A. Extended Derivations and Further Discussion

A.1. Derivation of Conditional EBMs

We first define the marginal EBMs at each diffusion step:

$$\begin{cases} p_\alpha(\mathbf{z}_t) = \dfrac{1}{Z_{\alpha,t}} \exp(F_\alpha(\mathbf{z}_t, t)) p_0(\mathbf{z}_t), \ t = T-1 \\ p_\alpha(\mathbf{z}_t) = \dfrac{1}{Z_{\alpha,t}} \exp(F_\alpha(\mathbf{z}_t, t)), \ t = 0, 1, ..., T-2 \end{cases} \tag{A1}$$

where the marginal energy term is in a log-sum-exponential form $F_\alpha(\mathbf{z}_t, t) = \log \sum_{\mathbf{y}} \exp(\langle \mathbf{y}, f_\alpha(\mathbf{z}_t, t) \rangle)$; it serves to aggregate the energy score from each category. Of note, the marginal EBM corresponding with the last diffusion step has a slightly different definition. We set this term as exponential tilting of a non-informative Gaussian prior $p_0(\mathbf{z}_t)$ which helps to stabilize training in practice.

Recall that $\mathbf{z}_{t+1} = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$. Let $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$. For $t = 0, 1, ..., T-2$, we have

$$\begin{aligned} p_\alpha(\tilde{\mathbf{z}}_t | \mathbf{z}_{t+1}) &= \frac{p_\alpha(\tilde{\mathbf{z}}_t) p(\mathbf{z}_{t+1} | \tilde{\mathbf{z}}_t)}{p_\alpha(\mathbf{z}_{t+1})} \\ &= \frac{1}{\tilde{Z}_{\alpha,t}} \frac{\exp(F_\alpha(\tilde{\mathbf{z}}_t, t))}{p_\alpha(\mathbf{z}_{t+1})} \exp\left(-\frac{1}{2\sigma_{t+1}^2} ||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2 \right) \\ &= \frac{1}{\tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1})} \exp\left(F_\alpha(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2} ||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2 \right), \end{aligned} \tag{A2}$$

where $\tilde{Z}_{\alpha,t} = (2\pi\sigma_{t+1}^2)^{\frac{n}{2}} Z_{\alpha,t}$; we slightly abuse the notation and use $p(\mathbf{z}_{t+1}|\tilde{\mathbf{z}}_t)$ to represent the forward transition $q(\mathbf{z}_{t+1}|\mathbf{z}_t)$ defined in Eq. (4) for notation consistency.

The diffused samples at time step T are close to Gaussian white noise; $p_\alpha(\tilde{\mathbf{z}}_{T-1}|\mathbf{z}_T)$ therefore falls to its marginal distribution $p(\tilde{\mathbf{z}}_{T-1})$ defined in Eq. (A1).

A.2. Derivation of the ELBO

Recall that the ELBO in SVEBM is

$$\begin{aligned} \text{ELBO}_{\theta,\phi} &= \log p_\theta(\mathbf{x}) - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z})] - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\alpha(\mathbf{z})) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\beta(\mathbf{x}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\alpha(\mathbf{z})\right], \end{aligned} \tag{A3}$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence. Let us consider the full trajectory of the perturbed samples $\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_T$. The above equation can be written as

$$\begin{aligned} \text{ELBO}_{\theta,\phi} = \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log p_\beta(\mathbf{x}|\mathbf{z}_0) - \log q_\phi(\mathbf{z}_0|\mathbf{x})\right] \\ + \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} p_\alpha(\mathbf{z}_{0:T}) d\mathbf{z}_{1:T}\right], \end{aligned} \tag{A4}$$

where the last term is further lower-bounded by introducing the forward trajectory distribution; the inequality holds by applying Jensen's Inequality:

$$\begin{aligned} & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} p_\alpha(\mathbf{z}_{0:T}) d\mathbf{z}_{1:T}\right] \\ = \ & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} q(\mathbf{z}_{1:T}|\mathbf{z}_0) \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} d\mathbf{z}_{1:T}\right] \\ \geq \ & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\int_{\mathbf{z}_{1:T}} q(\mathbf{z}_{1:T}|\mathbf{z}_0) \log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} d\mathbf{z}_{1:T}\right] \\ = \ & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x}), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right]. \end{aligned} \tag{A5}$$

