Discrete Memory Addressing Variational Autoencoder for Visual Concept Learning

1st Yanze Min
dept. computer science and technology
Tsinghua University
Beijing, China
minyz16@mails.tsinghua.edu.cn

2nd Hang Su
dept. computer science and technology
Tsinghua University
Beijing, China
suhangss@mail.tsinghua.edu.cn

2nd Jun Zhu
dept. computer science and technology
Tsinghua University
Beijing, China
dcszj@mail.tsinghua.edu.cn

2nd Bo Zhang
dept. computer science and technology
Tsinghua University
Beijing, China
dcszb@mail.tsinghua.edu.cn

***Abstract*—A substantial aspect of general intelligence is the ability to summarize basic building blocks from various high-level concepts. Artificial vision systems with such hierarchical property can not only perform accurate reasoning for complex observations, but also learn useful low-level knowledge shared across scenes. To achieve this goal, we propose a discrete memory addressing VAE model (DM-VAE) for explicitly memorizing and reasoning about shared primitives in images. A time-persistence memory module is used to store the learned abstract knowledge and to interact with the generative model. The model decides what to pay attention to at each step, and constructs the primitive library automatically as the learning progresses in a fully unsupervised setting. While performing inference, the model attempts to interpret a new observation as a combination of previously learned elements. We further derive a proper variational lower bound which can be optimized efficiently. We conduct visual comprehension experiments on images and demonstrate that our model is able to search, identify, and memorize semantically meaningful primitive concepts.**

***Index Terms*—deep generative model, hierarchical Bayesian model, concept learning, deep model with memory**

I. Introduction

Human can effortlessly discover and utilize low-level concepts which are shared among higher hierarchies. Especially in visual perception, we can quickly decompose an observation with simpler building blocks we have been familiar with, identify their properties, and can then focus on learning and memorizing the remaining novel part. We summarize these fundamental concepts from everyday experience in an unsupervised fashion and build complex concepts hierarchically upon simpler ones. This is considered as a fundamental cognitive ability that allows us to understand and interact with the world efficiently and robustly [1]. Some researchers suppose this ability is at the core of fields where people still easily beat machines like continual learning or few-shot learning [2]. Artificial systems with this ability are expected to solve tasks which need in-depth vision comprehension, including scene decomposition, multi-agent control, etc.

Despite remarkable progress in deep generative models [3]–[5], this kind of human conceptual behavior has usually eluded their representation power. Compared to the hierarchical generative process of human, most of these models attempt to map directly from a random noise vector z into an image x with a neural network $p(x|z)$. This non-hierarchical generative process prevents the discovery of useful hidden structures in original observations such as shared primitives or spatial relations. On the other hand, even if the model may have learned some structures implicitly, these knowledge are all stored as parameters of neural networks, which has little semantic meaning and is hard to deal with. The central challenge to design a hierarchical generative model has three aspects: 1) *primitive*: in what form the learned concepts is to store in the model; 2) *variability*: how to describe the uncertainty which naturally occurs in primitive realization; 3) *inference*: how to efficiently parse a new observation with learned concepts in a (probably huge) compositional space.

To design such a generative model with hierarchical structure, some methods have been proposed. Compositional grammar models [2], [6] use a separated procedure to construct a fixed primitive library first, and then apply a probabilistic generative process to depict the variability of primitive realization and combination. However, since these models typically learn representation directly on the raw image space, plenty of domain-specific assumptions and special handcrafted care are often needed. Inference in these models (e.g., via MCMC sampling) can be extremely slow since the original primitive combination space is huge. On the other hand, hierarchical generative methods have largely been incompatible with deep learning. Instance segmentation models like mask-RCNN [7] are relevant to decomposing a scene into basic elements, yet these methods are not generative and need heavily annotated data. Sequential generative models including AIR [8] and MONet [9] use a sequential variational auto-encoder that attempts to generate one object from a latent code per time-step. Despite the success, these methods do not model the

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

hierarchy explicitly. Low-level time invariant knowledge is not preserved well, and information of different level can be tangled in the latent space. Hence, there is no straight forward way for the model to summarize and reuse those basic elements.

In this paper, a novel hierarchical generative model (DM-VAE) is proposed, which attempts to bridge the gap between traditional hierarchical models and deep methods. Equipped with a discrete addressing memory buffer, DM-VAE can actively allocate memory space for shared low-level knowledge. DM-VAE is capable of distinguishing the universal information (i.e., types of shared primitives) from the individual information (i.e., randomness that occurs when rendering a primitive). DM-VAE reads/writes universal information from/into the time-persistence memory buffer to lay stress on the invariability. As for individual information, DM-VAE models their randomness with random variables. The model comprises three major modules: (1) a memory buffer which serves as the primitive library to store learned deep features, (2) a generator interacting with the memory buffer while producing realized primitives, and (3) a spatial attention model implemented as a recurrent neural network to attend to regions of interest at every time step. These three modules are combined in a fully-differentiable manner, and the model in its mathematical form is a variant of the sequential variational auto-encoder. The complete learning and inference procedure is fast, amortized, and unsupervised via maximizing evidence lower bound (**ELBO**). It is worth pointing out that the primitive library in our model is jointly learned as the training progresses, unlike the fully pre-defined way. In our experiments, we show that our model can obtain semantically interpretable primitives and relations parsed from the original data, while achieving similar or better representational power with the state-of-the-art generative models which focus on object-wise inference, such as the Attend-Infer-Repeat (AIR) framework [8].

