymbol{\alpha}_{k_l,k_{l+1}}^{(l)} \rangle, \tag{50}$$

where $\langle\cdot\rangle$ is the expectation under recognition distribution $q(\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha}|\boldsymbol{x})$, and variables outside are sampled (mean-field approximated) from the generative distribution $p(\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\boldsymbol{\alpha})$.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE



(a) Pretrain first capsule RBM. (b) Pretrain first capsule RBM. (c) Fine-tuning.

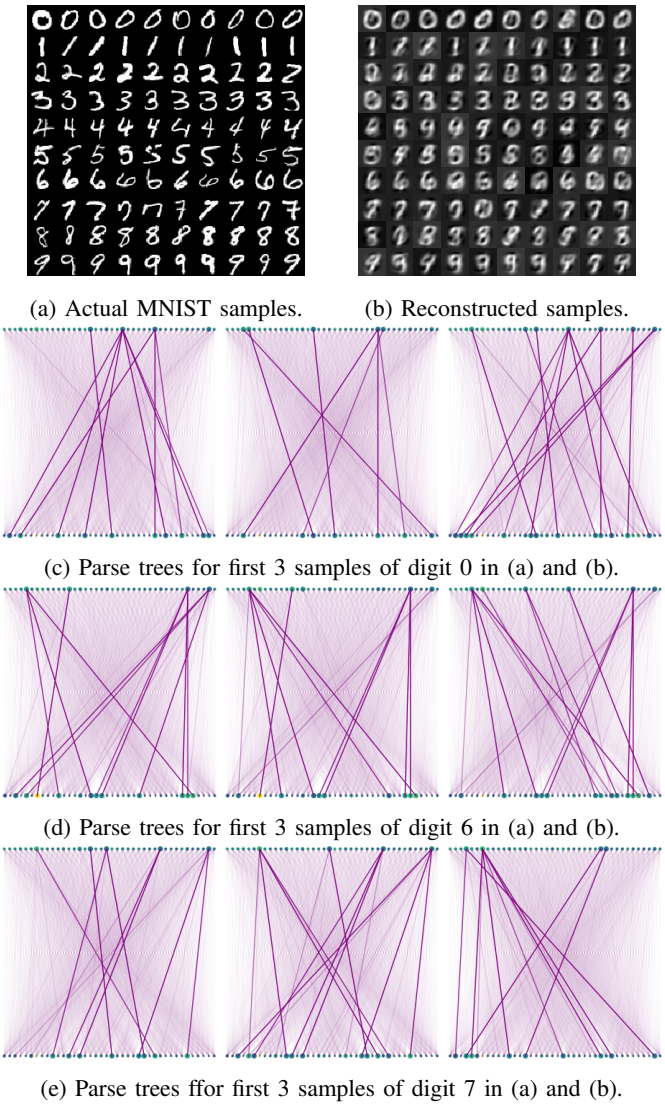
Fig. 7: Learning curves of cap-DGM in terms of reconstruction error in pretraining and finetuning phases on MNIST.

IV. Preliminary Experiments

We conducted a set of preliminary experiments on the MNIST and Fashion-MNIST data. First of all, we learned a cap-DGM, on both datasets, with two latent layers of capsules, each of which has 40 capsules and each capsule has 9 Gaussian variables. The network was layer-wise pretrained as cap-RBMs, and then finetuned using our wake-sleep learning algorithm. The pretraining phase is essential to help initialize the model parameters. The learning curves in terms of reconstruction error in both phases on MNIST are shown in Fig. 7. It shows that both training phases gradually reduce the reconstruction error. The learning of cap-RBMs does help guide the learning of the entire network into a comfort parameter space. In our experiments, we found that the learning of cap-DGM would not be successful without pretraining.

After model learning, we generated parse trees for real samples from the data and visualized it in Fig. 8 and 9. It can be seen that in each case, a tree structure indeed was constructed with the parent capsules on the top layer and child capsules at the bottom layer. Each child capsule is only connected to just one parent capsule, which is an advantage over the activity spectra of capsules from cap-HM reported in [11]. Further more, from these parse trees, we can observe that different samples from the same class obtain similar but not identical structures, and parse trees across different classes exhibit distinct patterns. Thus, it contributes a novel way of visualizing the latent space in deep generative models.

Due to the property that the structure of a tree expands exponentially as the increase of depth, the parse tree can not be very deep. But by adding one more latent capsule layers, we can observe, e.g. from Fig. 10, parse trees more clearly for corresponding samples. Again, in contrast to the results reported in [11], it implies that our new model helps enforce more structured latent space than cap-HM (providing spectrum-like structure) and other deep generative models such as VAE [16] (providing full connections between layers).



(a) Actual MNIST samples. (b) Reconstructed samples.

(c) Parse trees for first 3 samples of digit 0 in (a) and (b).

(d) Parse trees for first 3 samples of digit 6 in (a) and (b).

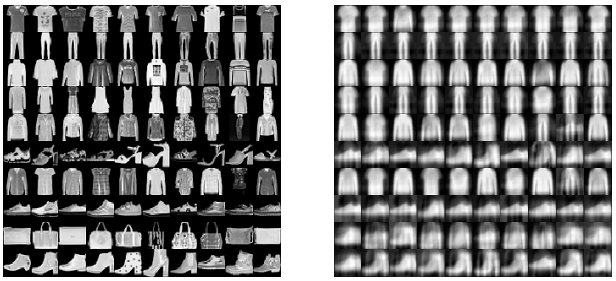
(e) Parse trees ffor first 3 samples of digit 7 in (a) and (b).

