# Denoising Diffusion Variational Inference:
# Diffusion Models as Expressive Variational Posteriors

**Wasu Top Piriyakulkij** [* 1]  **Yingheng Wang** [* 1]  **Volodymyr Kuleshov** [1 2]

## Abstract

We propose denoising diffusion variational inference (DDVI), a black-box variational inference algorithm for latent variable models which relies on diffusion models as flexible approximate posteriors. Specifically, our method introduces an expressive class of diffusion-based variational posteriors that perform iterative refinement in latent space; we train these posteriors with a novel regularized evidence lower bound (ELBO) on the marginal likelihood inspired by the wake-sleep algorithm. Our method is easy to implement (it fits a regularized extension of the ELBO), is compatible with black-box variational inference, and outperforms alternative classes of approximate posteriors based on normalizing flows or adversarial networks. We find that DDVI improves inference and learning in deep latent variable models across common benchmarks as well as on a motivating task in biology—inferring latent ancestry from human genomes—where it outperforms strong baselines on the Thousand Genomes dataset.

## 1. Introduction

We are interested in amortized black-box variational inference problems of the form

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \max_{\phi} \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] \quad (1)$$
$$:= \max_{\phi} \text{ELBO}(\mathbf{x},\boldsymbol{\theta},\phi),$$

in which we approximate the marginal likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ of a latent variable model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ with an evidence lower bound $\text{ELBO}(\mathbf{x}, \boldsymbol{\theta}, \phi)$ that is a function of an approximate

variational posterior $q_{\phi}(\mathbf{z}|\mathbf{x})$. We assume that $p_{\boldsymbol{\theta}}$ factorizes as $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$ and admits efficient sampling: examples of such $p_{\boldsymbol{\theta}}$ include Bayesian networks, topic models (Blei et al., 2003), variational autoencoders (VAEs), and broad classes of $p_{\boldsymbol{\theta}}$ defined via modern probabilistic programming frameworks (Gordon et al., 2014).

Maximizing $\text{ELBO}(\mathbf{x}, \boldsymbol{\theta}, \phi)$ over $\phi$ yields a variational posterior $q_{\phi}(\mathbf{z}|\mathbf{x})$ that approximates $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ as well as a tight bound on $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ that serves as a learning objective for $p_{\boldsymbol{\theta}}$. The approximation gap $\log p_{\boldsymbol{\theta}}(\mathbf{x}) - \max_{\phi}\text{ELBO}(\mathbf{x},\boldsymbol{\theta},\phi)$ equals precisely $\min_{\phi}\text{KL}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$, which motivates the design of expressive classes of posteriors $q_{\phi}(\mathbf{z}|\mathbf{x})$ that reduce this gap. Recent efforts leverage modern generative models—including normalizing flows (Rezende and Mohamed, 2015; Kingma et al., 2016) and generative adversarial networks (Goodfellow et al., 2014; Makhzani et al., 2015)—as expressive model families for $q_{\phi}$ that tighten the ELBO.

This work seeks to further improve variational inference via expressive posteriors based on diffusion models (Ho et al., 2020; Song et al., 2020b). Diffusion methods have become the de-facto standard for high-quality image synthesis (Rombach et al., 2022). Here, we use diffusion in latent space to parameterize $q_{\phi}(\mathbf{z}|\mathbf{x})$. We train this distribution with a denoising diffusion-like objective that does not involve adversarial training (Makhzani et al., 2015) or constrained invertible normalizing flow architectures (Kingma et al., 2016). Samples from $q_{\phi}(\mathbf{z}|\mathbf{x})$ are obtained via iterative refinement of $\mathbf{z}$, starting from a Gaussian distribution, and gradually forming one that is multi-modal and complex.

Our work expands upon existing diffusion-based approximate inference methods (Berner et al., 2022; Zhang and Chen, 2021; Vargas et al., 2023; Zhang et al., 2023; Richter et al., 2023; Sendera et al., 2024; Akhound-Sadegh et al., 2024) that focus on the task of drawing samples from unnormalized distributions $\tilde{p}(\mathbf{z})$ and estimating the partition function $Z = \int_{\mathbf{z}} \tilde{p}(\mathbf{z})d\mathbf{z}$. While these methods are applicable in our setting—we set the unnormalized $\tilde{p}(\mathbf{z})$ to $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})$ such that $Z = p_{\boldsymbol{\theta}}(\mathbf{x})$—they do not make use of characteristics of $p_{\boldsymbol{\theta}}$ that are common in many types of models (VAEs, Bayes networks, etc.), namely the factorization $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$ and efficient sampling. We find that leveraging these properties yields simpler algorithms that avoid backpropagating through a sampling process, and that

[*]Equal contribution [1]Department of Computer Science, Cornell University [2]The Jacobs Technion-Cornell Institute, Cornell Tech. Correspondence to: Wasu Top Piriyakulkij<wp237@cornell.edu>, Yingheng Wang<yw2349@cornell.edu>, Volodymyr Kuleshov <kuleshov@cornell.edu>.

are fast enough to perform learning in addition to inference.

Specifically, we propose denoising diffusion variational inference (DDVI), an approximate inference algorithm defined by a class of approximate posterior distribution based on diffusion as well as a learning objective inspired by the wake-sleep algorithm (Hinton et al., 1995) that can be interpreted as a regularized variational inference. We also derive extensions of our method to semi-supervised learning and clustering.

Our method is easy to implement (it fits a regularized extension of the ELBO), is compatible with black-box variational inference, and outperforms alternative classes of approximate posteriors based on normalizing flows or adversarial networks. We evaluate DDVI on synthetic benchmarks and on a real problem in biological data analysis—inferring human ancestry from genetic data. Our method outperforms strong baselines on the Thousand Genomes dataset (Siva, 2008) and learns a low-dimensional latent space that preserves biologically meaningful structure (Haghverdi et al., 2015).

**Contributions.** In summary, this work introduces denoising diffusion variational inference, an approximate inference algorithm defined by two components: a class of approximate posteriors $q(\mathbf{z}|\mathbf{x})$ parameterized by diffusion, and a lower bound on the marginal likelihood inspired by wake-sleep. We complement DDVI with extensions to semi-supervised learning and clustering. Our method is especially suited for probabilistic programming, representation learning, and dimensionality reduction, where it outperforms alternative methods based on normalizing flows and adversarial training.

## 2. Background

**Deep Latent Variable Models** Latent variable models (LVMs) $p_{\boldsymbol\theta}(\mathbf{x}, \mathbf{z})$ are usually fit by optimizing the evidence lower bound (ELBO)

$$\log p_{\boldsymbol\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol\theta}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol\theta}(\mathbf{z})), \tag{2}$$

which serves as a tractable surrogate for the marginal log-likelihood (MLL). The gap between the MLL and the ELBO equals precisely $D_{\mathrm{KL}}(q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol\theta}(\mathbf{z}|\mathbf{x}))$—thus, a more expressive $q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})$ may better fit the true posterior and induce a tighter ELBO (Kingma and Welling, 2013).

Expressive variational posteriors can be formed by choosing more expressive model families—including auxiliary variable methods (Maaløe et al., 2016), MCMC-based methods (Salimans et al., 2015), normalizing flows (Rezende and Mohamed, 2015)—or improved learning objectives—e.g., adversarial or sample-based losses (Makhzani et al., 2015; Zhao et al., 2017; Si et al., 2022; 2023).

The wake-sleep algorithm (Hinton et al., 1995) optimizes an alternative objective

$$\mathbb{E}_{q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol\theta}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(p_{\boldsymbol\theta}(\mathbf{z}|\mathbf{x})||q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})), \tag{3}$$

in which the KL divergence term is reversed. The learning procedure for wake-sleep involves alternating between "wake" phases where the recognition model is updated and "sleep" phases where the generative model is refined.

**Denoising Diffusion Models** A diffusion model is defined via a user-specified noising process $q$ that maps data $\mathbf{x}_0$ into a sequence of $T$ variables $\mathbf{y}_{1:T} = \mathbf{y}_1, ..., \mathbf{y}_T$ that represent increasing levels of corruption to $\mathbf{x}_0$. We obtain $\mathbf{y}_{1:T}$ by applying a Markov chain $q(\mathbf{y}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{y}_t|\mathbf{y}_{t-1})$, where we define $\mathbf{y}_0 = \mathbf{x}_0$ for convenience. When $\mathbf{x}_0$ is a continuous vector, a standard choice of transition kernel is $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{y}_t; \sqrt{\alpha_t}\mathbf{y}_{t-1}, \sqrt{1-\alpha_t}\mathbf{I})$, which is a Gaussian centered around a copy of $\mathbf{y}_{t-1}$ to which we added noise following a schedule $0 < \alpha_1 < \alpha_2 < ... < \alpha_T = 1$.

A diffusion model can then be represented as a latent variable distribution $p(\mathbf{x}_0, \mathbf{y}_{1:T})$ that factorizes as $p(\mathbf{x}_0, \mathbf{y}_{1:T}) = p(\mathbf{y}_T)\prod_{t=0}^{T-1} p_{\boldsymbol\theta}(\mathbf{y}_t \mid \mathbf{y}_{t+1})$ (again using $\mathbf{y}_0$ as shorthand for $\mathbf{x}_0$). This model seeks to approximate the reverse of the forward diffusion $q$ and map noise $\mathbf{y}_T$ into data $\mathbf{x}_0$.