Further, we can decompose the joint distribution of forward and backward trajectories as

$$\begin{aligned} & \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x}), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right] = \\ & \mathbb{E}_{q_\phi(\mathbf{z}_0|x), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log \frac{p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})}{q(\mathbf{z}_{t+1}|\mathbf{z}_t)}\right] = \\ & \mathbb{E}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right] + \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t), \end{aligned} \tag{A6}$$

where $p(\mathbf{z}_T)$ is standard Gaussian distribution; $\mathbb{E}$ is the abbreviation of $\mathbb{E}_{q_\phi(\mathbf{z}_0|x), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}$. $\mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t), t = 0, ..., 1$ is the conditional entropy under the forward trajectory distribution. We obtain $\mathbf{z}_t$ by sampling $\tilde{\mathbf{z}}_t$ from $p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ and then applying $\mathbf{z}_t = \tilde{\mathbf{z}}_t / \sqrt{1-\sigma_{t+1}^2}$; the reverse trajectory in our model is primarily defined by $p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ for $t > 0$. We use $[\mathbf{z}_t|\mathbf{z}_{t+1}]$ to represent this process in the following sections; we may interchangeably use the notation of $\tilde{\mathbf{z}}_t$ and $\mathbf{z}_t$ for simplicity.

Note that the entropies can be analytically computed and do not involve learnable parameters. The joint training of inference, prior and generation models can be largely reduced to finding the agreement of the forward and reverse Markov transitions defined by q_ϕ and p_θ respectively.

A.3. Detailed Discussion of Symbol Coupling

In Sec. 2, we briefly describe how to introduce the symbolic one-hot vector $\mathbf{y}$. Since only $\mathbf{z}_0$ is connected with $\mathbf{y}$, we can first define the joint prior $p_\alpha(\mathbf{y}, \mathbf{z}_0)$ as in Eq. (A1) by substituting $F_\alpha(\tilde{\mathbf{z}}_0, 0)$ with $\langle \mathbf{y}, f_\alpha(\tilde{\mathbf{z}}_0, 0) \rangle$. Then the conditional symbol-vector coupling joint distribution follows as

$$\begin{aligned} p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) = \frac{1}{\tilde{Z}_{\alpha,t=0}} \exp\left(\langle \mathbf{y}, f_\alpha(\tilde{\mathbf{z}}_0, 0) \rangle\right) \\ \exp\left(-\frac{1}{2\sigma_1^2} ||\tilde{\mathbf{z}}_0 - \mathbf{z}_1||^2 \right). \end{aligned} \tag{A7}$$

Note that $p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) = p_\alpha(\mathbf{y}|\mathbf{z}_0)p_\alpha(\mathbf{z}_0|\mathbf{z}_1)$, *i.e.*, $\mathbf{z}_0$ is sufficient for inferring $\mathbf{y}$ in this formulation:

$$\begin{aligned} p_\alpha(\mathbf{y}|\mathbf{z}_0, \mathbf{z}_1) &= \frac{p_\alpha(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1)}{p_\alpha(\mathbf{z}_0|\mathbf{z}_1)} \\ &= \frac{\exp\left(\langle \mathbf{y}, f_\alpha(\tilde{\mathbf{z}}_0, 0)\rangle\right)}{\exp\left(F_\alpha(\tilde{\mathbf{z}}_0, 0)\right)}, \end{aligned} \tag{A8}$$

so that given $\mathbf{z}_0$,

$$p_\alpha(\mathbf{y}|\mathbf{z}_0) \propto \exp(\langle \mathbf{y}, f_\alpha(\tilde{\mathbf{z}}_0, 0)\rangle). \tag{A9}$$

It similarly becomes a softmax classifier where $f_\alpha(\tilde{\mathbf{z}}_0, 0)$ provides the logit scores for the K categories.

A.4. Derivation of the Information Bottleneck

We first define the mutual information term between $\mathbf{z}_0$ and $\mathbf{y}$. Consider the joint distribution of $\mathbf{x}, \mathbf{z}_0$ and $\mathbf{y}$, $\pi(\mathbf{y}, \mathbf{z}_0, \mathbf{x}) = p_\alpha(\mathbf{y}|\mathbf{z}_0)q_\phi(\mathbf{z}_0|\mathbf{x})q_{\text{data}}(\mathbf{x})$; the mutual information $\mathcal{I}(\mathbf{z}_0, \mathbf{y})$ defined under π then follows as:

$$\begin{aligned} \mathcal{I}(\mathbf{z}_0, \mathbf{y}) &= \mathcal{H}(\mathbf{y}) - \mathcal{H}(\mathbf{y}|\mathbf{z}_0) \\ &= -\sum_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y}) \\ &\quad + \mathbb{E}_{q_\phi(\mathbf{z}_0)} \sum_{\mathbf{y}} p_\alpha(\mathbf{y}|\mathbf{z}_0) \log p_\alpha(\mathbf{y}|\mathbf{z}_0), \end{aligned} \tag{A10}$$

where $q(\mathbf{y}) = \mathbb{E}_{q_\phi(\mathbf{z}_0)}[p_\alpha(\mathbf{y}|\mathbf{z}_0)]$; $p_\alpha(\mathbf{y}|\mathbf{z}_0)$ is the softmax probability over K categories in Eq. (A9).

We then show how to obtain the quantities defined in Sec. 3.2. For the marginal distribution of $\mathbf{z}_0$:

$$\begin{aligned} q_\phi(\mathbf{z}_0) &= \int_{\mathbf{x}, \mathbf{z}_{1:T}} Q_\phi(\mathbf{x}, \mathbf{z}_{0:T}) d\mathbf{x} d\mathbf{z}_{1:T} \\ &= \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}_0|\mathbf{x})]. \end{aligned} \tag{A11}$$

The entropy and conditional entropy of $\mathbf{z}_0$ are thus

$$\begin{aligned} &\mathcal{H}(\mathbf{z}_0) = -\mathbb{E}_{q_\phi(\mathbf{z}_0)}[\log q_\phi(\mathbf{z}_0)]; \\ &\mathcal{H}(\mathbf{z}_0|\mathbf{x}) = -\mathbb{E}_{Q_\phi(\mathbf{x}, \mathbf{z}_0)}[\log q_\phi(\mathbf{z}_0|\mathbf{x})]. \end{aligned} \tag{A12}$$

Taking together, we can then decompose the KL-Divergence, $\mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta)$, in Eq. (8) as:

$$\begin{aligned} \mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta) &= \mathbb{E}_{Q_\phi}\left[q_{\text{data}}(\mathbf{x})\right] + \mathbb{E}_{Q_\phi}\left[q_\phi(\mathbf{z}_{0:T}|x)\right] \\ &\quad - \mathbb{E}_{Q_\phi}\left[p_\alpha(\mathbf{z}_{0:T})\right] - \mathbb{E}_{Q_\phi}\left[p_\beta(\mathbf{x}|\mathbf{z}_0)\right], \end{aligned} \tag{A13}$$

and further as:

$$\begin{aligned} &- \mathcal{H}(\mathbf{x}) + \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t) - \mathcal{H}(\mathbf{z}_0|\mathbf{x}) + \mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0) \\ &\quad - \mathbb{E}_{Q_\phi}\left[p_\alpha(\mathbf{z}_{0:T})\right] - \mathbb{E}_{Q_\phi}\left[p_\beta(\mathbf{x}|\mathbf{z}_0)\right], \end{aligned} \tag{A14}$$

by plugging in $\mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0) = 0$. Rearranging Eq. (A14), we can obtain

$$\begin{aligned} \mathbb{D}_{\text{KL}}(Q_\phi \| P_\theta) &= \mathcal{C} - \mathbb{E}_{Q_\phi}\left[p_\beta(\mathbf{x}|\mathbf{z}_0)\right] \\ &\quad + \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}_0) \| p_\alpha(\mathbf{z}_{0:T})) + \mathcal{I}(\mathbf{x}, \mathbf{z}_0), \end{aligned} \tag{A15}$$

which leads to our result in Eq. (9).

A.5. Derivation of the Learning Gradient

Recall that we derive the extended version of Eq. (6) in Appx. A.2. To calculate the gradient of α, we have

$$\begin{aligned} \nabla_\alpha \text{ELBO}_{\text{Diff}, \theta, \phi} &= \nabla_\alpha \mathbb{E}\left[\sum_{t=0}^{T-1} \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\alpha \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right], \end{aligned} \tag{A16}$$

where $\mathbb{E}$ is the abbreviation of $\mathbb{E}_{q_\phi(\mathbf{z}_0|x), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}$; in practice, we use Monte-Carlo average to approximate the expectation. We next examine the learning gradient for each diffusion step t.