The main contributions of the paper are as follows. First, we formulate a hierarchical deep generative model and show how the deep model can interact with the lower-level primitive library appropriately. The second contribution is that we design an efficient variational inference procedure which can make the whole system trained end-to-end without supervision. Finally, we demonstrate empirically how our model allows for constructing the library automatically, and for decomposing images into their underlying structure.

II. Related Work

The concept of understanding the latent structure of an image has a long history. The inverse graphics line of work consists of a function to go from compact descriptions of scenes called the graphics code to images. This graphics code is essential for image comprehension since it mostly captures variables in the real image generating process. Works of inverse graphics such as [10]–[12] follow the framework of defining a probabilistic or deterministic model with latent random variables or parameters, then using an inference or optimization algorithm to find the most appropriate set of latent parameters given the observations. Methods in this line are not necessarily generative and often need assumptions which tend to be too strong for the model to be broadly useful.

Compositional grammar models like and-or graph (AOG) and Bayesian program learning (BPL) [2], [6], [13] typically define a primitive set and a probabilistic generative language for generating the observation. They then search the program space and find a suitable generative program for the observed image. Despite the success in specific scenarios such as few-shot learning, these models do not leverage the expressive power of deep neural networks, so they often need plenty of special hand-crafted care. The inference algorithms for these methods are also especially hard to design and slow in practice.

Recently, deep generative models [3]–[5] have proven successful in generating realistic data from a latent code. These models draw support from the powerful neural networks to parameterize distributions defined in a generative process and thus they can provide powerful representation ability. By plugging in a structured prior, models such as [8], [9], [14], [15] use deep generative models to either infer the underlying structure, or generate images. These models typically follow the encoder-decoder framework generalized from variational auto-encoders. After a proper joint distribution is defined, the training and inference procedures can be left to the optimization of the variational lower bound. Therefore, these methods do not need extensive effort to design a bottom-up inference procedure as traditional compositional models. Though inspiring, the hierarchical nature that occurs in most concepts is not modeled in these methods explicitly. We make a step further to show how the generative model can interact with preserved low-level primitives.

Instead of inducing a structured generative process in deep generative models to get a structured representation, there is also a series of works named disentanglement. These methods attempt to endow different dimensions of latent variables with different semantic meanings, both supervised by attribute label or unsupervised. Kulkarni et al [16] propose a training procedure that produces a representation in the auto-encoder, in which lighting and pose variables are disentangled. Higgins et al [17] propose a modification of the variational lower-bound in encouraging disentangled representations. Chen et al [18] use the mutual information criterion for controlling the information flow from observed data to the with-semantic part of the latent variable. These works are more focus on the interpretability of the latent code, not on designing an interpretable architecture itself.

The use of a memory block in deep neural networks also gains much popularity recently. In this family of models including [19], [20], a memory buffer is allowed to read and write and is persistent in time. This memory buffer is often used to preserve valuable information which has some invariability with regard to proceeding operation steps. There are typically two kinds of read/write mechanism, soft v.s. hard read/write. The soft operation draws a proper distribution from all the memory buffers and outputs some kind of average

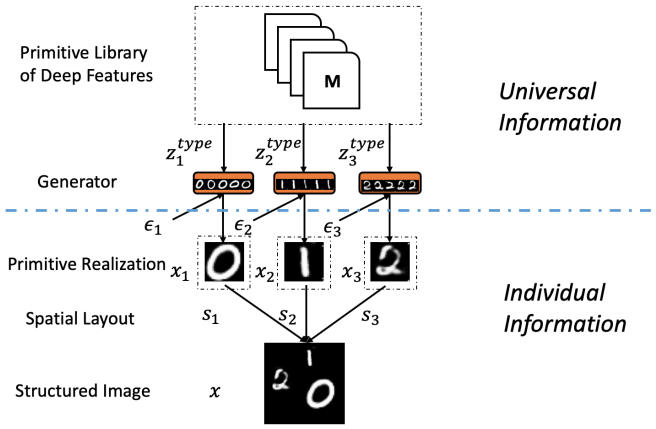


Fig. 1. The generative process of DM-VAE. First sampling the memory address $j = z_i^{type}$ and retrieving the content m_j. Then the noise vector ϵ_i is fed into the conditioned generator to produce x_i. Several primitive realizations are transformed into correct global scale and locations by transformation variables s. The universal information is written in M as the learning progresses.

embeddings of these slots. This is much more similar to the attention technique as used in [21]. The hard addressing mechanic follows some kind of discrete distribution and outputs one slot content at a time. Despite the extensive use in sequence learning and reinforcement learning, only little works including variational memory addressing [22] attempt to explore connections between the deep generative model and the memory network.