The true reverse of the process $q$ cannot be expressed in closed form; as such, we parameterize $p_{\boldsymbol\theta}$ with $\boldsymbol\theta$ trained by maximizing the ELBO:

$$\log p_{\boldsymbol\theta}(\mathbf{x}_0) \geq \mathbb{E}_q\left[\log p_{\boldsymbol\theta}(\mathbf{x}_0|\mathbf{x}_1) - \sum_{t=2}^{T} D_{\mathrm{KL}}(q_t||p_t)\right] \tag{4}$$
$$- D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))$$

where $q_t$, $p_t$ denote the distributions $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_{\boldsymbol\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, respectively.

## 3. Variational Inference With Denoising Diffusion Models

We introduce *denoising diffusion variational inference (DDVI)*, which improves variational inference with diffusion-based techniques. The goal of DDVI is to fit a latent variable model $p_{\boldsymbol\theta}(\mathbf{x}, \mathbf{z})$. We assume that $p_{\boldsymbol\theta}$ factorizes as $p_{\boldsymbol\theta}(\mathbf{x}|\mathbf{z})p_{\boldsymbol\theta}(\mathbf{z})$ and admits efficient sampling: examples of such $p_{\boldsymbol\theta}$ include Bayesian networks and variational autoencoders (VAEs) (Kingma and Welling, 2013). Our approach is comprised of two key components: a family of approximate posteriors $q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})$ based on diffusion and a learning objective that can be seen as a regularized ELBO. The $q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})$ iteratively refines latents $\mathbf{z}$, starting from a Gaussian distribution. The learning objective trains $q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x})$ to reverse a used-specified forward diffusion process.

### 3.1. Diffusion-Based Variational Posteriors

Specifically, DDVI performs variational inference using an approximate posterior $q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{x}) = \int_{\mathbf{y}} q_{\boldsymbol\phi}(\mathbf{z}|\mathbf{y}, \mathbf{x})q_{\boldsymbol\phi}(\mathbf{y}|\mathbf{x})d\mathbf{y}$,

*Figure 1.* Denoising diffusion variational inference in a VAE. Between the encoder and decoder, we have a diffusion model to map a simple distribution into a complex distribution over latents.

which itself contains latent variables $\mathbf{y} \in \mathcal{Y}$. The models $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y},\mathbf{x}), q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x})$ must have tractable densities and support gradient-based optimization over $\boldsymbol{\phi}$.

We choose the latent $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T)$ to be a vector of $T$ variables that represent progressively simplified versions of $\mathbf{z}$, with $\mathbf{y}_T$ corresponding to a simple distribution (e.g., a Gaussian). The model $q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y}_1,\mathbf{x}) \prod_{t=1}^{T-1} q_{\boldsymbol{\phi}}(\mathbf{y}_t|\mathbf{y}_{t+1},\mathbf{x})$ transforms $\mathbf{y}_T$ into $\mathbf{z}$ via iterative refinement. To sample from $q_{\boldsymbol{\phi}}$, we first sample $\mathbf{y}_T$—this is an easier task since we can define $\mathbf{y}_T$ to have a simple (e.g., Gaussian) distribution—and then by sampling from the denoising model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y}_1,\mathbf{x}) \prod_{t=1}^{T-1} q_{\boldsymbol{\phi}}(\mathbf{y}_t|\mathbf{y}_{t+1},\mathbf{x})$.

We define the relationship between $\mathbf{y}$ and $\mathbf{z}$ via a forward diffusion process $r(\mathbf{y}|\mathbf{z}) = r(\mathbf{y}_{1:T}|\mathbf{z}) = r(\mathbf{y}_1|\mathbf{z}) \prod_{t=1}^{T-1} r(\mathbf{y}_{t+1}|\mathbf{y}_t)$, which transforms $\mathbf{z}$—the latent whose intractable posterior we seek to approximate—into $\mathbf{y}_T$, whose posterior is easier to model. Examples of $r$ include Gaussian forward diffusion processes and discrete noising processes (Austin et al., 2021). The model $q_{\boldsymbol{\phi}}$ is trained to approximately reverse this forward diffusion process.

### 3.2. The Wake-Sleep Regularized ELBO

The standard approach to fit auxiliary-variable generative models (Maaløe et al., 2016) is to apply the ELBO twice:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \quad (5)$$

$$\geq \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \quad (6)$$
$$- \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x},\mathbf{z})\|r(\mathbf{y}|\mathbf{x},\mathbf{z}))]$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \quad (7)$$
$$- D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})\|r(\mathbf{y}|\mathbf{x},\mathbf{z})p(\mathbf{z}))$$

In Equation (5), we applied the ELBO over $\mathbf{z}$, and in Equation (6) we applied the ELBO again over

the latent $\mathbf{y}$ of $q$. Notice that the gap between the ELBO and $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ is $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x},\mathbf{z})\|r(\mathbf{y}|\mathbf{x},\mathbf{z}))]$. Thus, if we correctly match $q$ and $r$, we will achieve a tight bound.

**Wake-Sleep Regularization** Notice that optimizing $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})\|r(\mathbf{y}|\mathbf{z},\mathbf{x})p(\mathbf{z}))$ in Equation 7 involves sampling from the approximate reverse process $q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})$ to match the true reverse process $r(\mathbf{y}|\mathbf{z},\mathbf{x})$: this is the opposite of diffusion training, where we would sample from $r$ to fit $q$. We have found the standard ELBO to insufficient to learn a good $q_{\boldsymbol{\phi}}$ that reverses the noising process $r$, as illustrated in Table 9.

Instead, our strategy is to introduce a modified ELBO that yields diffusion-like training. This objective is the ELBO in Equation 7 augmented with an additional regularizer $\mathcal{L}_{\mathrm{sleep}}(\boldsymbol{\phi})$.

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{wake / recons. term } \mathcal{L}_{\mathrm{rec}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})} \quad (8)$$
$$\underbrace{- D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})\|r(\mathbf{y}|\mathbf{x},\mathbf{z})p(\mathbf{z}))}_{\text{prior regularization term } \mathcal{L}_{\mathrm{reg}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})}$$
$$\underbrace{- \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[D_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\|q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))]}_{\text{sleep term } \mathcal{L}_{\mathrm{sleep}}(\boldsymbol{\phi})}$$

The optimization of the regularizer $\mathcal{L}_{\mathrm{sleep}}(\boldsymbol{\phi})$ is similar to the sleep phase of the wake-sleep algorithm, and closely resembles diffusion model training (see below). As in wake-sleep, $\mathcal{L}_{\mathrm{sleep}}$ is optimized over $\boldsymbol{\phi}$ only, and the $\mathbf{x}$ are sampled from the model.

### 3.3. Optimizing the Regularized ELBO

Computing $\mathcal{L}_{\mathrm{sleep}}(\boldsymbol{\phi})$ still involves intractable distributions $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}), q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. To optimize $\mathcal{L}_{\mathrm{sleep}}(\boldsymbol{\phi})$, we introduce another lower bound $\mathcal{L}_{\mathrm{diff}}(\boldsymbol{\phi})$, which we call the denoising diffusion

loss (for reasons that will become apparent shortly):

$$\mathcal{L}_{\text{sleep}}(\boldsymbol{\phi}) = -\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}[D_{\text{KL}}(p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))] \qquad (9)$$

$$= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] + \bar{H}(p_{\boldsymbol{\theta}})$$

$$\geq \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}[\mathbb{E}_r[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})}{r(\mathbf{y}|\mathbf{z},\mathbf{x})}] + \bar{H}(p_{\boldsymbol{\theta}}) = \mathcal{L}_{\text{diff}}(\boldsymbol{\phi})$$

In Equation (10), we applied the ELBO with $r(\mathbf{y}|\mathbf{z},\mathbf{x})$ playing the role of the variational posterior over the latent $\mathbf{y}$ in $q_{\boldsymbol{\phi}}$; $\bar{H}(p_{\boldsymbol{\theta}})$ is the expected conditional entropy of $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, a constant that does not depend on $\boldsymbol{\phi}$.

We define the DDVI learning objective $\mathcal{L}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})$ as the sum of the aforementioned terms:

$$\mathcal{L}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi}) = \mathcal{L}_{\text{rec}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi}) + \mathcal{L}_{\text{reg}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi}) + \mathcal{L}_{\text{diff}}(\boldsymbol{\phi}) \qquad (10)$$

Terms $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{diff}}$ may be weighted by hyper-parameters $\beta_{\text{reg}}, \beta_{\text{diff}} > 0$, as in the $\beta$-VAE framework. In our experiments, assume $\beta_{\text{reg}} = \beta_{\text{diff}} = 1$ unless otherwise specified. Note that since $\mathcal{L}_{\text{diff}} \leq \mathcal{L}_{\text{sleep}} \leq 0$, $\mathcal{L}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})$ is a valid lower bound on $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ that is tight when $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

**Wake-Sleep Optimization** We optimize our bound on $\mathcal{L}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})$ using gradient descent by alternating between ELBO optimization and taking sleep steps (see Section 5). Note that by maximizing $\mathcal{L}_{\text{diff}}$, we fit $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ via the forward KL divergence; similarly, by optimizing $\mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{reg}}$ (the ELBO), we fit $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ via the reverse KL divergence. Thus, optimizing $\mathcal{L}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})$ encourages $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to approximate $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, and when the two are equal, the bound on $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ is tight.

**Simplifying Wake-Sleep** We also consider a light-weight algorithm, in which $r(\mathbf{y}|\mathbf{z})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y})$ do not depend on $\mathbf{x}$. In this case, $\mathcal{L}_{\text{diff}}$ requires only sampling from $p(\mathbf{z})$, and the entire loss $\mathcal{L}$ can be optimized end-to-end using gradient descent. This algorithm is a simpler (there is no separate sleep phase); however, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ may not perfectly approximate $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, hence $\mathcal{L}$ may no longer be a tight bound. We report results in Appendix B.

