A. Two Naive Methods

We present two naive methods for distilling knowledge from a teacher to a student for a general DGM as follows.

A.1. Marginalized Distillation

To minimize the desired distillation loss in Eq. (2), the most naive idea is to marginalize over all latent variables to analytically calculate the marginal distribution $p(\boldsymbol{y}|\boldsymbol{x}) = \int p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})\mathrm{d}\boldsymbol{z}$, and then directly minimize $d\left(p_\phi(\boldsymbol{y}|\boldsymbol{x}), p_\theta(\boldsymbol{y}|\boldsymbol{x})\right)$. However, this idea is infeasible, because $p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})$ is generally a very complicated black-box distribution parameterized by DNNs. Hence, it is generally intractable for us to calculate $p(\boldsymbol{y}|\boldsymbol{x}) = \int p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})\mathrm{d}\boldsymbol{z}$, and to derive a closed-form solution for marginal distribution $p(\boldsymbol{y}|\boldsymbol{x})$.

The next question is, if we can not analytically calculate $p(\boldsymbol{y}|\boldsymbol{x}) = \int p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})\mathrm{d}\boldsymbol{z}$, can we numerically estimate $p(\boldsymbol{y}|\boldsymbol{x})$ based on Monte Carlo method? The Monte Carlo estimation of $p(\boldsymbol{y}|\boldsymbol{x})$ can be written as follows.

$$\hat{p}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} p(\boldsymbol{y}|\boldsymbol{z}^{(n)}, \boldsymbol{x}), \tag{10}$$

where $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ is a set of samples drawn from $p(\boldsymbol{z}|\boldsymbol{x})$. Unfortunately, this idea is also problematic. There are two main issues of this idea: i) even if $p(\boldsymbol{y}|\boldsymbol{z}^{(n)}, \boldsymbol{x})$ is indeed a simple distribution (e.g. Gaussian distribution, Bernoulli distribution), $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ is not a simple distribution in most cases. For example, in the cases where $\boldsymbol{y}$ is not a scalar random variable (e.g., multi-label learning and generative modeling of high-dimensional data) or $\boldsymbol{y}$ is a continuous variable (e.g, regression), $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ is a complicated mixture distribution. Consequently, in these cases, $d\left(\hat{p}_\phi(\boldsymbol{y}|\boldsymbol{x}), \hat{p}_\theta(\boldsymbol{y}|\boldsymbol{x})\right)$ is generally intractable and has no closed-form solution for most commonly used measure d (e.g. KL Divergence and Wasserstein Distance). ii) even if in some rare cases where $\boldsymbol{y}$ is both scalar and discrete, although $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ becomes a simple distribution, $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ is unfortunately a high-variance estimation of $p(\boldsymbol{y}|\boldsymbol{x})$, because of the limitation of Monte Carlo method. The variance is especially high if $\boldsymbol{z}$ is high-dimensional, which seriously undermines the scalability of Monte Carlo based marginal distillation to the general DGMs.

In conclusion, both the original and Monte Carlo based marginalized distillation are generally intractable in most cases, especially in our experiments.

A.2. Local Distillation

Another naive method is called local distillation, which applies distillation in a layer-wise manner in deep DGMs, as shown in Fig. 3d. In local distribution, each of student's local component is encouraged to mimic its corresponding teacher's local component in an independent manner. With this method, we can factorize the joint distribution of a deep DGM into a product of a series of conditional distributions, as shown in Eq. (1), so that the distillation loss of each conditional distribution can be calculated separately. Formally, the distillation loss for target variable $\boldsymbol{y}_j$ is given by

$$\mathcal{L}_{y,j} = \mathbb{E}_{p_{data}(\boldsymbol{x})p_\phi(Pa(\boldsymbol{y}_j)|\boldsymbol{x})}\left[d\left(p_\phi(\boldsymbol{y}_j|Pa(\boldsymbol{y}_j), \boldsymbol{x}), p_\theta(\boldsymbol{y}_j|Pa(\boldsymbol{y}_j), \boldsymbol{x})\right)\right] \tag{11}$$