III. Method

We now describe the proposed DM-VAE model, illustrated in Figure 1. First, we shall introduce some notations, and then we will describe the full generative process in details in the next two subsections. We treat an observation x as a combination of basic components $\{x_i\}$. $M = \{m_i\}_{i=1,\cdots,n}$ denotes the memory buffer, where each $m_i \in R^d$ is a memory slot that stores general features of a primitive. A generator G conditioned on M takes noisy vector ϵ_i as input. When G is conditioned on a specific slot m_j, it is expected to produce the j^{th} type of primitive, so we call this a conditioned generator g_{m_j}. G can be implemented as a bank of generators or simply a generator with different m_js as a part of its input, which is our choice in practice. Each conditioned generator can be formulated as $g_{m_j}(\epsilon_i)$, and outputs a realized primitive x_i. The index j is chosen by the corresponding discrete random variable z_i^{type}, which serves as the memory address. Finally, spatial layout random variables $S = \{s_i\}$ is used to model the spatial relation among these primitives.

A. Generative Model for Primitives

At each time step i, the primitive generative process (see Fig.1) proceeds by first sampling the discrete memory address z_i^{type} from the discrete prior $p(z^{type})$, retrieving the content $m_j = m_{z_i^{type}}$ and using this universal information to guide the conditioned generator g_{m_j} of G. Then we sample a continuous noise vector ϵ_i from the prior distribution $p(\epsilon|m_j)$, and feed the noise vector into the conditioned generator to produce $g_{m_j}(\epsilon_i) \sim p(.|\epsilon, m_{z^{type}})$, i.e. the realized primitive x_i.

The intuition behind the design is that for a particular pattern that appears multiple times, we wish the model to memorize it explicitly, and recognize it as a learned type the next time when the model sees it. Therefore, unlike most previous methods which directly feed the generator a noise vector, we guide the generator with a huge amount of information from the memory buffer, which should act as a "template" of the desired output. To explain the remaining "distortion" of an observation given a correct type, the noise generative model $p(\epsilon|m_{z^{type}})$ is learned. Note that the distribution is conditioned on the type, so for each type, the noise distribution is different. This is reasonable because the distortion for different templates may not act as the same. In practice, both the generators g_{m_j} and the prior of the noise vector $p(\epsilon|m_{z^{type}})$ are parameterized by deep neural networks to capture the complex dependencies within them.

The distribution for the primitive x_i has the form

$$p(x_i|M) = \sum_{z_i^{type}} \int_{\epsilon_i} p(z_i^{type}) p(\epsilon_i|m_{z_i^{type}}) p(x_i|\epsilon_i, m_{z_i^{type}}) d\epsilon_i. \tag{1}$$

B. Image Generation Process

After sampling n primitives, the whole image x is generated by sampling global locations for x_is, distorting by scale, and adding them up. The image can be generated by:

$$x = \sum_{i=1}^{n} T_{s_i}(x_i), \tag{2}$$

where T_{s_i} is a spatial transformation for primitive x_i, parameterized by the transformation random variable s_i. For example, in the common 2D attention model which only allow degrees of freedom on location and scale, s is a four-dimensional vector that uses 2 dimension to describe the x and y coordinates, and another 2 dimension to model its scale. We use the differentiable spatial transformer architecture [23] to achieve the transformation T_s.

The distribution for image x is

$$p(x|M) = \sum_{n=1}^{N} p_N(n) \prod_{i=1}^{n} \int_{s_i} p(x_i|M) p(s_i|M) ds_i, \tag{3}$$

where $p_N(n)$ is a prior for the number of primitives n in the image with maximum value N, $p(x_i|M)$ is defined in Eqn.1, and $p(s_i|M)$ is a suitable prior (e.g., a Gaussian distribution) for transformation parameters. In practice, we adopt the similar trick as AIR did to parameterize n as a variable length binary latent vector z^{pres} that, for a given value n, z^{pres} is a vector formed of n ones followed by one zero. As we will see in the following section, this will simplify the inference procedure.

IV. Training and Inference

Inference for most models in the form of Eqns.1 and 3 is intractable due to the intractable integral or sum operation. Such models usually use an approximation inference technique called variational inference. In the most general setting where x denotes the observed data and z denotes the latent random variable, this kind of algorithms approximate the true posterior $p_\theta(z|x)$ with a parameterized $q_\phi(z)$ distribution, which is often implemented as a deep network. The whole system is then trained by maximizing the evidence lower bound (**ELBO**), and a good inference approximator $q_\phi(z)$ is obtained in the meanwhile.

However, the specific form of our model remains challenging even in the variational inference framework. The main challenges are: *1)Discreteness.* The discrete nature of the hard read/write mechanism of M (performed by the type variable z_{type}) makes the original problem a combinatorial optimization one which is typically hard to optimize. *2)Coupling.* The variable ϵ, the memory slot chosen by z_{type}, and parameters of the generator $g(.)$ are coupled together to produce a primitive in the image space, so it is difficult to credit only the universal information to z^{type} while keeping the individual information to ϵ. *3)Trans-dimensionality:* The number of primitives of a scene is a random variable itself. There is a trade-off that, the model must choose whether to cut up the image into smaller pieces to get a better explanation, or to take some parts as a whole for the overall model simplicity. We now describe the proposed variational inference procedure.