$$\nabla_\alpha \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1}) = \nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t) - \nabla_\alpha \tilde{Z}_{\alpha, t}(\mathbf{z}_{t+1}), \tag{A17}$$

where the quadratic term $\frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2$ is not related to α and gets cancelled. According to the definition of the partition function in Sec. 2, we can similarly derive

$$\nabla_\alpha \tilde{Z}_{\alpha, t}(\mathbf{z}_{t+1}) = \mathbb{E}_{p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}\left[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)\right], \tag{A18}$$

as in Pang et al. (2020a). For the prior model, we thus have

$$\begin{aligned} \nabla_\alpha \text{ELBO}_t &= \mathbb{E}_{q_\phi(\tilde{\mathbf{z}}_t, \mathbf{z}_0|\mathbf{x})}[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)] \\ &\quad - \mathbb{E}_{q_\phi(\mathbf{z}_{t+1}, \mathbf{z}_0|\mathbf{x}), p_\alpha(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}[\nabla_\alpha F_\alpha(\tilde{\mathbf{z}}_t, t)], \end{aligned} \tag{A19}$$

where $q_\phi(\tilde{\mathbf{z}}_t, \mathbf{z}_0|x) = q(\tilde{\mathbf{z}}_t|\mathbf{z}_0)q_\phi(\mathbf{z}_0|\mathbf{x})$. Note that we can sample $\mathbf{z}_t,\ t > 0$ directly from

$$q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t; \sqrt{\bar{\gamma}_t}\mathbf{z}_{t-1}, (1 - \bar{\gamma}_t)\mathbf{I}), \tag{A20}$$

by merging the Gaussian noises during forward diffusion process; we denote $\gamma_t = 1 - \sigma_t^2$ and $\bar{\gamma}_t = \prod_{i=1}^t \gamma_t$.

For the encoder and decoder, based on Eq. (6) and Eq. (A6), we have

$$\begin{aligned} \nabla_\psi \text{ELBO} &= \nabla_\psi \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z}_0) - \log q_\phi(\mathbf{z}_0|\mathbf{x})] \\ &\quad - \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}_{0:T}|\mathbf{x})}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_\alpha(\mathbf{z}_t|\mathbf{z}_{t+1})\right], \end{aligned} \tag{A21}$$

where the summation of energy terms provides extra guidance for the optimization of encoder.

Table A1. **Network architecture for the LDEBM prior.** N is set to 12 for all the experiments.

Layers	Output size	Note
Time Embedding		
Input: t	1	Index of diffusion step
Sin. embedding	200	
Linear, LReLU	200	negative_slope 0.2
Linear	200	
Input Embedding		
Input: $\mathbf{z}$	d_{lat}	
Linear, LReLU	200	negative_slope 0.2
Linear	200	
Context Embedding (for response generation only)		
Input: $\mathbf{z}_{\text{ctx}}$	512	ctx. embedding
Linear, LReLU	200	negative_slope 0.2
Linear	200	
LDEBM Prior		
Input: $\mathbf{z}, t$ $*\mathbf{z}_{\text{ctx}}$	1, d_{lat} 512	optional $\mathbf{z}_{\text{ctx}}$
Embedding	200	Embedding of each input
Concatenate	400 600	w/o ctx. w/ ctx.
LReLU, Linear	200	negative_slope 0.2
N ResBlocks	200	LReLU, Linear + Input
LReLU, Linear	K	K class logits
Log-Sum-Exp	1	energy score

B. Extra Experiment Details and Discussion

B.1. Network Architecture and Hyperparameters

We provide detailed network architecture for the latent space model of this work in Tab. A1; we adopt the same architecture throughout the experiments. Spectral normalization (Miyato et al., 2018) is used to regularize parameters in linear layers. The encoder and decoder in all models are the same as in Pang & Wu (2021), implemented with a single-layer GRU with a hidden size of 512. The key hyperparameters of LDEBM for each dataset are listed in Tab. A2. Of note, we use the same dimension of the latent space as in (Pang & Wu, 2021) for a fair comparison.

λ_1 is the hyperparameter that reweights the term in Eq. (A6); it generally controls how fast q_ϕ and p_θ run towards each other. λ_2 refers to the hyperparameter in Eq. (9); it controls the trade-off between the compressivity of $\mathbf{z}_0$ about $\mathbf{x}$ and its expressivity to $\mathbf{y}$. λ_3 controls the weight of classification loss mentioned in Sec. 3.3; recall that we use pseudo-label $\hat{\mathbf{y}}$ inferred by the geometric clustering algorithm or the ground-truth label $\mathbf{y}$ to supervise $p_\alpha(\mathbf{y}|\mathbf{z}_0)$ in our modeling. For controllable generation and semi-supervised classification, we find it important to have a larger weight on the classification loss so that the model is forced to capture the major modes of the data.