Similarly, the distillation loss for latent variable $\boldsymbol{z}_i$ is

$$\mathcal{L}_{z,i} = \mathbb{E}_{p_{data}(\boldsymbol{x})p_\phi(Pa(\boldsymbol{z}_i)|\boldsymbol{x})}\left[d\left(p_\phi(\boldsymbol{z}_i|Pa(\boldsymbol{z}_i), \boldsymbol{x}), p_\theta(\boldsymbol{z}_i|Pa(\boldsymbol{z}_i), \boldsymbol{x})\right)\right] \tag{12}$$

For the above equations, all the expectations can be estimated using Monte Carlo method, and Monte Carlo samples can be drawn using ancestral sampling from the teacher model.

Finally, we sum up all the local distillation losses and then jointly minimize the overall distillation loss as follows.

$$\mathcal{L}_{kd} = \sum_j \mathcal{L}_{y,j} + \sum_i \mathcal{L}_{z,i} \tag{13}$$

Let $d\left(\cdot, \cdot\right)$ be KL divergence, then Eq. (13) can be simplified as

$$\mathcal{L}_{kd} = \mathbb{E}_{p_{data}(\boldsymbol{x})}\left[KL(p_\phi(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}))\right] \tag{14}$$

The above equation is the KL divergence between the joint probability on all the latent variables and target variables of the teacher and that of the student model. It can be easily proved by factorizing $p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})$.

For the above Eq. (14), if there are no latent variables in a DGM, and if we choose $d(\cdot,\cdot)$ to be KL divergence, the local distillation loss can be simply reduced to the distillation loss in Eq. (2) or can be expanded to Eq. (3). However, when there exist latent variables, local distillation will add extra constraints on latent variables, which is redundant and dampens the distillation performance in practice. This is because we are only interested in minimizing the dissimilarity between target variables rather than latent variables.

Next, we provide an insight of why local distillation does not work well in deep DGMs. Take the DGM in Fig. 3d as an illustrative example. After distillation, each conditional distribution in the student model may still slightly deviate from the teacher model because of the inferior capacity of the student model or the randomness of optimization process. We call this deviation from student to teacher *imitation error*. Imitation errors will accumulate through multiple layers. For instance, in the imitation process of variable $\boldsymbol{y}_1$, its local distillation loss in Eq. (11) is $\mathcal{L}_{y,1} = \mathbb{E}_{p_{data}(\boldsymbol{x})p_\phi(\boldsymbol{z}_1|\boldsymbol{x})}\left[d\left(p_\phi(\boldsymbol{y}_1|\boldsymbol{z}_1), p_\theta(\boldsymbol{y}_1|\boldsymbol{z}_1)\right)\right]$. Even if this local distillation loss for $\boldsymbol{y}_1$ is very well optimized, when the student's latent distribution $p_\theta(\boldsymbol{z}_1|\boldsymbol{x})$ deviates far from the teacher's distribution, $p_\phi(\boldsymbol{z}_1|\boldsymbol{x})$, there will be no guarantee that $p_\theta(\boldsymbol{y}_1|\boldsymbol{x})$ is a good approximation for $p_\phi(\boldsymbol{y}_1|\boldsymbol{x})$ because $p(\boldsymbol{y}_1|\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{z}_1|\boldsymbol{x})}p(\boldsymbol{y}_1|\boldsymbol{z}_1)$. $p_\theta(\boldsymbol{y}_1|\boldsymbol{z}_1)$ is only well constrained in the domain of $p_\phi(\boldsymbol{z}_1|\boldsymbol{x})$. As a result, imitation errors may accumulate through layers of latent variables in a deep DGM.