### 3.4. Combining The Wake-Sleep Regularized ELBO With Diffusion Models

Lastly, we can further simplify $\mathcal{L}_{\text{diff}}$ by leveraging the Markov structure of the forward and reverse processes $r, q$. Recall that each $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T)$ can be a vector of $T$ latents, which we also denote as $\mathbf{y}_{1:T}$, and that $r(\mathbf{y}_{1:T}|\mathbf{z}) = \prod_{t=1}^{T} r(\mathbf{y}_t|\mathbf{y}_{t-1})$, where $\mathbf{y}_0 = \mathbf{z}$ and the $\mathbf{y}_{1:T}$ are increasingly noised versions of $\mathbf{y}_0$. Similarly, $q_{\boldsymbol{\phi}}(\mathbf{y}, \mathbf{z}|\mathbf{x})$ is an approximate reverse diffusion process $q_{\boldsymbol{\phi}}(\mathbf{y}, \mathbf{z}|\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{y}_{0:T}|\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{y}_T|\mathbf{x})\prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\mathbf{y}_{t-1}|\mathbf{y}_t,\mathbf{x})$ that is trained to reverse $r$.

In order to fit $q_{\boldsymbol{\phi}}$, we form a lower bound $\mathcal{L}_{\text{diff}}(\mathbf{x},\boldsymbol{\phi})$ on the sleep term $\mathbb{E}_{p(\mathbf{x},\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ in Equation (10). This bound is identical to the ELBO of a diffusion model, and has the same derivation.

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_r\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y}_1,\mathbf{x}) - \sum_{t=2}^{T} D_{\text{KL}}(r_t||q_t)\right] \qquad (11)$$
$$- D_{\text{KL}}(r(\mathbf{y}_T|\mathbf{z})||q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x})).$$

where $r_t$, $q_t$ denote the distributions $r(\mathbf{y}_{t-1}|\mathbf{y}_t, \mathbf{y}_0)$ and $q_{\boldsymbol{\phi}}(\mathbf{y}_{t-1}|\mathbf{y}_t,\mathbf{x})$.

**Parameterizing Diffusion-Based Encoders** A common type of noising process compatible with this bound when $\mathbf{z}$ is continuous is Gaussian diffusion, where we define $r(\mathbf{y}_t|\mathbf{y}_{t-1}) = \mathcal{N}(\mathbf{y}_t; \sqrt{1-\alpha_t}\mathbf{y}_{t-1}, \alpha_t\mathbf{I})$ for a suitable schedule $(\alpha_t)_{t=1}^{T}$. We then adopt the parameterization $q_{\boldsymbol{\phi}}(\mathbf{y}_{t-1}|\mathbf{y}_t,\mathbf{x}) = \mathcal{N}(\mathbf{y}_{t-1}; \mu_{\boldsymbol{\phi}}(\mathbf{y}_t,\mathbf{x},t), \Sigma_{\boldsymbol{\phi}}(\mathbf{y}_t,\mathbf{x},t))$. It is then common to parameterize $q_{\boldsymbol{\phi}}$ with a noise prediction network $\epsilon_{\boldsymbol{\phi}}$ (Ho et al., 2020); the sum of KL divergences can be approximated by $\mathbb{E}_{t,\epsilon_t \sim r(\mathbf{y}_0,t)}||\epsilon_t - \epsilon_{\boldsymbol{\phi}}(\sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t,\mathbf{x},t)||^2$.

Lastly, in order to use diffusion-based encoders with our ELBO-based objective, we need to show that we can tractably compute $-D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})||r(\mathbf{y}|\mathbf{x},\mathbf{z})p(\mathbf{z}))$, which we equivalently rewrite as:

$$-D_{\text{KL}}(q_{\boldsymbol{\phi}}||r \cdot p) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log(r(\mathbf{y}|\mathbf{x},\mathbf{z})p(\mathbf{z}))] + H(q). \qquad (12)$$

The term $H(q)$ denotes the entropy. We approximate the first term using Monte-Carlo; the entropy is computed as

$$H(q) = -\sum_{t=1}^{T+1} \mathbb{E}_q[\log q_{\boldsymbol{\phi}}(\mathbf{y}_{t-1}|\mathbf{y}_t,\mathbf{x})] \qquad (13)$$
$$= \sum_{t=1}^{T+1} \mathbb{E}_q\left[\frac{d}{2}(1+\log(2\pi)) + \frac{1}{2}\log|\Sigma_{\boldsymbol{\phi}}(\mathbf{y}_t,\mathbf{x})|\right]$$

where $d$ is the dimension of $\mathbf{y}$ and we use the notation $\mathbf{y}_{T+1} = \mathbf{x}$. The right-hand term can be approximated using Monte Carlo; it is also common to leave the variance $\Sigma_{\boldsymbol{\phi}}$ fixed (as we typically do in our experiments), in which case $H(q)$ is a constant.

## 4. Extensions

### 4.1. Semi-Supervised Learning

Following Makhzani et al. (2015), we extend our algorithm to the semi-supervised learning setting where some data points have labels denoted by $l$. We assume the user provides a model of the form $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y},\mathbf{z},l) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},l)r(\mathbf{y}|\mathbf{z},l)p_{\boldsymbol{\theta}}(\mathbf{z}|l)p(l)$; we set the variational distributions to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{y},l), q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x}), q_{\boldsymbol{\phi}}(l|\mathbf{x})$. In this setting, we consider two cases, depending on whether the label is observed (Kingma et al., 2014). We extend Equation (9) to

incorporate the label $l$ corresponding to a data point as follows:

$$\mathcal{L}_{\text{semi}} = \mathbb{E}_{q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x},l)}[\log p_\theta(\mathbf{x}|\mathbf{z},l)] \qquad (14)$$
$$- D_{\text{KL}}(q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x},l)||p_\theta(\mathbf{y},\mathbf{z}|l))$$
$$- \mathbb{E}_{p_\theta(\mathbf{x})}[D_{\text{KL}}(p_\theta(\mathbf{z}|\mathbf{x},l)||q_\phi(\mathbf{z}|\mathbf{x},l))]$$

When the label $c$ cannot be observed, we treat it as a latent variable and modify the learning objective $\mathcal{U}_{\text{semi}} = \sum_c q_\phi(l|\mathbf{x})\mathcal{L}_{\text{semi}}(\mathbf{x},l,\theta,\phi) + D_{\text{KL}}(q_\phi(l|\mathbf{x})||p(l))$. Therefore, we can conclude a marginal likelihood on our dataset as follows: $\tilde{\mathcal{L}}_{\text{semi}} = \sum_{(\mathbf{x},l)\in L}\mathcal{L}_{\text{semi}}(\mathbf{x},l,\theta,\phi) + \sum_{\mathbf{x}\in U}\mathcal{U}_{\text{semi}}(\mathbf{x},\theta,\phi)$. where $L$ and $U$ are the sets of data with and without labels, respectively.

We also want to guarantee that all model parameters can be learned in all cases, including $q_\phi(l|\mathbf{x})$, such that this posterior can be applied as a classifier during inference. Thus, we combine the marginal likelihood with a classification loss to form an extended learning objective: $\tilde{\mathcal{L}}_{\text{semi}_\alpha} = \tilde{\mathcal{L}}_{\text{semi}} + \alpha \cdot \mathbb{E}_{\tilde{p}(\mathbf{x},l)}[-\log q_\phi(l|\mathbf{x})]$

### 4.2. Clustering

We have further extended our algorithm to encompass the clustering paradigm. We propose two distinct strategies. In the first approach, we simply fir a model in which $p_\theta(\mathbf{z})$ as a mixture of desired priors. The means of these priors are characterized by $\theta$. From these means, cluster membership, denoted as $\mathbf{c}$ can be deduced. This approach requires no alteration to the existing learning objective.

Alternatively, the second method retains the original prior but introduces an additional cluster latent variable $\mathbf{c}$ where $\sum_i c_i = 1$. Thus, the model can be specified as $p_\theta(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{c}) = p_\theta(\mathbf{x}|\mathbf{z},\mathbf{c})r(\mathbf{y}|\mathbf{z})p_\theta(\mathbf{z})p(\mathbf{c})$ with $p(\mathbf{c}) = Dir(\epsilon)$. Consequently, the variational distributions become $q_\phi(\mathbf{z}|\mathbf{y},\mathbf{c},\mathbf{x}), q_\phi(\mathbf{y},\mathbf{c}|\mathbf{x})$. This reformulates the learning objective as:

$$\mathcal{L}_{\text{clus}}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{y},\mathbf{z},\mathbf{c}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{c})] \qquad (15)$$
$$- D_{\text{KL}}(q_\phi(\mathbf{y},\mathbf{z},\mathbf{c}|\mathbf{x})||p_\theta(\mathbf{y},\mathbf{z},\mathbf{c}))$$
$$- \mathbb{E}_{p_\theta(\mathbf{x})}[D_{\text{KL}}(p_\theta(\mathbf{z}|\mathbf{x})||q_\phi(\mathbf{z}|\mathbf{x}))]$$

## 5. Experiments

We compare DDVI with Auto-Encoding Variational Bayes (AEVB) (Kingma and Welling, 2013), AEVB with inverse autoregressive flow posteriors (AEVB-IAF) (Kingma et al., 2016), Adversarial Auto-Encoding Bayes (AAEB) (Makhzani et al., 2015), and Path Integral Sampler (PIS) (Zhang and Chen, 2021) on MNIST (Lecun et al., 1998) and CIFAR-10 (Krizhevsky and Hinton, 2009) in unsupervised and semi-supervised learning settings, and also on the Thousand Genomes dataset (Siva, 2008). We also compare with Hierachical Auto-Encoding Variational

Bayes (H-AEVB) (Ranganath et al., 2016; Vahdat and Kautz, 2020) in unsupervised setting. We discuss the computational costs of all methods in Appendix D. The priors, model architecture, and training details can be founded in Appendix F, Appendix G, and Appendix H respectively. All results below are reported with 95% confidence interval using 3 different seeds.