A. The ELBO Objective with Memory

We can write the evidence lower bound for Eqn.3:

$$\log p(x) \geq \mathbb{E}_{q_\phi}[\log p_\theta(x, z^{type}, z^{pres}, \epsilon, s) - \log q_\phi(z^{type}, z^{pres}, \epsilon, s|x)]. \quad (4)$$

Here we have omitted the condition of M in each term for clearness. The variational posterior is factorized as:

$$q_\phi(z^{type}, z^{pres}, \epsilon, s|x) = q_\phi(z^{pres}_{n+1} = 0|z_{1:n}, x) * \prod_{i=1}^{n} Q_i \quad (5)$$

$$Q_i = q_\phi(z_i^{pres} = 1|x, z_{1:i-1}) * q_\phi(s_i, \epsilon_i, z_i^{type}|z_i^{pres} = 1, x, z_{1:i-1}). \quad (6)$$

Let $z_{1:n}$ denote the set of all latent random variables $\{z_i^{pres}, z_i^{type}, \epsilon_i, s_i\}$ from step 1 to n, and z_i denote the set of variables at step i. Note that we have omitted the edge case of $z_1^{pres} = 0$, which means a blank image, in the above equations for clearness. We shall now have a detailed discussion about the variational posterior Eqn.5. First of all, to full-fill the need of the variation of the primitive number caused by z^{pres}, we implement the inference network $q(.)$ as a recurrent neural network, where $z_i^{pres} = 0$ is used as a terminate flag (corresponding to the first term in the RHS of Eqn.5). If not terminated, the RNN's output at step i is used to compute the second term in the RHS of Eqn.6. The second thing to notice is the factorization of the term $q(s_i, \epsilon_i, z_i^{type}|z_i^{pres} = 1, x, z_{1:i-1})$. Under the law of conditional probability, the three random variables of this term can be factorized in random order. However, reasonable order of these variables should follow the intuition that we first locate an object, then classify its general attribute and finally consider its specialty, i.e.

$$q_\phi(s_i, \epsilon_i, z_i^{type}|z_i^{pres}, x, z_{1:i-1}) = q_\phi(s_i|z_i^{pres}, x, z_{1:i-1}) \\ q_\phi(z_i^{type}|z_i^{pres}, x, z_{1:i-1}, s_i) q_\phi(\epsilon_i|z_i^{pres}, x, z_{1:i-1}, s_i, z_i^{type}) \quad (7)$$

We parameterize the first distribution in the RHS of the above equation, i.e. variational distribution of transformation variables s, simply as a Gaussian with its mean and variance computed by the RNN. The second term of z^{type} and the third term of ϵ needs to pay more attention. Remember that the discrete variable z^{type} serves as the address of the memory buffer, we follow the design pattern also used by [22] and [24] which computes the similarity between embeddings of inputs and each memory block, and then outputs the parameters controlling the discrete posterior distribution. To be more clear, the distribution $q(z_i^{type}|z_i^{pres}, x, z_{1:i-1}, s_i) \propto f_{sim}(m_{z_i^{type}}, x_i^{crop})$, where x_i^{crop} is ith patch cropped from x using s_i produced by the first term. The similarity function is implemented using a small MLP. The third term $q(\epsilon_i|z_i^{pres}, x, z_{1:i-1}, s_i, z_i^{type})$ is parameterized as a Gaussian distribution whose mean and covariance matrix is computed by a MLP using both the input patch x_i^{crop} and the selected memory slot $m_{z_i^{type}}$ as input.

The whole inference procedure is illustrated in Fig.2. Fig.2 *Left* explains the computation graph of the inference procedure. First the model decides whether there is a part remains to be explained, attends to the region of interest s_i, crops a part x_i^{crop} and tries to memorize its template index z_i^{type}. Then it explains the remaining error of x_i^{crop} through noise ϵ_i. Fig.2 *Right* shows the graphical model coinciding with Eqn.1 and Eqn.5.

B. Constrained Information Flow

Although the generative process designed above makes the memory slot m_i a template for all distorted primitives of a same semantic type, there is no explicit guarantee that the model properly assigns one object embedding accurately into one slot. In our earlier experiments, we find that the model may induce a flat distribution $q(m_{z^{type}})$ over memory slots while still gets a good reconstruction.