We also conduct a toy experiment to show the error accumulation issue in local distillation. In our experiment, we let student mimic the output distribution of the teacher with L-layers latent variables for $L \in 1, \ldots, 20$. Each layer of the teacher follows the Gaussian distribution $p(\boldsymbol{z}_{i+1}|\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{z}_i), 0.01\boldsymbol{I})$, where $\boldsymbol{\mu}(\boldsymbol{z}_i)$ is $\boldsymbol{z}_i^{1.1}$ for $\boldsymbol{z}_i \geq 0$ and $-(-\boldsymbol{z}_i)^{1.1}$ for $\boldsymbol{z}_i < 0$. $p(\boldsymbol{z}_1)$ is a uniform distribution $U[-1, 1]$. The student is parameterized by neural networks with proper residual structure [18]. Then *density ratio estimation* [43, 51] is used to measure the KL divergence between the teacher and student. We can observe from Fig. 2 that the accumulated error (KL divergence) grows linearly w.r.t the number of layers for local distillation because each layer of the student deviates from the teacher to an extent. In contrast, the accumulated error (KL divergence) of our method increases slowly. We further conduct a similar toy experiment on models with discrete latent variables, as described in Appendix 4.1. It shows that, for models with discrete latent variables, local distillation is still afflicted by error accumulation issue, and our method can still mitigate this problem effectively.

B. About Upper Bound of Vanilla Distillation Loss

We present the detailed proof of Proposition 3.1 below.

Proof. Since the distributions of the auxiliary variables for both student and teacher are the same, i.e., $p_\phi(\boldsymbol{\epsilon}) = p_\theta(\boldsymbol{\epsilon})$, the surrogate distillation loss as in Eq. (4) can be rewritten as follows.

$$\begin{aligned}\mathcal{L}_{sd} &= \mathbb{E}_{p_\phi(\boldsymbol{\epsilon})p_{data}(\boldsymbol{x})p_\phi(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})}\left[\log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})}{p_\theta(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})}\right] \\ &= \mathbb{E}_{p_\phi(\boldsymbol{\epsilon})p_{data}(\boldsymbol{x})p_\phi(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})}\left[\log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})p_\phi(\boldsymbol{\epsilon})}{p_\theta(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})p_\theta(\boldsymbol{\epsilon})}\right]\end{aligned}$$

Notice that $p_\phi(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})p_\phi(\boldsymbol{\epsilon}) = p_\phi(\boldsymbol{y},\boldsymbol{\epsilon}|\boldsymbol{x}) = p_\phi(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x})p_\phi(\boldsymbol{y}|\boldsymbol{x})$. Similarly, it also holds that $p_\theta(\boldsymbol{y}|\boldsymbol{\epsilon},\boldsymbol{x})p_\theta(\boldsymbol{\epsilon}) = p_\theta(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x})p_\theta(\boldsymbol{y}|\boldsymbol{x})$. Thus, we can further rewrite the surrogate distillation loss $\mathcal{L}_{sd}$ as below.

$$\begin{aligned}\mathcal{L}_{sd} &= \mathbb{E}_{p_{data}(\boldsymbol{x})p_\phi(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x})p_\phi(\boldsymbol{y}|\boldsymbol{x})}\left[\log \frac{p_\phi(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x})}{p_\theta(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x})} + \log \frac{p_\phi(\boldsymbol{y}|\boldsymbol{x})}{p_\theta(\boldsymbol{y}|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{p_{data}(\boldsymbol{x})p_\phi(\boldsymbol{y}|\boldsymbol{x})}\left[KL\left(p_\phi(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x}) \parallel p_\theta(\boldsymbol{\epsilon}|\boldsymbol{y},\boldsymbol{x})\right)\right] + \mathbb{E}_{p_{data}(\boldsymbol{x})}\left[KL\left(\log p_\phi(\boldsymbol{y}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{y}|\boldsymbol{x})\right)\right]\end{aligned}$$

Since the first term is also non-negative, it further holds that

$$\mathcal{L}_{sd} \geq \mathbb{E}_{p_{data}(\boldsymbol{x})}\left[KL\left(\log p_\phi(\boldsymbol{y}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{y}|\boldsymbol{x})\right)\right] = \mathcal{L}_{kd}$$

which finishes our proof.