### 5.1. Unsupervised learning

We start with synthetic experiments that are aimed at benchmarking the expressivity of diffusion-based posteriors and their ability to improve fitting $p$, a distribution with a complex structured prior, like one might find in probabilistic programming, scientific analysis, or other applications. We fit a model $p_\theta(\mathbf{x},\mathbf{z})$ on the MNIST and CIFAR-10 datasets with three priors $p(\mathbf{z})$: pinwheel, swiss roll, and square and report our results in Table 1 and Table 7. The model distribution $p_\theta$ is instantiated by a deep Gaussian latent variable model (DGLVM) with multi-layer perceptrons (MLPs) on MNIST and convolutional neural networks (CNNs) on CIFAR-10. The details of model architecture are provided in Appendix G.

Our first set of metrics (ELBO and MMD) seeks to evaluate the learned generative model $p_\theta$ is good. In the ELBO calculation, we average the reconstruction loss across image pixels. We use MMD to measure sample quality: we generate images with the trained model and calculate MMD between the generated images and test images using a mixture of Gaussian kernel[1]. We only report MMD for MNIST, since CIFAR-10 generated samples are very low-quality for all methods because the latent dimension is 2.

Our last metric seeks to directly evaluate the expressivity of the posterior. We measure latent negative log-likelihood (Latent NLL) by fitting a kernel density estimator (KDE) on the latents produced by the model with test data as input and compute the log-likelihood of the latents sampled from the prior under the fitted KDE.

From Tables 1 and 7, we see our method DDVI achieve best ELBO in all but one scenario, in which it still performs competitively. We also see strong results in Latent NLL and Acc in many scenarios, except for Swiss Roll where AAEB does well. We present visualizations of MNIST using the baseline methods and our method in Figure 2.

### 5.2. Semi-supervised Learning

We also evaluate the performance of our method and the baselines under semi-supervised learning setting where some labels are observed (1,000 for MNIST and 10,000 for CIFAR-10) and the partitions of the priors are known.

For this setting, we evaluate ELBO, latent negative log-likelihood (Latent NLL), and a k-nearest neighbors

---
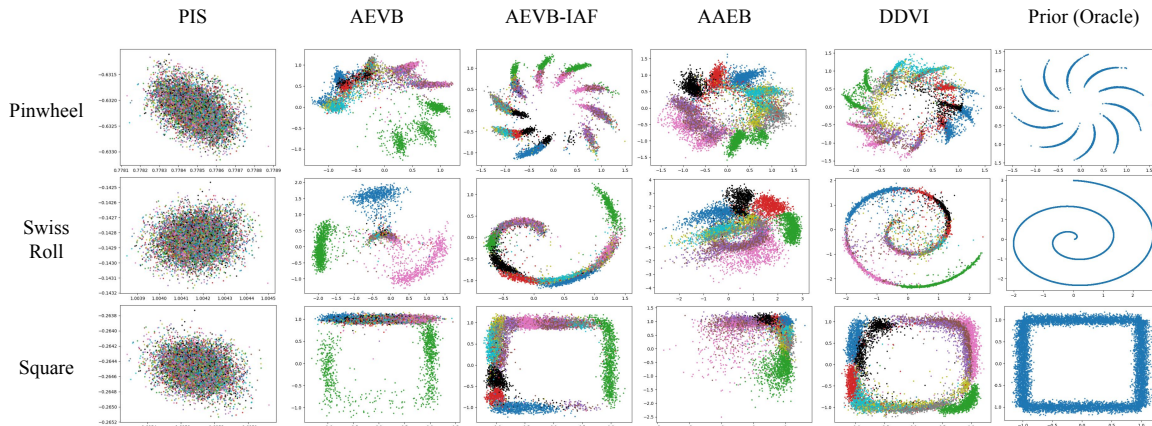
[1] with sigma equal to [2, 5, 10, 20, 40, 80]

*Figure 2.* Unsupervised visualization on MNIST using three different priors (pinwheel, swiss roll, and square). Each color indicates a class.

| Method | Pinwheel | | | Swiss Roll | | | Square | | |
|---|---|---|---|---|---|---|---|---|---|
| | ELBO | MMD | Latent NLL | ELBO | MMD | Latent NLL | ELBO | MMD | Latent NLL |
| AEVB | $-12.13\pm0.41$ | $0.77\pm0.04$ | $1.68\pm0.31$ | $-14.80\pm0.23$ | $0.78\pm0.17$ | $5.65\pm1.58$ | $-7.85\pm0.29$ | $1.10\pm0.66$ | $2.78\pm0.61$ |
| AEVB-IAF | $-4.19\pm0.05$ | $0.77\pm0.00$ | $1.64\pm0.73$ | $-5.10\pm0.30$ | $0.61\pm0.15$ | $4.43\pm1.09$ | $-3.97\pm0.22$ | $0.75\pm0.12$ | $1.68\pm0.27$ |
| AAEB | $N/A$ | $0.68\pm0.02$ | $1.54\pm0.19$ | $N/A$ | $\mathbf{0.52\pm0.03}$ | $3.34\pm0.16$ | $N/A$ | $0.80\pm0.02$ | $2.46\pm0.46$ |
| H-AEVB | $-7.03\pm3.13$ | $0.74\pm0.02$ | $2.25\pm3.02$ | $-7.21\pm4.62$ | $0.70\pm0.22$ | $4.04\pm4.62$ | $-5.71\pm3.05$ | $0.76\pm0.21$ | $2.22\pm2.03$ |
| PIS | $-7.83\pm0.64$ | $0.75\pm0.14$ | $6.50\pm1.11$ | $-9.83\pm0.61$ | $0.61\pm0.03$ | $\mathbf{2.40\pm1.01}$ | $-7.06\pm0.06$ | $0.77\pm0.04$ | $3.67\pm0.08$ |
| DDVI | $\mathbf{-3.88\pm0.96}$ | $\mathbf{0.67\pm0.04}$ | $\mathbf{1.27\pm0.21}$ | $\mathbf{-5.03\pm0.58}$ | $0.62\pm0.33$ | $3.86\pm0.17$ | $\mathbf{-3.79\pm0.14}$ | $\mathbf{0.66\pm0.07}$ | $\mathbf{1.56\pm0.09}$ |

*Table 1.* Unsupervised learning on MNIST. We report ELBO, MMD between generated images and test images, and latent negative log-likelihood (Latent NLL) with pinwheel, swiss roll, and square priors.

| Method | Pinwheel | | | Swiss Roll | | | Square | | |
|---|---|---|---|---|---|---|---|---|---|
| | ELBO | Acc | Latent NLL | ELBO | Acc | Latent NLL | ELBO | Acc | Latent NLL |
| AEVB | $-11.15\pm0.53$ | $0.93\pm0.01$ | $1.36\pm0.03$ | $-15.29\pm1.33$ | $0.68\pm0.01$ | $4.60\pm0.23$ | $-10.26\pm0.25$ | $0.86\pm0.01$ | $1.68\pm0.02$ |
| AEVB-IAF | $-2.10\pm0.26$ | $\mathbf{0.95\pm0.00}$ | $1.06\pm0.03$ | $-5.38\pm1.78$ | $0.90\pm0.02$ | $2.75\pm0.14$ | $-2.67\pm0.83$ | $\mathbf{0.91\pm0.01}$ | $\mathbf{0.90\pm0.02}$ |
| AAEB | $N/A$ | $0.89\pm0.01$ | $1.55\pm0.01$ | $N/A$ | $0.88\pm0.01$ | $3.07\pm0.05$ | $N/A$ | $1.94\pm0.38$ | $0.76\pm0.13$ |
| DDVI | $\mathbf{-0.24\pm0.13}$ | $\mathbf{0.95\pm0.00}$ | $\mathbf{1.06\pm0.01}$ | $\mathbf{-2.89\pm0.33}$ | $\mathbf{0.92\pm0.01}$ | $\mathbf{2.09\pm0.00}$ | $\mathbf{0.02\pm0.09}$ | $0.90\pm0.01$ | $1.49\pm0.03$ |

*Table 2.* Semi-supervised learning on MNIST (1,000 labels). We report ELBO, accuracy using KNN (K=20) classifier (Acc), and latent negative log-likelihood (Latent NLL) with pinwheel, swiss roll, and square priors.

classification accuracy of the latents (Acc). We choose classification accuracy since classification is a common downstream task for semi-supervised learning. We use the same set of priors and baselines. Details on how we partition each prior into $p_{\theta}(\mathbf{z}|\mathbf{x},l)$ can be founded in Appendix F. The partitions defined for our priors are local parts of the priors. We note that unlike unsupervised learning, we use the simplified sleep term from Appendix B in our objective for this setting, since $q_{\phi}$ already gets extra information from $l$ here.