In fact, if z^{type}s actually induce a partition of the whole semantic space, the variable ϵ should "span" each sub-space. Let $x = F(z^{type}, \epsilon)$ denote a primitive realization in the image space, the model is expected to infer z^{type} solely after seeing x. This can be full-filled by maximizing the mutual information between x and z^{type}. Making use of the variational information maximization technique [25], we constrain the information flow towards z^{type} with the follow objective:

$$I(X, Z^{type}) \geq \mathbb{E}_{x \sim g_{z^{type}}}[\log q(z_i^{type}|x)], \quad (8)$$

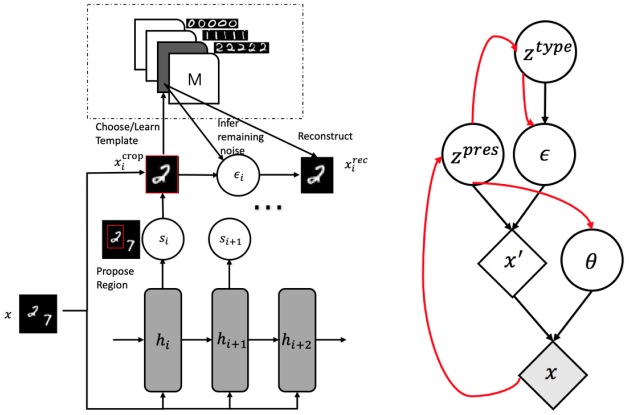


Fig. 2. *Left:*The inference process. We implement the inference network $q(.)$ as an RNN which outputs the inferred latent variables one at a time. At each time step we first locate an object, then infer its general attribute and finally consider its specialty. *Right:* The graphical model. Black lines correspond to the generation, while red lines the inference.

where capital letters denote random variables and samples are denoted using the lowercase. Also note that the x in the above equation represents an image patch selected by the model rather than the whole image. A good news is that the variational posterior $q(.|x)$ in Eqn.8 can be obtained immediately using the posterior of z^{type} in Eqn.7, except that the input x is now the synthetic data from generator bank rather than the true input data. We add this objective in the lower bound and update all $g_i \in G$ in each training step instead of sampling them. That means we sample images from every $g_{z_i^{type}}$, feed the synthetic data back to the variational network $q(z_i^{type}|.)$, and try to recover z_i^{type}. In experiments, we find this objective helps make a clearer separation of the semantic space.

C. Gradients for Training

The vanilla VAE uses the reparameterization trick to deal with the stochastic node arises from the probabilistic model in the computation graph. It is not our case because the discrete variables in the model fail to have a differentiable reparametrization. Maddison et al [26] propose a method that uses softmax instead of argmax operation in a function of Gumbel noise samples, which is the reparameterization term of discrete distribution. So the Gumbel-Softmax trick can be seen as a generalized form of reparametrization trick with discrete random variables. Here we show briefly how to use reparameterization trick for efficiently training and propagate gradients.

We essentially optimize the lower bound of the data log-likelihood as well as the mutual information:

$$\begin{aligned}\nabla_{\theta,\phi}\mathscr{L} = \nabla_{\theta,\phi}\mathbb{E}_{q_\phi(.|x)}[\log p_\theta(x, z^{type}, z^{pres}, \epsilon, s) \\ - \log q_\phi(z^{type}, z^{pres}, \epsilon, s|x)] \\ + \gamma\nabla_{\theta,\phi}\mathbb{E}_{x^{syn}\sim g_{z^{type}}}[\log q_\phi(z_i^{type}|x^{syn})], \quad (9)\end{aligned}$$

where θ denotes all the parameters in the generative process, and ϕ the inference process. γ is a hyper-parameter.

Now we check the random variables on which the inner term takes expectation. For continues variables ϵ and s we use the standard path-wise estimator $\epsilon = \mu_\epsilon + \sigma_\epsilon\xi_\epsilon$ and $\theta = \mu_\theta + \sigma_\theta\xi_\theta$ where ξs are standard Gaussian noises. For discrete variables z^{type} and z^{pres} we use the concrete distribution to reparameterize which means $z^{type} = Softmax((\log\alpha_{type} + G_{type})/T_{type})$. Let k denotes the number of memory slots. α is the parameter of the k dimensional discrete distribution, $P(z^{type} = k) \propto \alpha_k$. G_{type} is a k dimensional independent Gumble noise. T_{type} is a hyper-parameter. z^{pres} is treated similarly. For using the trick, the prior distribution for discrete variables are replaced by the corresponding Concrete distribution. For more details please refer to the paper [26] and [27]. Finally, since the synthetic image random variable x^{syn} is the output of the deterministic function of random variable ϵ and z^{type}, it can also be reparameterized first by ϵ and z^{type}, then by ξ_ϵ and G_{type}. The training procedure can then be done using the standard back-propagation after sampling all these random noises for reparameterization.

V. Experiments

A. Experiment Setup

Datasets: We apply the proposed DM-VAE to two datasets, multi-MNIST digits and CASIA offline handwritten characters [28]. Following the experiment settings in AIR [8], the Multi-MNIST dataset we use contains 50x50 0,1,or 2 MNIST digits and each number of digits have 20000 images.

Then, we apply DM-VAE to the more sophisticated dataset CASIA of handwritten Chinese characters. The full dataset contains 3000 classes of the most frequently used Chinese characters, each with 300 samples. A Chinese character is a recursive structured data composed with lower-level, simpler stroke primitives called radicals. We select a subset from the complete dataset of 50 types of characters which share a more compact basic radical set of about 30 radicals, and each character has 300 samples.