□

C. Algorithm Summary

We summarize our knowledge distillation method for a general DGM in Algorithm 1. Please note that we use $\boldsymbol{v}_i$ to denote a non-root random variable in DGMs, which is either a latent variable $\boldsymbol{z}_i$ or a target variable $\boldsymbol{y}_i$. $\boldsymbol{v}_{\phi,i}$ and $\boldsymbol{v}_{\theta,i}$ denote a sampling point of node $\boldsymbol{v}_i$ in the teacher DGM and the student, respectively. It can be observed from the distillation process that we only need to traverse the graph once to compute both the surrogate distillation loss and the latent distillation losses. Line 9 is our latent distillation loss. In Line 10-12, we describe the latent variables of the teacher and student model that are shared indirectly via sharing the auxiliary variables $\boldsymbol{\epsilon}$. Line 14 computes the surrogate distillation loss. Line 15-16 describe that the target variables of teacher and student model are shared directly. Line 19 presents the back propagation and gradient descent.

Algorithm 1 Our Distillation Method

Require: teacher DGM $\mathcal{G}_\phi$, student DGM $\mathcal{G}_\theta$, learning rate η, hyperparameter λ, empirical distribution of input dataset $p_{data}(\boldsymbol{x})$ (only required when $\boldsymbol{x} \neq \emptyset$)
Ensure: distilled student DGM $\mathcal{G}_\theta$

1: **while** θ not converged **do**
2: sample mini-batched $\boldsymbol{x}$ from $p_{data}(\boldsymbol{x})$ // (only required when $\boldsymbol{x} \neq \emptyset$)
3: set $L_{our} = 0$
4: **while** $\mathcal{G}$ not fully traversed **do**
5: grab variable $\boldsymbol{v}_i$ where $Pa(\boldsymbol{v}_i)$ have all been sampled
6: set $p_{\phi,i} = p_\phi(\boldsymbol{v}_i|Pa(\boldsymbol{v}_i), \boldsymbol{x})$
7: set $p_{\theta,i} = p_\theta(\boldsymbol{v}_i|Pa(\boldsymbol{v}_i), \boldsymbol{x})$
8: **if** $\boldsymbol{v}_i$ is latent variable **then**
9: $\mathcal{L}_{our} \leftarrow \mathcal{L}_{our} + \lambda d(p_{\phi,i}, p_{\theta,i})$
10: sample $\boldsymbol{\epsilon}_i$ from $p_\phi(\boldsymbol{\epsilon}_i)$
11: sample $\boldsymbol{v}_{\phi,i}$ from $p_{\phi,i}$ according to $\boldsymbol{\epsilon}_i$ // set $\boldsymbol{v}_{\phi,i} = g_\phi(Pa(\boldsymbol{v}_i), \boldsymbol{x}, \boldsymbol{\epsilon}_i)$
12: sample $\boldsymbol{v}_{\theta,i}$ from $p_{\theta,i}$ according to $\boldsymbol{\epsilon}_i$ // set $\boldsymbol{v}_{\theta,i} = g_\theta(Pa(\boldsymbol{v}_i), \boldsymbol{x}, \boldsymbol{\epsilon}_i)$
13: **else** {$\boldsymbol{v}_i$ is target variable}
14: $\mathcal{L}_{our} \leftarrow \mathcal{L}_{our} + d(p_{\phi,i}, p_{\theta,i})$
15: sample $\boldsymbol{v}_{\phi,i}$ from $p_{\phi,i}$
16: $\boldsymbol{v}_{\theta,i} \leftarrow \boldsymbol{v}_{\phi,i}$
17: **end if**
18: **end while**
19: $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_{our}}{\partial \theta}$
20: **end while**

D. Datasets

To evaluate the performance of the proposed distillation method, we conduct experiments on the following benchmark datasets.