The results are shown in Tables 2 and 8. DDVI mostly outperforms the baselines across different priors and metrics, especially on CIFAR-10 where DDVI is best across the board. For MNIST, DDVI always achieves the best ELBO, and it also performs competitively with other baselines in classification accuracy. We also show the visualizations of the latents in Figure 3 where DDVI matches the prior almost perfectly.

| Method | Cluster Purity | Cluster Completeness | NMI |
|---|---|---|---|
| AEVB | $0.28\pm0.02$ | $\mathbf{0.78\pm0.16}$ | $0.59\pm0.08$ |
| AEVB-IAF | $0.29\pm0.04$ | $\mathbf{0.73\pm0.06}$ | $0.55\pm0.06$ |
| AAEB | $0.37\pm0.06$ | $\mathbf{0.76\pm0.11}$ | $0.63\pm0.02$ |
| DDVI | $\mathbf{0.45\pm0.03}$ | $0.75\pm0.05$ | $\mathbf{0.66\pm0.04}$ |

*Table 3.* Quantitative genotype clustering results.

### 5.3. Clustering and Visualization for Genotype Analysis

In this section, we report results on an real-world task in genome analysis. Visualizing genotype data reveals patterns in the latent ancestry of individuals. We instantiate DDVI with a deep Gaussian latent variable model (DGLVM) and compare it against with the three strong clustering baselines using the 1000 Genomes dataset. We also report visualizations from three dimensionality reduction algorithms: PCA, TSNE, and UMAP.
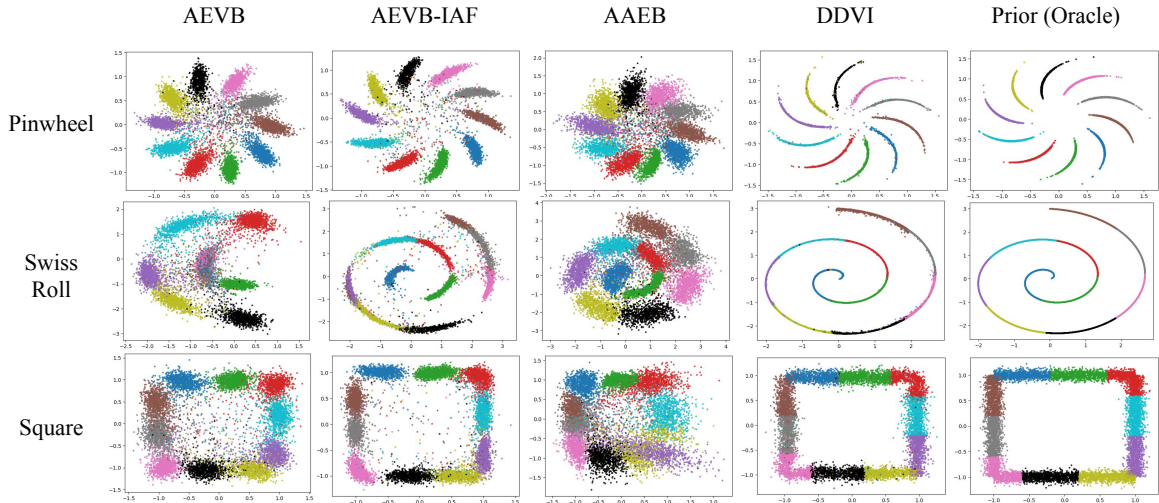
6

*Figure 3.* Semi-supervised visualization on MNIST with 1,000 labels using three different priors (pinwheel, swiss roll, and square). Each a indicates one class.
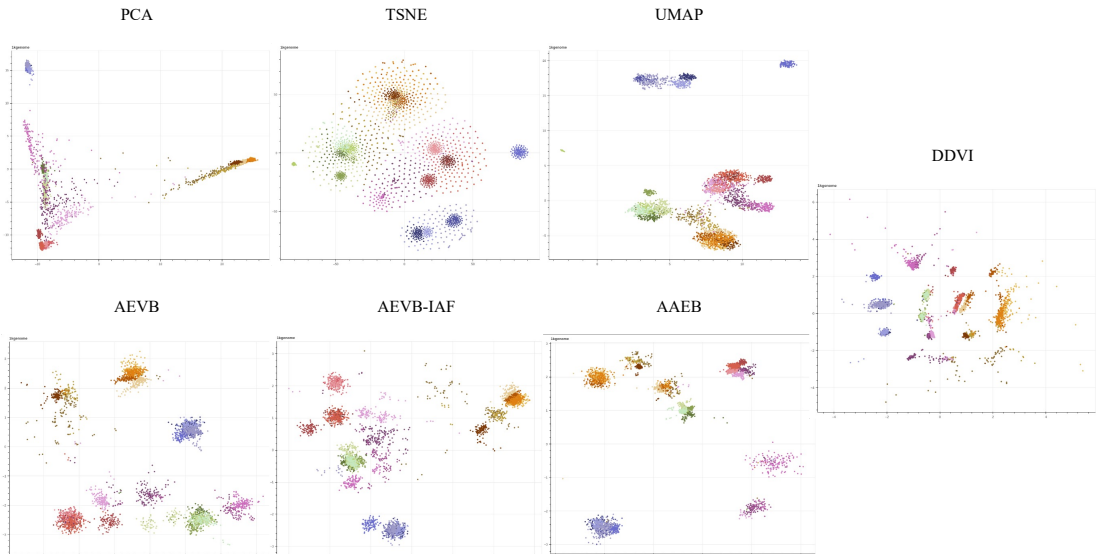


*Figure 4.* Visualization of genotype clusters. A color represents one ethnicity.

For each clustering algorithm, we seek to discover up to 20 clusters. We report quantitative results in terms of cluster purity, cluster completeness, and normalized mutual information (NMI). There is an inherent trade-off between cluster purity completeness. The overall clustering performance can be captured with NMI.

In Table 3, we see that DDVI attains the best performance on cluster purity and NMI. For cluster completeness, VAE and AAE have better means but much larger confidence interval. Furthermore, we visualize our genotype clustering results in latent space, shown in Figure 4, and also report results from classical dimensionality reduction and visualization methods that do not perform clustering (PCA (Wold et al., 1987), t-SNE

(Van der Maaten and Hinton, 2008), and UMAP (McInnes et al., 2018)). The legend of Figure 4 can be founded at Figure 5.

## 6. Discussion

**Diffusion vs. Normalizing Flows** Our approach is most similar to flow-based approximators (Rezende and Mohamed, 2015; Kingma et al., 2016); in fact when $T \rightarrow \infty$, our diffusion-based posterior effectively becomes a continuous-time normalizing flow (Song et al., 2020a). However, classical flow-based methods require invertible architectures for each flow layer: this constrains their expressivity and requires backpropagating through potentially a very deep network. Our approach, on the other hand, trains a model (a continuous-time flow when

$T \to \infty$) via a denoising objective (similar to score matching) that does not require invertible architectures and effectively admits an infinite number of layers (with weight sharing). This model is trained not by backpropagating through the ELBO, but rather via an auxiliary diffusion loss term (effectively, a score matching objective).

Despite training with a modified loss, we observe in Section 5 that a diffusion model with an expressive denoising architecture yields an improved ELBO relative to regular flows. Also, our modified loss based on the forward KL divergence reduces posterior collapse (i.e., all modes of the prior are covered well), and thus produces better samples.

**Diffusion vs. Other Generative Models** Variational posteriors based on GANs (Makhzani et al., 2015) also admit expressive architectures and require only sample-based access to the prior $p(\mathbf{z})$. Our diffusion-based approach admits a more stable loss, and is potentially more expressive, as it effectively supports an infinite number of layers (with shared parameters when $T \to \infty$). Unlike GANs, our models also admit explicit likelihoods and allow us to compute the ELBO for model evaluation. Our approach is similar to variational MCMC (Salimans et al., 2015); however, we train with a better objective augmented with a diffusion loss, and we adopt improved architectures with shared weights across layers.

**Diffusion for Approximate Inference** Existing diffusion-based approximate inference methods (Berner et al., 2022; Zhang and Chen, 2021; Vargas et al., 2023; Zhang et al., 2023; Richter et al., 2023; Sendera et al., 2024; Akhound-Sadegh et al., 2024) focus on the task of drawing samples from unnormalized distributions $\tilde{p}(\mathbf{z})$ and estimating the partition function $Z = \int_{\mathbf{z}} \tilde{p}(\mathbf{z})d\mathbf{z}$. While these methods are applicable in our setting— we set the unnormalized $\tilde{p}(\mathbf{z})$ to $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})$ such that $Z = p_{\boldsymbol{\theta}}(\mathbf{x})$— they also tackle a more challenging problem (drawing samples from energy-based models) in more general classes of models (arbitrary unnormalized distributions). In contrast, we focus on restricted but still important classes of models (VAEs, Bayes networks, etc.), and we solve more challenging sets of tasks (e.g., maximum-likelihood learning) by using properties of $p_{\boldsymbol{\theta}}$ (the factorization $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$ and efficient sampling from $p_{\boldsymbol{\theta}}$).

Our algorithms are also simpler. For example, diffusion sampling methods require backpropagating through a sampling process to minimize the reverse $\mathrm{KL}(q_{\boldsymbol{\phi}}\|p_{\boldsymbol{\theta}})$, which poses challenges with optimization and credit assignment. Some methods based on Schrodinger bridges require an iterative optimization process generalizing the sinkhorn algorithm or computationally expensive on-policy or off-policy (Malkin et al., 2022) trajectory-based optimization. In contrast, DDVI optimizes the forward $\mathrm{KL}(p_{\boldsymbol{\theta}}\|q_{\boldsymbol{\phi}})$ using simple gradient-based optimization that directly emulates diffusion-based training.