Table A2. **Hyperparameters of LDEBM.** DD-CLS presents the set of hyperparameters used in unsupervised clustering on DD dataset. DD-GEN presents the set of hyperparameters used in conditional response generation on DD dataset.

DATASET	d_{lat}	K	λ_1	λ_2	λ_3
2D GAUSSIAN	2	16	1	0.05	0.05
2D PINWHEEL	2	10	1	0.05	0.05
PTB	40	20	0.1	0.05	0.05
JERICHO	40	20	0.1	0.05	0.05
DD-CLS	32	125	0.01	0.05	0.5
DD-GEN	32	125	1	0.05	0.05
SMD	32	125	10	10	5
YELP	40	2	50	50	200
AGNEWS	20	4	1e-3	5	200

For optimization, we use Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for all the experiments. On all the datasets but 2D synthetic datasets and AGNews dataset, we use a batch size of 128 and a constant learning rate of $1e-3$ for encoder and decoder without weight decay. For LDEBM, we use a constant learning rate of $1e-4$. We use a larger batch size of 1000 on 2D synthetic datasets. On the AGNews dataset, we use the same set of hyperparameters as in Pang & Wu (2021) for optimization. The batch size is set to 200; the initial learning rate is $1e-4$ for encoder and decoder, and $1e-5$ for LDEBM. Learning rates are exponentially decayed with a decay rate of 0.998 for each model. Encoder and LDEBM have a weight decay rate of $2e-3$ and $1e-3$, respectively.

B.2. Experiment Settings and Baselines

Experiment settings For generative modeling, following previous methods (Shi et al., 2020; Pang & Wu, 2021), the NLL term is computed with importance sampling (Burda et al., 2016) using 500 importance samples. To compute rPPL, we set the generated sample size as $40,000$, which is the same size as PTB training set. We recruit ASGD Weight-Dropped LSTM (Merity et al., 2018) to compute rPPL as in previous works.

In terms of conditional response generation, for word-embedding-based evaluation on SMD and DD, we use the publicly available GloVe (Pennington et al., 2014) word embeddings of 300 dimension trained on 840B tokens, and report the score from 1 response per context. We use a context window size of 5 during training and evaluation.

The maximum length of each sentence is set to 40 words for most datasets and 70 words for the JerichoWorld dataset. On JerichoWorld dataset, we extract the description of each state as the text data.

Baselines On **PTB, DD** and **SMD**, our model is compared with the following baselines: (1) RNNLM (Mikolov et al., 2010), the language model implemented with GRU (Cho et al., 2014); (2) AE (Vincent et al., 2010), the deterministic auto-encoder which has no regularization to the latent space; (3) DAE, the AE with a discrete latent space; (4) VAE (Kingma & Welling, 2013), the vanilla VAE with a continuous latent space and a non-informative Gaussian prior; (5) DVAE, the VAE with a discrete latent space; (6) DI-VAE (Zhao et al., 2018b), a DVAE variant with a mutual information term between the observed piece of text $\mathbf{x}$ and its inferred latent variable $\mathbf{z}$; (7) semi-VAE (Kingma et al., 2014), the semi-supervised VAE model with independent discrete and continuous latent variables; (8) GM-VAE, the VAE with a Gaussian mixture prior; (9) DGM-VAE (Shi et al., 2020), the GM-VAE with a dispersion term that avoids the mode-collapse of Gaussian mixture prior; (10) semi-VAE $+\mathcal{I}(\mathbf{x},\mathbf{y})$, GM-VAE $+\mathcal{I}(\mathbf{x},\mathbf{y})$, DGM-VAE $+\mathcal{I}(\mathbf{x},\mathbf{y})$, are the same models as (7), (8), and (9) respectively, but with a mutual information term between $\mathbf{x}$ and $\mathbf{y}$ computed using separate inference networks for $\mathbf{y}$ and $\mathbf{z}$. We compare with the close competitors (11) SVEBM, the symbol-vector coupling prior model and (12) SVEBM-IB, SVEBM with regularization based on information-bottleneck.