- **Old Faithful Geyser** [17]: this is 2-dimensional Geyser data. We use all the 272 samples for training.
- **IAM Online Handwriting** [38]: it has 13040 samples and each sample is a sequence of (x, y) coordinate together with binary indicator of pen up/down. We use all the examples for training.
- **SVHN** [42]: This dataset is about house numbers in street views with 1 categorical label of 10 classes. We use 73257 images for training and 26032 images for testing respectively.
- **CIFAR10** [28]: This dataset consists of 60000 natural world images, with 1 categorical label of 10 classes. The data is split into 50000 and 10000 images for training and testing.
- **CelebA** [37]: It contains $182,732$ RGB $128 \times 128 \times 3$ images of celebrity faces. We use 162770 images for training and 19962 images for testing.

E. Model Configurations and Hyper-parameter Settings

We summarize the detailed model configurations and hyper-parameter settings for the proposed method in different tasks below.

For all the tasks, we choose $d(\cdot,\cdot)$ in the surrogate distillation loss $\mathcal{L}_{sd}$ to be KL divergence. When latent variables are discrete, $d(\cdot,\cdot)$ in the latent distillation loss $\mathcal{L}_{z,i}$ is chosen to be KL divergence and λ is set to 1. When latent variables are continuous, we choose $d(\cdot,\cdot)$ in $\mathcal{L}_{z,i}$ to be squared 2-Wasserstein distance because it has analytical solution on two Gaussian distributions. In addition, we normalize $\mathcal{L}_{z,i}$ w.r.t. the size of vector $\boldsymbol{z}_i$. We empirically found that normalized latent distillation loss works slightly better than unnormalized one. In all the following experiments, λ is chosen after a simple grid search.

The main training configurations for different tasks are summarized in the Table 3 below.

Table 3. Summary of model configurations

Configuration	VAE Compression	VRNN Compression	HM Compression
Teacher size	5.39M-6.60M	12M	354
Student size	0.01M-0.44M	2.4M	102
Optimizer	Adam	Adam	Adam
Learning rate	1e-4	1e-4	1e-2
Weight decay	1e-4	1e-4	1e-4
Batch size	128-256	32	272

E.1. Toy Examples

In our continuous and discrete toy examples, the conditional distributions of student model is parameterized by a 2-layer MLP with hidden dimension 2 and residual structure. We chose Adam optimizer with learning rate 1e-2 and weight decay 1e-4.

E.2. Data-free VAE Compression

In our experiments, we distill a VAE with the same DGM and parameterization as the Ladder VAE [50]. The DGM structure of its generative model (decoder) is shown in Fig. 4a. Slightly different from [50], we use Convolutional Neural Network (CNN) to parameterize both the bottom-up and top-down network in VAE, instead of simple MLP. The bottom-up network and top-down network have symmetric structure and almost the same number of parameters. In addition, we adopt spatial latent variables (latent variables are spatially arranged, as the feature map in CNN) in the neural networks. Each CNN is composed of consecutive residual blocks [18]. To demonstrate the robustness of our method, we conduct extensive experiments on different datasets using different student model sizes.

In the experiments, for both the teacher and student VAE, each latent variable $\boldsymbol{z}_i$ is a spatially arranged 3-order tensors with the dimension (H_i, W_i, C). The height H_i and width W_i are varied from 1 to 32 according to different i. The channel C of latent variable tensors is fixed to 32. We change the size of student model by varying the hidden dimension of the CNNs. The hidden dimension of CNNs is varied from 4 to 32 in the student models, and is set to 128 for the teacher model. Mini-batch sizes are set to 256 for SVHN and CIFAR10, and 128 for CelebA. When VAE is trained from scratch, we adopt linear β annealing, which is called *warm up* in [50] for first half of the iterations to stabilize model learning. For CelebA dataset, we resize the images to the shape $64 \times 64 \times 3$. Model training is stopped after 57k, 46k, 63k iterations for SVHN, CIFAR10, and CelebA, respectively.