## 7. Related Work

**Latent Diffusion** Vahdat et al. (2021); Wehenkel and Louppe

(2021); Rombach et al. (2022) perform diffusion in the latent space of a VAE. Their goal is high sample quality, and they introduce into $p$ hierarchical latents with simple Gaussian priors. *Our goal is different*: we seek a method to fit a $p$ with structured latents (e.g., in probabilistic programming or in science applications, users introduce prior knowledge via hand-crafted $p$), and we improve variational inference in this structured model by introducing auxiliary latents into $q$. Recent work (Preechakul et al., 2022; Zhang et al., 2022; Wang et al., 2023) has also melded auto-encoders with diffusion models, focusing on semantically meaningful low-dimensional latents in a diffuser $p$. Cohen et al. (2022) crafts a diffusion bridge linking a continuous coded vector to a non-informative prior distribution.

**Diffusion for Approximate Inference** Diffusion sampling (Berner et al., 2022; Zhang and Chen, 2021; Vargas et al., 2023; Zhang et al., 2023; Richter et al., 2023; Sendera et al., 2024; Akhound-Sadegh et al., 2024) mainly focuses on the task of drawing samples from unnormalized distributions and estimating the partition function. These works draw connections between diffusion (learning the denoising process) and stochastic control (learning the Föllmer drift). Some other works (Zhang et al., 2023; Akhound-Sadegh et al., 2024; Sendera et al., 2024) use continuous generative flow networks (GFlowNets) – deep reinforcement learning algorithms adapted to variational inference that offers stable off-policy training and thus flexible exploration.

**Dimensionality Reduction** Latent variable models in general are an attractive alternative to visualization methods like PCA, UMAP, and t-SNE (McInnes et al., 2018; Van der Maaten and Hinton, 2008). Domain-specific knowledge can be injected through the prior, and deep neural networks can be utilized to achieve a more expressive mapping from the data space to the latent space. Nevertheless, downsides of LVMs are that they are more computationally expensive and require careful hyperparameter tuning.

## 8. Conclusion

While this paper focuses on applications of DDVI to dimensionality reduction and visualization, there exist other tasks for the algorithm, e.g., density estimation or sample quality. Accurate variational inference has the potential to improve downstream applications of generative modeling, e.g., decision making (Nguyen and Grover, 2022; Deshpande and Kuleshov, 2023), meta-learning (Rastogi et al., 2023), or causal effect estimation (Deshpande et al., 2022). Since our learning objective differs from the ELBO (it adds a regularizer), we anticipate gains on models whose training benefits from regularization, but perhaps not on all models. Also, attaining competitive likelihood estimation requires architecture improvements that are orthogonal to this paper. However, our ability to generate diverse samples and achieve class separation in latent space hints at the method's potential on these tasks.

# References

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. *arXiv preprint arXiv:2402.06121*, 2024.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Max Cohen, Guillaume Quispe, Sylvain Le Corff, Charles Ollion, and Eric Moulines. Diffusion bridges vector quantized variational autoencoders. *arXiv preprint arXiv:2202.04895*, 2022.

Shachi Deshpande and Volodymyr Kuleshov. Calibrated uncertainty estimation improves bayesian optimization, 2023.

Shachi Deshpande, Kaiwen Wang, Dhruv Sreenivas, Zheng Li, and Volodymyr Kuleshov. Deep multi-modal structural equations for causal effect estimation with unstructured proxies. *Advances in Neural Information Processing Systems*, 35:10931–10944, 2022.

Arnaud Doucet, Will Grathwohl, Alexander G Matthews, and Heiko Strathmann. Score-based diffusion meets annealed importance sampling. *Advances in Neural Information Processing Systems*, 35:21482–21494, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Future of software engineering proceedings*, pages 167–181. 2014.

Laleh Haghverdi, Florian Buettner, and Fabian J Theis. Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics*, 31(18):2989–2998, 2015.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep Robert Datta, and Ryan P. Adams. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems (NIPS) 29*, 2016. arXiv:1603.06277 [stat.ML].

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Toronto, ON, Canada*, 2009.

Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

Stephen Marsland. *Machine Learning: An Algorithmic Perspective (2nd Edition)*. Chapman and Hall/CRC, 2014.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022,*

*17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 16569–16594. PMLR, 2022. URL `https://proceedings.mlr.press/v162/nguyen22b.html`.

Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10619–10629, 2022.

Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

Richa Rastogi, Yair Schiff, Alon Hacohen, Zhaozhi Li, Ian Lee, Yuntian Deng, Mert R. Sabuncu, and Volodymyr Kuleshov. Semi-parametric inducing point networks and neural processes. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=FE99-fDrWd5`.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Lorenz Richter, Julius Berner, and Guan-Horng Liu. Improved sampling via learned diffusions. *arXiv preprint arXiv:2307.01198*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Tim Salimans, Diederik P Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International conference on machine learning*, pages 1218–1226. PMLR, 2015.

Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. On diffusion models for amortized inference: Benchmarking and improving stochastic control and sampling. *arXiv preprint arXiv:2402.05098*, 2024.

Phillip Si, Allan Bishop, and Volodymyr Kuleshov. Autoregressive quantile flows for predictive uncertainty estimation. In *International Conference on Learning Representations*, 2022.

Phillip Si, Zeyi Chen, Subham Sekhar Sahoo, Yair Schiff, and Volodymyr Kuleshov. Semi-autoregressive energy flows: Exploring likelihood-free training of normalizing flows. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett,

editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31732–31753. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/si23a.html`.

Nayanah Siva. 1000 genomes project. *Nature biotechnology*, 26(3):256–257, 2008.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

Yingheng Wang, Yair Schiff, Aaron Gokaslan, Weishen Pan, Fei Wang, Christopher De Sa, and Volodymyr Kuleshov. InfoDiffusion: Representation learning using information maximizing diffusion models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 36336–36354. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/wang23ah.html`.

Antoine Wehenkel and Gilles Louppe. Diffusion priors in variational autoencoders. *arXiv preprint arXiv:2106.15671*, 2021.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Dinghuai Zhang, Ricky Tian Qi Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *arXiv preprint arXiv:2310.02679*, 2023.

Guodong Zhang, Kyle Hsu, Jianing Li, Chelsea Finn, and Roger B Grosse. Differentiable annealed importance sampling and the perils of gradient noise. *Advances in Neural Information Processing Systems*, 34:19398–19410, 2021.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised representation learning from pre-trained diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 35:22117–22130, 2022.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

# Appendix

July 17, 2024

## A. Pseudocode

Here we provide a pseudocode to illustrate the training process of DDVI. We release anonymized code repo at `https://anonymous.4open.science/r/vi-with-diffusion-D70F`.

---
**Algorithm 1** DDVI Pseudocode

---
0: (Optional) Pre-train $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and $q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x})$ with DDVI but with unconditional diffusion model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y})$
0: **for** epoch $=1,...,n$ **do**
0:     **for** $\mathbf{x}_1,...,\mathbf{x}_k \sim p_{\mathcal{D}}(\mathbf{x})$ **do**
0:         $\mathbf{y}_i \sim q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x}_i)$ and $\mathbf{z}_i \sim q(\mathbf{z}|\mathbf{y}_i,\mathbf{x}_i)$ for $i=1,...,k$
0:         Optimize $\boldsymbol{\theta},\boldsymbol{\phi}$ with respect to a Monte Carlo estimate of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]-\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{y},\mathbf{z}))$ for each $\mathbf{x}_i${Standard ELBO training part}
0:         **for** iteration $=1,...,m$ **do**{Do sleep for $m$ iterations}
0:             $\mathbf{z}_1,...,\mathbf{z}_k \sim p(\mathbf{z})${Batch-sample latents from prior}
0:             $\hat{\mathbf{x}}_i \sim p(\mathbf{x}|\mathbf{z}_i)$ for $i=1,...,k${Construct fantasy inputs}
0:             $\mathbf{y}_i \sim r(\mathbf{y}|\mathbf{z}_i)$ for $i=1,...,k${Construct fantasy inputs}
0:             Optimize $\phi$ using the standard diffusion noise prediction loss on $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{y}_i,\hat{\mathbf{x}}_i)$
0:         **end for**
0:     **end for**
0: **end for**=0

---

## B. Simplifying Wake-Sleep

In wake-sleep, sampling $\mathbf{x}$ from $p_{\boldsymbol{\theta}}$ to obtain gradients for the sleep term introduces computational overhead. To address this issue, we propose *wake-sleep in latent space*, an algorithm that optimizes an approximation $\hat{\mathcal{L}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})$ of $\mathcal{L}$:

$$\hat{\mathcal{L}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi}) = \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{wake / reconstr. term } \mathcal{L}_{\text{rec}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})} \underbrace{-D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{y},\mathbf{z}))}_{\text{prior regularization term } \mathcal{L}_{\text{reg}}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})} \underbrace{-D_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\mathbf{z})\|q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))}_{\text{latent sleep term } \mathcal{L}_{\text{sleep}}(\mathbf{x},\boldsymbol{\phi})}. \quad (16)$$

We have replaced $\mathcal{L}_{\text{sleep}}(\phi)$ with a latent sleep term $\mathcal{L}_{\text{sleep}}(\mathbf{x},\phi)$, in which $\mathbf{x}$ is given, and we only seek to fit the true reverse noising process $r(\mathbf{z}|\mathbf{y})$ independently of $\mathbf{x}$. We can similarly show that

$$\mathcal{L}_{\text{sleep}}(\mathbf{x},\phi) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] + \bar{H}(p_{\boldsymbol{\theta}}) \geq \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})r(\mathbf{y}|\mathbf{z})}[\log(q_{\boldsymbol{\phi}}(\mathbf{y},\mathbf{z}|\mathbf{x})/r(\mathbf{y}|\mathbf{z}))] + \bar{H}(p_{\boldsymbol{\theta}}) \quad (17)$$

$$= -\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z})}[D_{\mathrm{KL}}(r(\mathbf{y}|\mathbf{z})\|q_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{z},\mathbf{x}))] - D_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\mathbf{z})\|q(\mathbf{z}|\mathbf{x})), \quad (18)$$

where $\bar{H}(p_{\boldsymbol{\theta}})$ is an entropy term constant in $\phi$. Thus, we minimize the forward KL divergence by sampling $\mathbf{z}$, and applying the noising process to get $\mathbf{y}$; the $q_{\phi}$ is fit to denoise $\mathbf{z}$ from $\mathbf{y}$ as in Equation (10).