On **Yelp** dataset, we additionally include text conditional GAN (Subramanian et al., 2018) as a baseline for controllable generation. On **AGNews** dataset, we further compare our model to VAMPIRE (Gururangan et al., 2019), a VAE-based semi-supervised text learning model. Other baselines include its supervised learning variants: (1) the model trained with Glove embedding pre-trained on 840 billion words (Glove-OD); (2) the model trained with Glove embedding on in-domain unlabeled data (Glove-ID). We also include more recent baselines such as Hard EM and CatVAE (Jin et al., 2020) that improve over VAMPIRE.

B.3. Extra Details for Experiments

More ablation study We conduct additional experiments on both PTB and DD datasets to inspect the contribution of the proposed techniques. In Sec. 4.1, we have reported results on PTB and datasets of OURS W/O GC which represents the model with Information Bottleneck but without Geometric Clustering (GC); OURS denotes the full model.

We further conduct experiments on the proposed model without using IB or GC. We observe that the proposed model using only diffusion-based sampling scheme has a rPPL of 166.26, BLEU of 11.30, wKL of 0.07 and NLL of 80.76 on PTB; it has a MI of 0.01, BLEU of 19.28, Act. of 0.12 and Emo. of 0.06 on DD, which is better than SVEBMs (please see Tabs. 1 and 3 in Sec. 4.1).

We also add GC to SVEBM (denoted as SVE-IB W/ GC). We find that SVE-IB W/ GC does perform better compared with SVE-IB, showing a rPPL of 179.95, BLEU of 10.08, wKL of 0.15 and NLL of 93.28 on PTB; it has a MI of 2.88, BLEU of 11.75, Act. of 0.61 and Emo. of 0.60 on DD. Notably, SVE-IB W/ GC is still inferior to LDEBMs.

In summary, we think these additional experiments (1) emphasize the importance of our diffusion-based modeling approach, and (2) demonstrate the effectiveness of GC as additional regularization.

2D synthetic data We provide the full evolution of SVEBM-IB and our models as visualized in Fig. A2. Though SVEBM-IB can capture some regularities of the data in the early stages of training, the model is prone to collapse due to the degenerated sampling quality. This features an exploding KL-term and leads to poor performance on generation. Our preliminary experiments indicate that common deep learning heuristics for improving the model capacity barely help. These include but are not limited to increasing the number of parameters in SVEBM, *i.e.*, using larger models, and adopting deliberately designed activation functions or normalization modules. LDEBM w/o geometric clustering has a better sampling quality and effectively mitigates the instability in training. However, the mode coverage is not satisfying in data space; the structure is unclear in latent space. In contrast, LDEBM w/ geometric clustering shows superior generation quality with better mode coverage. It demonstrates a better-structured latent space.

Sentence completion To perform sentence completion, we adopt a two-stage training scheme. We first train the LDEBM with inference, prior and generation models on the JerichoWorld dataset. After the first-stage training, the parameters of prior, inference and generation models are fixed. We then train a shallow MLP in the latent space to project the inferred posterior $\mathbf{z}_0$ to a disentangled space; the variables in the projected $\mathbf{z}_0$ can be grouped as: (a) the representation of observable words $\hat{\mathbf{z}}_{\text{obs}}$ in the input sentence and (b) the representation of unknown words $\hat{\mathbf{z}}_{\text{unk}}$. Conditional sampling in the latent space then refers to updating $\hat{\mathbf{z}}_{\text{unk}}$ based on the fixed $\hat{\mathbf{z}}_{\text{obs}}$ by running Langevin dynamics guided by the latent space model.

We mask half of the words in the sentences with `<unk>` token to prepare the inputs. In the second stage of training, we supervise the MLP by minimizing the reconstruction error between only the observable words of the input the sentence and the corresponding outputs of the model.