E.3. KD-based VAE Continual Learning

In our experiment, we still use the same 5-layer VAE as the above VAE compression experiment. Following [46], we add an additional discrete input variable to the graph for the sake of indicating where the data comes from. As a result, our VAE model is turned to a 6-layer Conditional VAE, as illustrated in Fig. 4d.

For KD-based VAE continual learning, we have the following distillation loss.

$$\begin{aligned}\tilde{L}_{continual} &= \mathbb{E}_{\frac{1}{2}(p_{old}(\boldsymbol{y})+p_{new}(\boldsymbol{y}))}\left[-\log p_\theta(\boldsymbol{y})\right] \\ &= \frac{1}{2}(\mathbb{E}_{p_{old}(\boldsymbol{y})}\left[-\log p_\theta(\boldsymbol{y})\right] + \mathbb{E}_{p_{new}(\boldsymbol{y})}\left[-\log p_\theta(\boldsymbol{y})\right]) \\ &= \frac{1}{2}(KL[p_{old}(\boldsymbol{y}) \parallel p_\theta(\boldsymbol{y})] + KL[p_{new}(\boldsymbol{y}) \parallel p_\theta(\boldsymbol{y})]) + c\end{aligned} \tag{15}$$

where c is a constant independent of θ. $p_{old}(\boldsymbol{y})$ and $p_{new}(\boldsymbol{y})$ are the empirical distribution of old and new dataset, respectively. For the above loss, since $p_{old}(\boldsymbol{y})$ is not accessible, we use the old learned VAE's distribution $p'_\theta(\boldsymbol{y})$ as a replacement. Accordingly, we replace $KL[p_{old}(\boldsymbol{y}) \parallel p_\theta(\boldsymbol{y})]$ with $KL[p'_\theta(\boldsymbol{y}) \parallel p_\theta(\boldsymbol{y})]$. This results in our desired distillation loss, transferring knowledge from teacher (old VAE model) $p'_\theta(\boldsymbol{y})$ to student (new VAE model) $p_\theta(\boldsymbol{y})$. However, this desired distillation loss is intractable as discussed earlier. To tackle this issue, we use our distillation loss $\mathcal{L}_{our}$ instead. As a result, the total loss for continual learning is rewritten as

$$L_{continual} = \frac{1}{2}(L_{our} + KL[p_{new}(\boldsymbol{y}) \parallel p_\theta(\boldsymbol{y})]) \tag{16}$$

In this experiment, we follow the model configurations as the above VAE compression, except that we resize images to $32 \times 32 \times 3$ and do not use running statistics for batch normalization layers during evaluation.

E.4. Data-free VRNN Compression

In this experiment, we build and distill a VRNN with the same DGM and parameterization as in [12], as illustrated in Fig 4b. VRNN consists of as many latent variables and target variable as its sequence length. We fix the number of hidden layer in all densely connected networks to 1, and fix the internal RNN to be 1-layer LSTM as in [12]. Each latent variable is set to 1. For the teacher model, the hidden dimension of LSTM is set to 1200. Feature size of target variables, latent variables, and hidden dimension of all densely connect networks are set to 512. For the student model, the hidden dimension of LSTM is set to 600. Feature dimension of target/latent variables, and hidden dimension of all densely connect networks are set to 128.

E.5. Data-free HM Compression

Helmholtz Machine (HM) [41, 49] is a classical generative modeling technology [3, 4, 15, 54] that consists of: 1) Sigmoid Belief Network (SBN) as generative graphical model; 2) another SBN as amortized inference network; 3) Wake-Sleep learning strategy. The classical HM has only one target variable and each layer is parameterized by a linear transformation. In our experiment, in order to demonstrate the applicability of our distillation method, we extend it to a 5-layer DGM with two target variables, as shown in Fig. 4c. The inference network is also modified to match the true posterior dependency structure. Following the similar idea in [27] and [3], each conditional distribution in both generative and inference network are parameterized by 2-layer MLP instead of linear transformation, which turns our HM to *deep HM*.