Model Architecture: Unless otherwise specified, we use the following architecture and hyper-parameters. We use a 256 dimension LSTM unit as the basic recurrent unit in the variation network. The similarity function $f_{sim}(m_i, x^{crop})$ between a slot m_i and the attended input x^{crop} is the cosine distance between $h_m(m_i)$ and $h_x(x^{crop})$, where the hs are MLPs with 2 hidden layers of size (256, 128). The latent variable ϵ is a 32 dimension Gaussian variable. All the encoders $q(.)$s and decoders $p(.)$s for latent variables is 2-layer MLPs with 256 and 128 hidden units with ReLU non-linearities. We train the model by optimizing with Eqn.9, with batch size equals to 64. Note that we keep the network architecture simple on purpose to justify the model effectiveness. As for the AIR model to compare, we do our best to implement it for ensuring the similar model complexity as ours. The attention RNN of AIR is also a 256 dimension LSTM network, and the encoder and decoder networks in most situation are 2-layer MLPs with the same size as ours.

B. Results on Multi-MNIST Digits

We keep 2000 images as test data. In DM-VAE, the number of memory slot is set to 15, and memory buffer is initialized using random noise from a uniform distribution. We train the model 300 epochs using the Adam optimizer with a learning rate of 1e-4. Deep generative models with discrete random variables are known for being difficult to train. Since our model chooses a discrete addressing mechanism, the temperature hyper-parameter T for the posterior concrete distribution of $q(z^{type})$, which controls the "sharpness" of the distribution, is essential for the training stability. We show training curves in Fig.3 *Left* under different Ts. In a typical learning procedure, there is first a period when the model memorizes nothing, and when it only starts to learn where to attend to. The noisy attention location can provide the memory buffer with nothing useful at this time, so the training loss maintains high and steady. After the attention mechanism finishes warming-up itself, the memory and the attention mechanism can help each other converge, which results in a fast loss drop in Fig.3 *Left*.

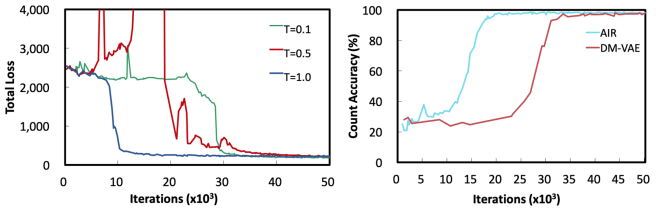


Fig. 3. *Left:* Training curves on Multi-MNIST dataset. *Right:* Count accuracy on test set.

To show that DM-VAE is able to conduct accurate object detection and reconstruction, the reconstruction experiment of test set images is illustrated in Fig.4. For each pair, left is an image from the test set and right is the reconstruction by our model. The colored bounding box is painted using the inferred transformation parameter s, for showing how the model constructs the overall image from primitives step by step. It is worth emphasizing that, besides getting a good reconstruction of structured images, the latent variables inferred by DM-VAE including z^{pres}, z^{type}, and s have obvious semantic meaning. We already use variable s to full-fill the object detection task in Fig.4, and this result is comparable to AIR (refer to Fig.3 in the AIR paper). z^{pres} can be used to infer the number of digits in test set images. Shown in Fig.3 *Right*, both models achieve almost 100% accuracy after convergence. For the quantitative result of model performance, we report the test set negative log-likelihood averaged by 10 training processes of each model in Tbl.I. DAIR is a modified version of AIR which employs a slightly different recurrent architecture for the inference network (see [8] for details). To compare the complexity of models, we also report the number of parameters for each model in the table. As we can see, the extra space overhead introduced by the DM-VAE model is reasonable, in particular, the space occupied by the memory buffer itself is negligible, meaning that the storage unit is highly scalable.

TABLE I
NLLs ON MULTI-MNIST AND ON CASIA OFFLINE CHARACTER DATASET. NUMBERS IN PARENTHESES INDICATE THE PARAMETER NUMBERS OF THE MEMORY BUFFER.

	Multi-MNIST	CASIA-CHAR
AIR	185.1±6.3	1310.6±28.8
DAIR	189.5±3.9	1302.4±19.4
DM-VAE (mem=15)	154.0±12.1	1277.9±26.1
DM-VAE (mem=50)	154.2±11.2	1228.3±33.3

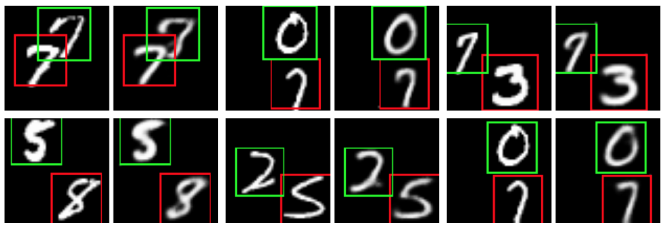


Fig. 4. Reconstruction of the test image. For each pair, left is an image from the test set and right is the reconstruction by our model. *Top Row:* The full DM-VAE model. *Bottom Row:* The DM-VAE model trained without the information loss. The model still learns a good reconstruction with its memory, but it may get help from different slots.