Sentence sentiment control Recall that in our formulation only $\mathbf{z}_0$ is connected to $\mathbf{y}$. We therefore condition only the final reverse diffusion step $[\mathbf{z}_0|\mathbf{z}_1]$ on $\mathbf{y}$ when performing controllable generation, *i.e.*, using $\mathbf{y}$ to guide the generation only when $t=0$ in Alg. 2. This can be a bit counter-intuitive since no label information is injected in previous reverse

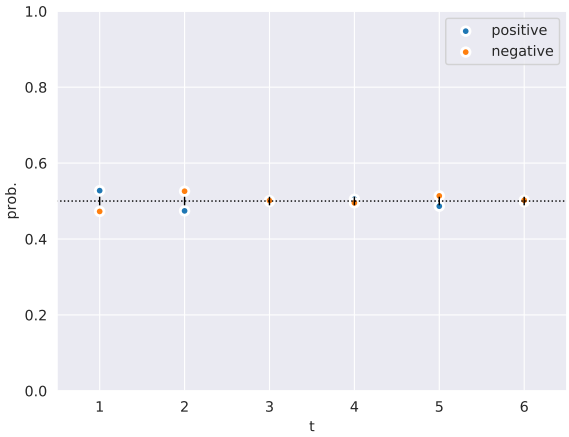


Figure A1. **Visualization of $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ over t.** $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ is constantly around the probability of 0.5 over t.

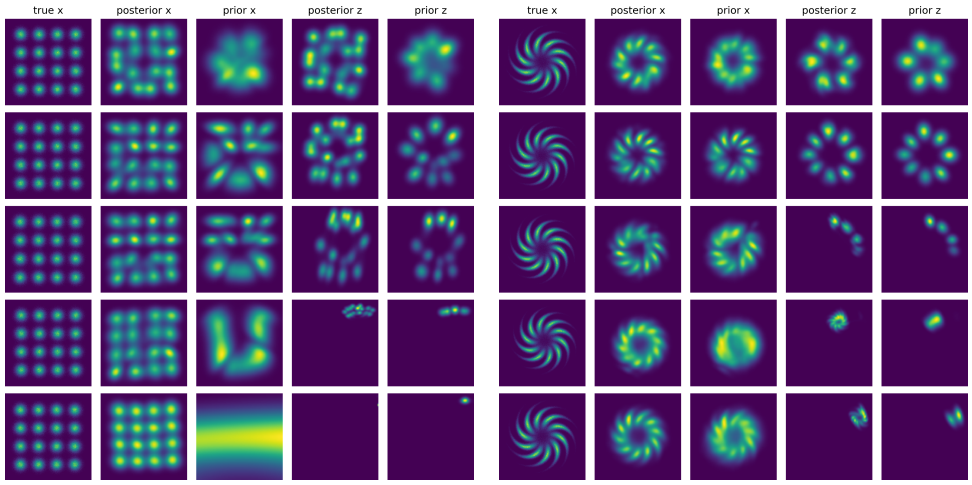
steps. Theoretically, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are independent given $\mathbf{z}_0$ in our formulation; however, we empirically observe that $\mathbf{y}$ and $\mathbf{z}_t$ for $t>0$ are nearly independent even marginally after we integrating out $\mathbf{z}_{0:t-1}$ in our model. In other words, $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ are in general non-informative since adding noise in the latent space could be much more corrupting than adding noise in the data space. The model learns to enjoy the less multi-modal energy landscape in previous reverse steps; it then seeks the given mode only in the most informative final reverse step. We examine $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ for the model trained on Yelp dataset by marginalizing out $\mathbf{z}_{t-1}$ of $p_\alpha(\mathbf{y},\mathbf{z}_{t-1}|\mathbf{z}_t),\ t>0$. For example, for $t=1$, we may calculate

$$\begin{aligned} p_\alpha(\mathbf{y}|\mathbf{z}_1) &= \int_{\mathbf{z}_0} p_\alpha(\mathbf{y}|\mathbf{z}_0)p_\alpha(\mathbf{z}_0|\mathbf{z}_1)d\mathbf{z}_0 \\ &= \mathbb{E}_{p_\alpha(\mathbf{z}_0|\mathbf{z}_1)}\left[p_\alpha(\mathbf{y}|\mathbf{z}_0)\right] \\ &\approx \frac{1}{M}\sum_{i=1}^{M} p_\alpha(\mathbf{y}|\mathbf{z}_0^{(i)}). \end{aligned} \tag{A22}$$