In our experiment, we fix latent variables to two-dimensional vectors with binary elements. The hidden dimension of MLP is set to 8 and 2 for teacher and student model respectively. The inference model (encoder) has symmetric structure and almost the same number of parameters as the generative model (decoder).

F. Qualitative Results on CelebA

We also conduct experiments to compare the generated samples by different knowledge distillation (KD) methods on the CelebA dataset, as illustrated in Fig. 6. It can be observed from Fig. 6(c) that images generated by our method are of high-fidelity, which means the student model distilled by our method can effectively imitate the teacher's performance. However, the images generated by the student VAE trained from scratch are of low quality, as illustrated in Fig. 6(d), which suggests that a limited-size VAE model is incapable of learning such a complicated distribution from scratch on its own.

F.1. Hyperparameter Stability in the Loss

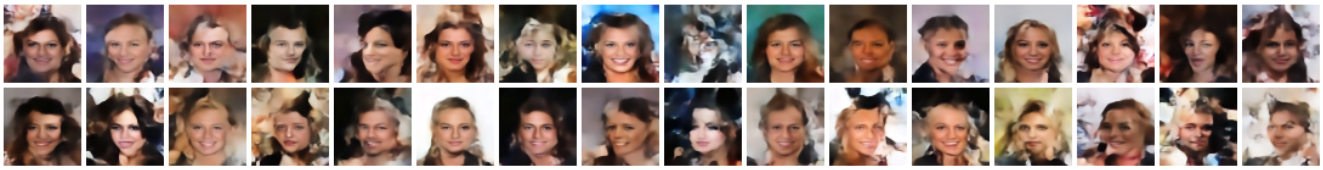
We demonstrate the hyper-parameter stability of our method. Fig. 7 shows the relationship between λ and FID score. It can be observed that as λ varies from to 0.0 to $1e3$, the FID score of our distillation method does not vary too much. It means that our method is stable and robust to the hyper-parameter λ.

(a) Ground truth

(b) Teacher

(c) Our method

(d) Local distillation



(e) From scratch

Figure 6. Images generated by VAE trained on CelebA dataset. (a) Ground truth dataset. (b) Teacher VAE (5.4M parameters) trained from scratch. (c) Student VAE (1.4M parameters, 3.9 times smaller) distilled by our method. (d) Student VAE distilled using local distillation. (e) Student VAE trained from scratch.

G. Evaluation on Data-free VRNN Compression

We also evaluate the proposed KD method on data-free VRNN compression. as shown in Fig 8. We can observe that our KD based method can generate handwriting strokes with higher quality compared to the baselines: local distillation and model training from scratch.

H. Evaluation on Data-free Helmholtz Machine Compression

We apply the proposed approach to compress a 5-layer HM trained with Wake-Sleep algorithm to a smaller HM in a data-free manner. We still compare it with the baselines: training from scratch and local distillation. The experimental results are shown in Fig. 9.

Importantly, We can observe that our distillation method outperforms the student model trained from scratch for all VAE, VRNN and HM compression. It suggests that directly optimizing a capacity-limited model is incapable of high-fidelity generation. In contrast, our method can learn a small high-fidelity generative student model from its teacher even without accessing training data.

In our experimental results, we find that there is tiny difference between our method and local distillation, partly because this 2-dimensional modeling task is too easy for highly flexible HMs to learn. Both our method and local distillation can