A central feature of DM-VAE is that the hierarchical design makes it possible to summarize basic building blocks and distill the low-level knowledge into the memory buffer. We illustrate this feature in Fig.5 *left* by visualizing all the 15 slots of the model used in this experiment. By saying visualization, we mean that we sample from each g_{m_i}. As we can see, DM-VAE indeed allocates each slot the latent knowledge of a "shared template". Note that we use a slightly larger number of slots than the actual number of primitive types so that some slots may contain the same type of primitive or become useless. We also do the ablation study of removing the information loss used to constrain information flow during training. Shown in Fig.4 *Bottom Row*, though this still provide a good reconstruction, the latent knowledge stored in memory slots now seems to be mixed together (see Fig.5 *right*). Therefore $q(z^{type})$ contains much less information to interpret the scene in a sense that each slot can provide similar amount of information. Together with Fig.4, the experiment result shows how the memory helps the model to understand the raw data better: since the buffer already provides good templates, the model only needs a little extra effort to recover the individual noise information.

In all our experiments, we show that our model is comparable to the similar kind of state-of-the-art generative models in object detection and reconstruction power. In the meanwhile, we show that our model can build its primitive library automatically by visualizing the memory buffer. We demonstrate the effect of the information constrained procedure through ablation study.

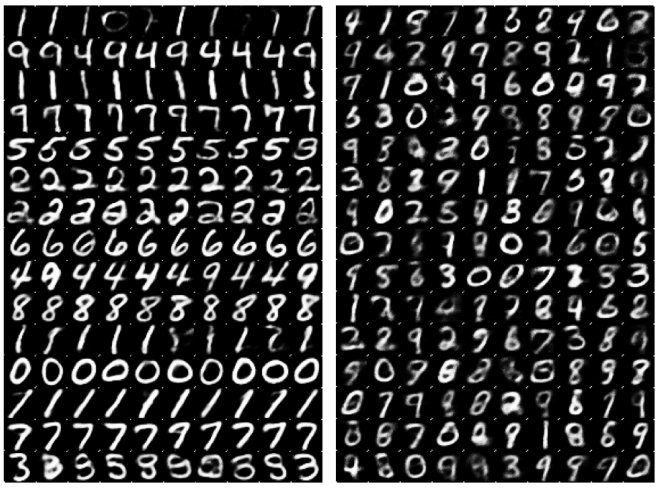


Fig. 5. Visualization of the memory by sampling images from each generator. Each row corresponds to a memory slot. *Left:*DM-VAE indeed allocates each slot the latent knowledge of a type of digit, i.e., a "shared template". *Right:*DM-VAE without the information loss.

C. Results on CASIA Characters

We now apply the model to the more challenging CASIA offline Chinese handwritten characters dataset. Note that the diverse writing styles and cursive con-junction of strokes make the dataset a more difficult one for comprehension tasks compared to other 2D or 3D synthetic datasets, whose primitives usually lack variety in shape. Parsing from handwritten Chinese characters into radicals is challenging. Such a task is often solved under a strong supervision signal as well as a handcrafted specific structure configuration such as [29]. We test both DM-VAE and AIR models for the parsing ability. We will show that, to some extent, DM-VAE can interpret the whole character in a part-relation form without any hand-design structure configuration.

We split 10 percent of the overall data as the test set and trains 3000 epochs. To demonstrate the model versatility, we keep the overall model architecture unchanged except that we increase the number of memory slot to 50.

We visualize the first 10 memory slots in Fig.6 at the training step 20k and 200k. The resulting buffer contains radical or stroke-like information, which is supposed essential for generating data by the model. Some rows represent a clear type of standard radical, and we add the label in Fig.6 for these rows. Although no supervision is given to the model, DM-VAE pays more attention to repeatedly occurred patterns in raw data and allocates empty memory slots to record them. As the learning progresses, the model is more certain to these patterns so that more slots become meaningful and clear.

We then let the model parse the test data into primitives it has already learned. For each image pair in Fig. 7, left is the original image, and right is the reconstruction using the parsed information. We have given the correct way of parsing these characters in the leftmost column, and we also make a comparison with AIR in Fig.7 *Right*. The model itself must

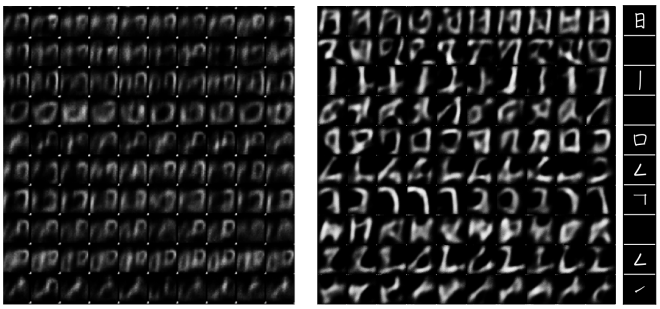
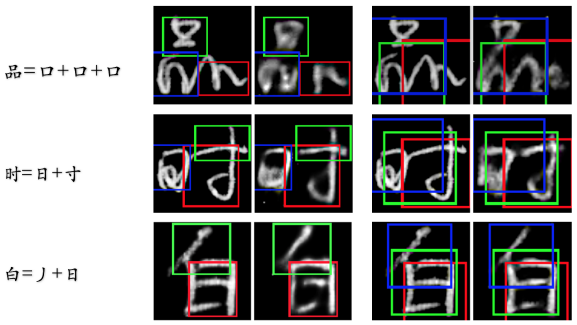


Fig. 6. The first 10 memory slots at training step 20k and 200k trained on character dataset. Each row in one image corresponds to a memory slot.