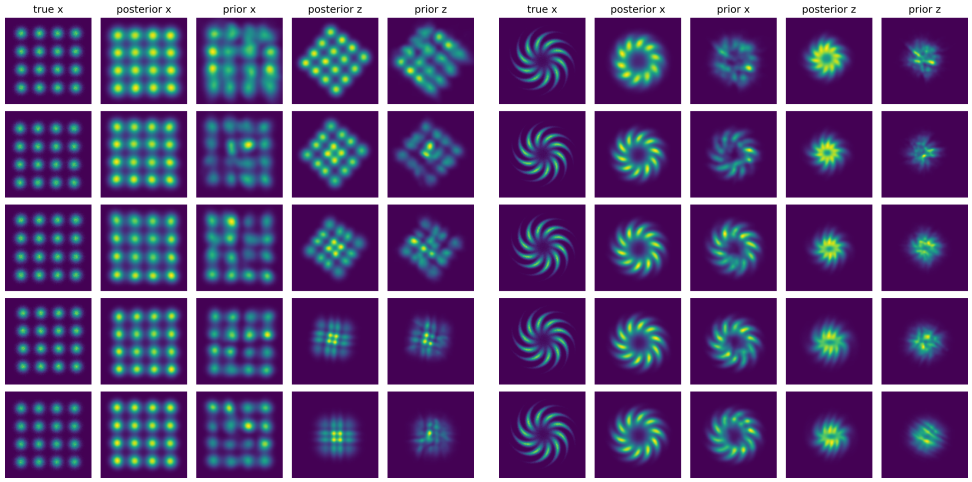
See Fig. A1 for the visualization of $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ over t.

A more intuitive method is to use the data label $\mathbf{y}$ to supervise each $[\mathbf{y},\mathbf{z}_t|\mathbf{z}_{t+1}]$, so that we can propagate the label information through the whole trajectory. Given $\mathbf{z}_0$, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are independent. But if we marginalize out $\mathbf{z}_0$, $\mathbf{y}$ will depend on $\mathbf{z}_1$. Similarly, if we continue to marginalize out $\mathbf{z}_1$, $\mathbf{y}$ will depend on $\mathbf{z}_2$. Repeating this process results in $p_\alpha(\mathbf{y}|\mathbf{z}_t)$ for each t after integrating out $\mathbf{z}_{0:t-1}$. Supervising $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ using $\mathbf{y}$ therefore effectively encodes the label information into the whole trajectory.

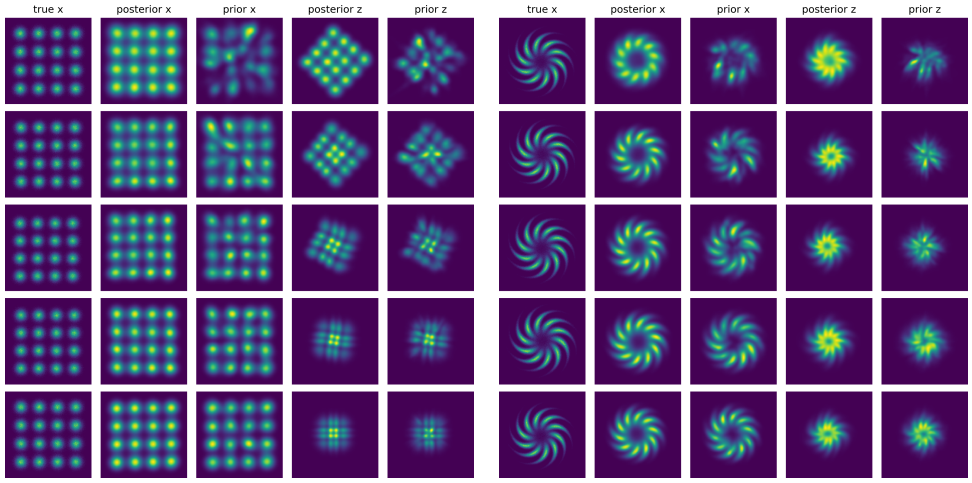
While the marginalization can be difficult, we may approximate it by learning the amortized version of $p_\alpha(\mathbf{y}|\mathbf{z}_t),\ t>0$ as $p_\alpha(\mathbf{y},\mathbf{z}_{t-1}=\mu_{\phi,t-1}|\mathbf{z}_t),\ t>0$, where $\mu_{\phi,t}$ is the posterior mean of $\mathbf{z}_t$. We may therefore circumvent the intractable integration in practice and guide the whole trajectory for controllable generation.



(a) SVEBM-IB Gaussian (b) SVEBM-IB Pinwheel



(c) LDEBM w/o PL Gaussian (d) LDEBM w/o PL Pinwheel



(e) LDEBM Gaussian (f) LDEBM Pinwheel

Figure A2. **Full evolution of SVEBM-IB and our models.** In each sub-figure, we provide the typical states of the model trained on the corresponding dataset, sequentially from the top row to the bottom row.