We optimize our bound on $\hat{\mathcal{L}}(x,\boldsymbol{\theta},\boldsymbol{\phi})$ end-to-end using minibatch gradient descent over $\boldsymbol{\theta},\boldsymbol{\phi}$. While the wake term is a reconstruction loss as in wake-sleep, the sleep term generates latent samples $\mathbf{z},\mathbf{y}$ from $r(\mathbf{y}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$ (by analogy with $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$ in normal wake-sleep); the denoiser $q_{\phi}$ is trained to recover $\mathbf{z}$ from $\mathbf{y}$. Thus, we perform wake-sleep *in latent space*, which obviates the need for alternating wake and sleep phases, and allows efficient end-to-end training. A limitation of this approximation is that the sleep term does not fit $q_{\phi}$ to the true $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x},\mathbf{y})$, and as a consequence $\hat{L}$ is not a tight lower bound on $\log p_{\boldsymbol{\theta}}(\mathbf{x})$. We may think of $\mathcal{L}_{\text{sleep}}(\mathbf{x},\phi)$ as a regularizer to the ELBO.

## C. Comparision of Methods

We provide a comprehensive comparison of different methods in Table 4. Vahdat et al. (2021); Wehenkel and Louppe (2021); Rombach et al. (2022) perform diffusion in the latent space of a VAE to improve the efficiency of image generation. Their goal is high sample quality, and they introduce into $p$ hierarchical latents with simple Gaussian priors. **Our goal is different**: we seek a method to fit a $p$ with structured latents (e.g., in probabilistic programming or in science applications, users introduce prior knowledge via hand-crafted $p$), and we improve variational inference in this structured model by introducing auxiliary latents into $q$.

Recent work (Preechakul et al., 2022; Zhang et al., 2022; Wang et al., 2023) has also melded auto-encoders with diffusion models, focusing on semantically meaningful low-dimensional latents in a diffuser $p$. Cohen et al. (2022) crafts a diffusion bridge linking a continuous coded vector to a non-informative prior distribution.

| Model | Training Objective | Approximating Family | Sample-based Prior | Auxiliary Variable | Tasks | Simplified Graphical Illustration |
|---|---|---|---|---|---|---|
| AEVB | ELBO | Diagonal Gaussian | ✖ | ✖ | Density estimation | $\mathbf{x} \to \mathbf{z} \to \mathbf{x}$ |
| AEVB-IAF | ELBO | Normalizing flow | ✖ | ✔ | Density estimation / Visualization | $\mathbf{x} \to \mathbf{z}_0 \to \mathbf{z}_T \to \mathbf{x}$ |
| AAEB | Adversarial training | Adversarial generator | ✔ | ✖ | Visualization | $\mathbf{x} \to \mathbf{z} \to \mathbf{x}$ |
| H-AEVB-(IAF) | ELBO | Factorial Normal / Normalizing flow | ✖ | ✔ | Density estimation / High-quality sample generation | $\mathbf{x} \to \mathbf{z}_0 \to \mathbf{z}_T \to \mathbf{z}_0 \to \mathbf{x}$ |
| ADGM | ELBO | Non-Gaussian | ✖ | ✔ | Density estimation | $\mathbf{x} \to \mathbf{a} \to \mathbf{z} \to \mathbf{x}$ |
| LDM | ELBO | Diagonal Gaussian | ✖ | ✔ | High-quality sample generation | $\mathbf{x} \to \mathbf{z}_0 \to \mathbf{z}_T \to \mathbf{z}_0 \to \mathbf{x}$ |
| LSGM | ELBO & score matching | Diagonal Gaussian | ✖ | ✔ | High-quality sample generation | $\mathbf{x} \to \mathbf{z}_0 \to \mathbf{z}_T \to \mathbf{z}_0 \to \mathbf{x}$ |
| DDVI | ELBO & sleep term | Denoising diffusion | ✔ | ✔ | Density estimation / Visualization | $\mathbf{x} \to \mathbf{z}_T(\mathbf{y}) \to \mathbf{z}_0(\mathbf{z}) \to \mathbf{x}$ |

*Table 4.* Comparison of DDVI to other relevant methods. $\mathbf{x}$ represents the original data input to the model. $\mathbf{z}$ denotes the latent (hidden) representation of the input data. $\mathbf{a}$ represents an auxiliary variable introduced in some models (like ADGM) to capture additional aspects of the data distribution or to assist in the model's learning process.

## D. Computational Cost Analysis

| Method | NMI values at different wall-clock training times | | | | | |
|---|---|---|---|---|---|---|
| | NMI @ 10 min | NMI @ 20 min | NMI @ 30 min | NMI @ 40 min | NMI @ 50 min | NMI @ 60 min |
| AEVB | 0.52 | 0.52 | 0.52 | 0.52 | 0.52 | 0.52 |
| AEVB-IAF | 0.54 | 0.52 | 0.52 | 0.52 | 0.52 | 0.52 |
| AAEB | 0.61 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 |
| DDVI (T=5) | *warm up* | 0.63 | 0.63 | 0.66 | 0.66 | 0.66 |
| DDVI (T=10) | *warm up* | **0.64** | **0.68** | **0.70** | **0.70** | **0.70** |
| DDVI (T=20) | *warm up* | 0.50 | 0.51 | 0.56 | 0.64 | 0.68 |
| DDVI (T=50) | *warm up* | 0.52 | 0.54 | 0.51 | 0.59 | 0.59 |

*Table 5.* Computational cost trade-off on 1kgenome: NMI vs wall-clock training time

We conduct a computational cost analysis between the baselines and DDVI with various timesteps on the genotype clustering/visualization experiments. Table 5 shows that DDVI outperforms baselines at all timestamps and continues to improve after the baselines have plateaued.

# E. Connections to Diffusion Samplers

Diffusion sampling (Berner et al., 2022; Zhang and Chen, 2021; Vargas et al., 2023; Zhang et al., 2023; Richter et al., 2023; Sendera et al., 2024; Akhound-Sadegh et al., 2024) mainly focuses on the task of drawing samples from unnormalized distributions and estimating the partition function. These works draw connections between diffusion (learning the denoising process) and stochastic control (learning the Föllmer drift), leading to several approaches, e.g., path integral sampler (PIS) (Zhang and Chen, 2021), denoising diffusion sampler (DDS) (Vargas et al., 2023), and time-reversed diffusion sampler (DIS) (Berner et al., 2022), which have been unified by Richter et al. (2023). Some other works (Zhang et al., 2023; Akhound-Sadegh et al., 2024) use continuous generative flow networks (GFlowNets) – deep reinforcement learning algorithms adapted to variational inference that offers stable off-policy training and thus flexible exploration. Sendera et al. (2024) benchmarked these previous diffusion-structured amortized inference methods and studied how to improve credit assignment in diffusion samplers, which refers to the propagation of learning signals from the target density to the parameters of earlier sampling steps. Overall, there are indeed some strong connections between these works and ours:

- They also focus on variational methods that directly fit a parametric family of tractable distributions (given by controlled SDEs) to the target density.

- They cast the density estimation/sampling problem into an optimization problem over a control objective, which learns control drifts (and diffusion) parameterized by neural networks.

But we would like to clarify that there are also some clear differences between them:

- The diffusion-structured samplers only focus on density estimation/sampling but ignore the problem of learning a generative model, which is one of the main focuses of our work. We aim to perform more accurate variational inference using an auxiliary variable model augmented by diffusion models to improve generative modeling. In our setting, $p_{\boldsymbol{\theta}}(z|x)$ is a moving target density, as we jointly learn $\boldsymbol{\theta}$ with $\phi$, as opposed to a static target density that diffusion-structured samplers are designed to solve.

- To tackle the challenge of credit assignment – propagating weak learning signals through the sampling trajectory, the techniques proposed in diffusion-structured samplers are mostly based on partial trajectory information, which has higher training costs over on-policy (Zhang and Chen, 2021) or off-policy (Malkin et al., 2022) trajectory-based optimization. Instead, we introduce a wake-sleep optimization algorithm and its simplified version to alleviate the weak learning signal issue and optimize the evidence lower bound in a better way.

- In Equation (10), we are minimizing the forward KL divergence $\mathcal{D}_{KL}(p_{\boldsymbol{\theta}}||q_{\phi})$, where diffusion samplers are minimizing the reverse $\mathcal{D}_{KL}(q_{\phi}||p_{\boldsymbol{\theta}})$.

We also summarize the connections and differences in the table below.

# F. Priors

Below we describe the sampling process for each prior.

**Pinwheel.** This distribution was used in (Johnson et al., 2016). We define the number of clusters to be 10. For semi-supervised learning experiments, this prior is partitioned into 10 partitions, each partition being a cluster.