(a) DM-VAE (b) AIR

Fig. 7. Parsing Chinese characters. (a) DM-VAE can parse the data using primitives it has learned. (b) AIR is more inclined to recognize the image as a whole, which is not desirable.

decide how many primitives the data has as well as what type it is. For the AIR model, without the help of the primitive library, the model could only simply recognize and reconstruct the image as a whole and try to refine its reconstruction by adding more steps. Though by design the AIR model is encouraged to interpret the observation part-by-part through the sequential VAE, the attention windows of AIR are not well separated. In our experiments, training AIR model on the Chinese character dataset fails to converge from time to time. We suspect this is because the AIR model compresses too much information of various kinds of components into the same set of network parameters. Due to the discrete nature of the memory reader and the strong guide provided by the buffer, every step in DM-VAE is more likely to identify a local pattern. We have also reported the negative log-likelihood on this dataset in Tbl.I. Our model achieves a better performance in the quantitative result, which further proves the advantages of our methods.

VI. Conclusion

In this paper, we propose a novel hierarchical generative model of DM-VAE, which bridges the gap between traditional hierarchical models and deep generative models by combining a sequential VAE with a memory buffer. Together with the proposed training method, our model is capable of extracting the universal low-level knowledge and using the knowledge

to interpret high-level observations in a fully unsupervised manner. Extensive experiments demonstrate that the resulting method is able to search, identify, and memorize semantically meaningful primitive concepts.

ACKNOWLEDGEMENTS

This work was supported by the National Key Research and Development Program of China (No. 2017YFA0700904), NSFC Projects (Nos. 61620106010, U19B2034, U1811461), Beijing NSF Project (No. L172037), Beijing Academy of Artificial Intelligence (BAAI), Tsinghua-Huawei Joint Research Program, a grant from Tsinghua Institute for Guo Qiang, Tiangong Institute for Intelligent Computing, the JP Morgan Faculty Research Program and the NVIDIA NVAIL Program with GPU/DGX Acceleration.

REFERENCES

[1] P. D. Zelazo and S. P. Johnson, "Object perception," 12 2013.

[2] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[6] Z. Si and S.-C. Zhu, "Learning and-or templates for object recognition and detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 9, pp. 2189–2205, 2013.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[8] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton *et al.*, "Attend, infer, repeat: Fast scene understanding with generative models," in *Advances in Neural Information Processing Systems*, 2016, pp. 3225–3233.

[9] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner, "Monet: Unsupervised scene decomposition and representation," *arXiv preprint arXiv:1901.11390*, 2019.

[10] T. D. Kulkarni, V. K. Mansinghka, P. Kohli, and J. B. Tenenbaum, "Inverse graphics with probabilistic cad models," *arXiv preprint arXiv:1407.1339*, 2014.

[11] V. K. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum, "Approximate bayesian image interpretation using generative probabilistic graphics programs," in *Advances in Neural Information Processing Systems*, 2013, pp. 1520–1528.

[12] V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler, "The informed sampler: A discriminative approach to bayesian inference in generative computer vision models," *Computer Vision and Image Understanding*, vol. 136, pp. 32–44, 2015.

[13] K. Ellis, D. Ritchie, A. Solar-Lezama, and J. Tenenbaum, "Learning to infer graphics programs from hand-drawn images," in *Advances in Neural Information Processing Systems*, 2018, pp. 6062–6071.

[14] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess, "Unsupervised learning of 3d structure from images," in *Advances in Neural Information Processing Systems*, 2016, pp. 4996–5004.

[15] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[16] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in neural information processing systems*, 2015, pp. 2539–2547.

[17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[18] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016, pp. 2172–2180.

[19] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, p. 471, 2016.

[20] C. Li, J. Zhu, and B. Zhang, "Learning to generate with memory," in *International Conference on Machine Learning*, 2016, pp. 1177–1186.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[22] J. Bornschein, A. Mnih, D. Zoran, and D. J. Rezende, "Variational memory addressing in generative models," in *Advances in Neural Information Processing Systems*, 2017, pp. 3920–3929.

[23] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[24] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[25] D. B. F. Agakov, "The im algorithm: a variational approach to information maximization," *Advances in Neural Information Processing Systems*, vol. 16, p. 201, 2004.

[26] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[28] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "Casia online and offline chinese handwriting databases," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 37–41.

[29] J. Zhang, Y. Zhu, J. Du, and L. Dai, "Trajectory-based radical analysis network for online handwritten chinese character recognition," *arXiv preprint arXiv:1801.10109*, 2018.