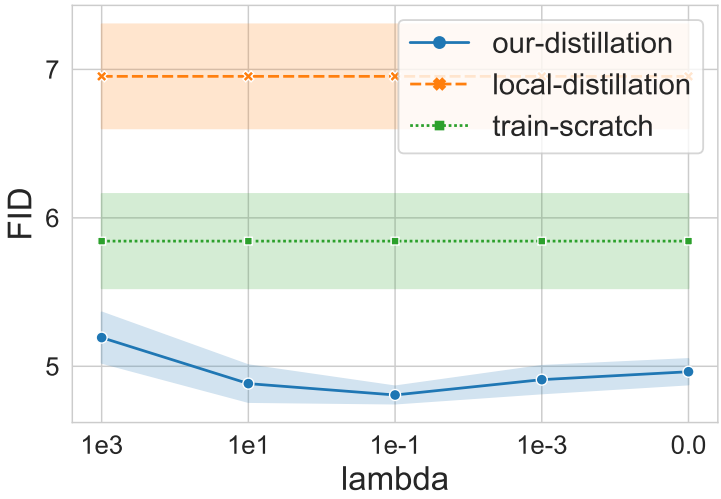
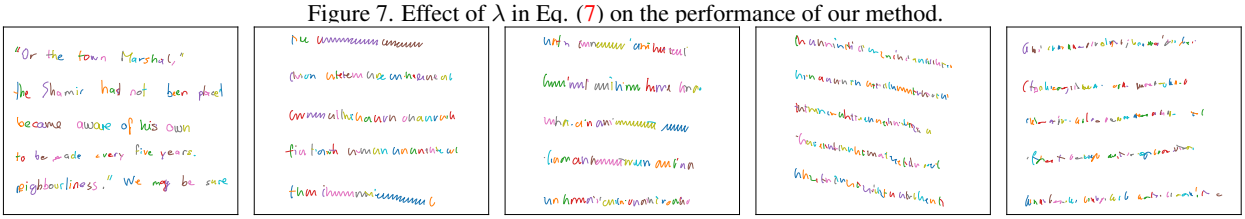
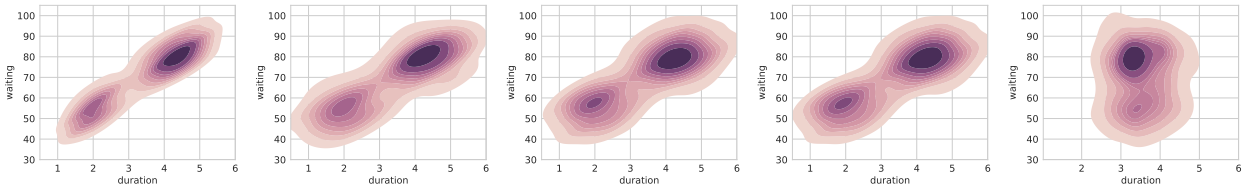


Figure 7. Effect of λ in Eq. (7) on the performance of our method.



(a) Ground truth (b) Teacher (c) Our method (d) Local distillation (e) From scratch

Figure 8. Strokes generated by VRNNs. Different strokes are represented with different colors. (a) Ground truth. (b) Teacher VRNN (12M parameters) trained from scratch. (c) Student VRNN (2.4M parameters, 5.0 times smaller) distilled by our method. (d) Student VRNN distilled using local distillation. (e) Student VRNN trained from scratch.



(a) Ground truth (b) Teacher (c) Our method (d) Local distillation (e) From scratch

Figure 9. Generation distribution of HMs. We performed and plot kernel density estimation using 272 generated samples for each HM. **(a)** Ground truth. **(b)** Teacher HM (354 parameters) trained from scratch. **(c)** Student HM (102 parameters, 3.5 times smaller) distilled using our method. **(d)** Student HM distilled using local distillation. **(e)** Student HM trained from scratch.

converge after a few iterations. Hence, they have comparable performance to each other.

I. Discussions and Future Work

We further discuss the limitation of the proposed framework. First, our framework is only applied to DGMs where latent variables can be reparameterized.

However, We would like to point out that the definition of reparameterization trick in our framework is slightly different from the classical one for VAE training [25]. Reparameterization in our work is only used for converting latent variables to auxiliary forms. We do not need to propagate gradients through these latent variables.

Another limitation of our framework is that it needs to consist of the same number of random variables (these variables can be arranged in different structures). For future work, we plan to extend the proposed method to deal with more difficult cases where the teacher and students have different number of random variables