Fig. 8: Reconstructed MNIST samples and corresponding parse trees from a cap-DGM with two latent capsule layers. Visible layer is not visualized in the trees. Size of a node indicates the activity of the capsule.

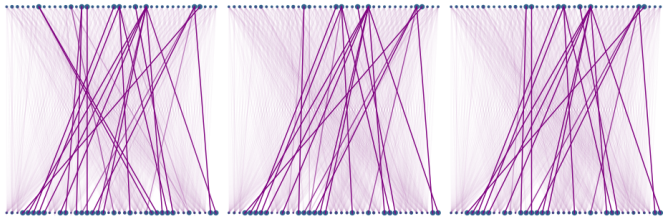
V. Discussion and Conclusion

It is a challenging task to model hierarchically structured latent space in deep generative models. In this paper, we present a novel capsule deep generative model, from the perspective of exponential family, that can form parse trees in the latent space to represent part-whole relationship. Our preliminary experimental results show that such tree structures

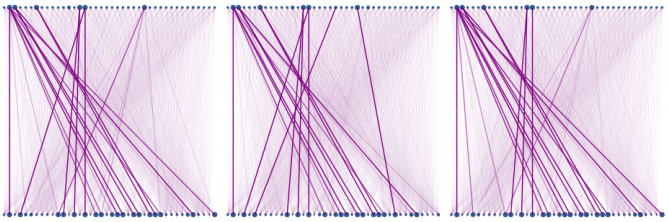
978-1-7281-6926-2/20/$31.00 ©2020 IEEE



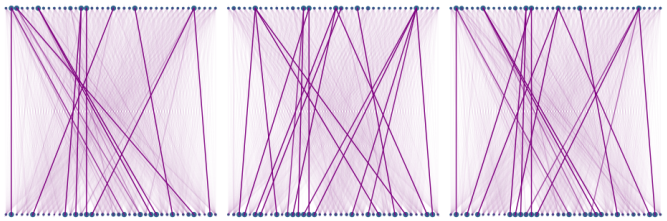
(a) Actual Fashion-MNIST samples. (b) Reconstructed samples.



(c) Parse trees for first three trousers in (a) and (b).



(d) Parse tree for first three sneakers in (a) and (b).



(e) Parse tree for first three ankle boots in (a) and (b).

Fig. 9: Reconstructed Fashion-MNIST samples and corresponding parse trees from a cap-DGM with two latent capsule layers. Visible layer is not visualized in the trees. Size of a node indicates the activity of the capsule.

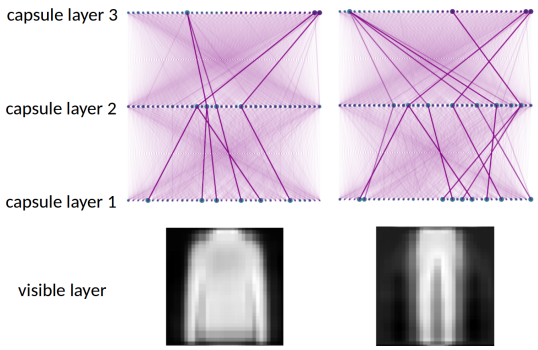


Fig. 10: Parse trees of corresponding Fashion-MNIST samples from a cap-DGM of three latent layers.

can be automatically learned from the data using our model. Please be reminded that, this work is a position paper, proving the concept, but needing more studies and improvements in implementation, learning algorithms, and validations. We derived the wake-sleep learning algorithm for our model. To improve learning quality and possibly eliminate the pretraining phase, new learning algorithms (such as REINFORCE method [17] and back-propagation [16], [18]) will be explored. The method is currently implemented using pure Python. It will be re-designed in PyTorch or TensorFlow so that components such as convolution can be added to the structure for better quality of generation.

Acknowledgment

The authors would like to thank the anonymous reviewers and all organisers of the conference.

References

[1] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.
[2] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.
[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
[4] G. Hinton, J. McClelland, and D. Rumelhart, "Distributed representations," in *Parallel Distributed Processing: Explorations in The Microstructure of Cognition*, D. Rumelhart and J. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 77–109.
[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 2, pp. 1137–1155, 2003.
[6] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference on Learning Representations*, 2013.
[7] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 255–258.
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
[9] S. Sabour, N. Frosst, and G. Hinton, "Dynamic routing between capsules," in *Neural Information Processing Systems*, 2017, pp. 3856–3866.
[10] G. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *International Conference on Learning Representations*, 2018.
[11] Y. Li and X. Zhu, "Capsule generative models," in *International Conference on Artificial Neural Networks*, Munich, Germany, Sep. 2019, pp. 281–295.
[12] G. Hinton, A. Krizhevsky, and S. Wang, "Transforming auto-encoder," in *International Conference on Artificial Neural Networks*, 2011, pp. 44–51.
[13] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Advances in Neural Information Processing Systems*, 2005, pp. 1481–1488.
[14] Y. Li and X. Zhu, "Exponential family restricted Boltzmann machines and annealed importance sampling," in *International Joint Conference on Neural Networks*, 2018, pp. 39–48.
[15] ——, "Exploring Helmholtz machine and deep belief net in the exponential family perspective," in *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.
[16] D. Kingma and M. Welling, "Auto-encoding variational Bayes," in *International Conference on Learning Representations*, 2014.
[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.
[18] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International Conference on Machine Learning*, 2014, pp. II–1278–II–1286.