**Swiss Roll.** This distribution was used in (Marsland, 2014). For semi-supervised learning experiments, this prior is partitioned into 10 partitions. The samples from the prior can actually be characterized by a single scalar representing how far you are long the swiss roll from the center. The paritioning is done by creating 10 equal-length intervals in this 1D space.

**Square.** This distribution has the shaped of a square going from -1 to 1 in both axes. Each position on the square can be characterized by a single scalar representing how far you are from the top left corner. Sampling is done by sampling the position uniformly and turn the 1D position to 2D latent. We add noise $\sigma = 0.06$ to the prior. For semi-supervised learning experiments, this prior is partitioned into 10 partitions. The partitioning is done by creating 10 equal-length intervals in the 1D position space.

AEVB and AEVB-IAF requires that we can evaluate the prior density. To do this, for all priors, we evaluate the density by fitting a kernel density estimator with mixture of gaussian kernel with bandwidth equal to 0.005, 0.008, 0.01, 0.03, and 0.05.

| | Diffusion Samplers (Berner et al., 2022; Zhang and Chen, 2021; Vargas et al., 2023; Akhound-Sadegh et al., 2024) | GFlowNet-based (Zhang et al., 2023; Richter et al., 2023; Sendera et al., 2024; Malkin et al., 2022) | DDVI (ours) |
|---|---|---|---|
| Tasks | Sampling, density estimation | Sampling, density estimation | Learning, Sampling, Dimensionality reduction |
| Model Family for $p$ | Any energy-based | Any energy-based | Latent with tractable $p(x|z), p(z)$ |
| Model Family for $q$ | Markov chain | Markov chain | Markov chain |
| Objective | $D_{KL}(q\|p)$ with regularizer | Trajectory balance objective | $D_{KL}(p\|q)$ with ELBO |
| Algorithm | Gradient descent (with reference process), importance sampling | RL-motivated off-policy optimization (replay buffers, Thompson sampling, etc.) | Gradient descent with wake-sleep |
| Compatible Models | Anything energy-based | Anything energy-based | LDA, deep latent-variable models |
| Applications | Sampling from physics-based models, model selection based on NLL | Sampling from physics-based models, model selection based on NLL | Probabilistic programming, visualization |

*Table 6.* Comparison of Diffusion-structured Samplers, GFlowNet-based Approaches, and DDVI

## G. Model Architecture

All methods use the same architecture for encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$, excluding the extra parts specific to each method which we describe below, for the same dataset. For MNIST, the encoder and decoder are multi-layer perceptron with two hidden layers, each with 1000 hidden units. For CIFAR-10, the encoder is a 4-layer convolutional neural network with (16, 32, 64, 128) channels with a linear layer on top, and the decoder is a 4-layer tranposed convolutional neural network with (64, 32, 16, 3) channels where a linear layer is used to first turn the feature dimension from 2 to 64.

**AEVB-IAF** employs 4 IAF transformations on top of the encoder, each is implemented with a 4-layer MADE. The number of hidden units in MADE is 128. The ordering is reversed between every other IAF transformation.

**AAEB** has a discriminator, used in adversarial training, which is a multi-layer perceptron with two hidden layers, each with 1000 hidden units.

**DDVI** has a diffusion model on top of the encoder. The time-conditioned reverse diffusion distribution is implemented with a 5-layer time-conditioned multi-layer perceptron, each with 128 hidden units. A time-conditioned linear layer learns an additional embedding for each timestep and adds it to the output of the linear layer.

## H. Training Details

For training, we update the parameters for each batch of inputs by alternating between the ELBO phase (optimizing $\theta$ and $\phi$ with respect to the ELBO, i.e., the reconstruction term and the prior matching term) and the sleep phase (optimizing $\phi$ with respect to the sleep term). We use Adam optimizer and latent size of 2 for all of our experiments. Each algorithm takes roughly 2 hours on a single Nvidia GeForce RTX 3090 to complete one run of experiment. The training details of each algorithm are detailed below:

**AEVB.** The batch size is set to 128. The number of epochs is 200 for unsupervised and clustering experiments and 50 for semi-supervised experiments. The learning rate is 0.0001. The loss is BCE for MNIST and CIFAR-10 experiments and MSE for genotype analysis experiments. For semi-supervised MNIST experiments, the kl divergence weight is set to be 0.01, while for semi-supervised CIFAR-10 experiments, the kl divergence weight is set to be 0.01. For other experiments, the KL divergence weight is set with a schedule linear on number of epochs going from 0 to 0.01. We also have a weight of 5 multiplied to the prior density.

**AEVB-IAF.** The batch size, number of epochs, learning rate, loss, KL divergence weight, and prior density weight are the same as VAE. The context size, i.e., the size of features used to initialize the flow layers for different datat point, is 10.
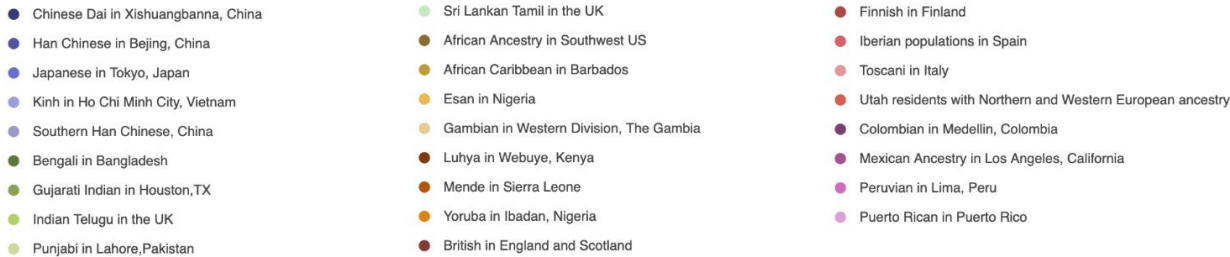
- Chinese Dai in Xishuangbanna, China
- Han Chinese in Bejing, China
- Japanese in Tokyo, Japan
- Kinh in Ho Chi Minh City, Vietnam
- Southern Han Chinese, China
- Bengali in Bangladesh
- Gujarati Indian in Houston,TX
- Indian Telugu in the UK
- Punjabi in Lahore,Pakistan
- Sri Lankan Tamil in the UK
- African Ancestry in Southwest US
- African Caribbean in Barbados
- Esan in Nigeria
- Gambian in Western Division, The Gambia
- Luhya in Webuye, Kenya
- Mende in Sierra Leone
- Yoruba in Ibadan, Nigeria
- British in England and Scotland
- Finnish in Finland
- Iberian populations in Spain
- Toscani in Italy
- Utah residents with Northern and Western European ancestry
- Colombian in Medellin, Colombia
- Mexican Ancestry in Los Angeles, California
- Peruvian in Lima, Peru
- Puerto Rican in Puerto Rico

*Figure 5.* Legend showing what ethnicity each color corresponds to in the 1000 Genomes dataset

**AAEB.** The batch size is set to 128. The number of epochs is 200 for all experiments. The learning rate is 0.0002. The loss is MSE for all experiments. To stabilize the training, we add noise to the input to the discriminator with sigma 0.3 at the start and lower it by 0.1 for every 50 epochs. The noise equals to 0 at epoch 150.

**DDVI.** The batch size is set to 128 for most experiments, except for semi-supervied experiments where the batch size is 1024. The number of epochs is 200 for unsupervised and clustering experiments and 30 for semi-supervised experiments. The learning rate is 0.0001. The loss is BCE for MNIST and CIFAR-10 experiments and MSE for genotype analysis experiments. For unsupervised MNIST and CIFAR-10 experiments, the KL divergence weight is set to 0.003. For semi-supervised MNIST experiment, we use KL divergence weight of 0.1. For semi-supervised CIFAR-10 experiment, we use KL divergence weight of 0.5. For clustering experiment, we use KL divergence weight of 0.005. The number of timesteps is 20 for unsupervised and clustering experiments and 100 for semi-supervised experiments.

## I. Genotype Analysis Experiments Details

Before inputting the data points into any of the visualization methods, we first pre-process it by running a PCA and keep only the first 1000 principal components of the data points. We further divide the features by 30 for all latent variables model methods.

The legend of the 1000 Genomes Visualization plot can be found at Figure 5.

| Method | Pinwheel | | Swiss Roll | | Square | |
|---|---|---|---|---|---|---|
| | ELBO | Latent NLL | ELBO | Latent NLL | ELBO | Latent NLL |
| AEVB | $-12.96\pm1.81$ | $3.26\pm0.60$ | $-12.87\pm4.55$ | $6.25\pm1.58$ | $-7.91\pm0.11$ | $2.91\pm0.17$ |
| AEVB-IAF | $-3.24\pm0.16$ | $1.71\pm0.84$ | $-4.03\pm0.73$ | $5.51\pm0.51$ | $\mathbf{-2.10\pm0.31}$ | $1.71\pm0.77$ |
| AAEB | $N/A$ | $1.70\pm0.41$ | $N/A$ | $\mathbf{3.18\pm0.22}$ | $N/A$ | $1.67\pm0.17$ |
| H-AEVB | $-4.42\pm0.46$ | $\mathbf{1.69\pm0.17}$ | $-5.36\pm0.77$ | $5.74\pm0.55$ | $-2.86\pm0.11$ | $1.64\pm0.09$ |
| PIS | $-2.92\pm1.23$ | $3.61\pm0.62$ | $-4.14\pm0.49$ | $7.14\pm0.14$ | $-4.85\pm0.06$ | $3.91\pm0.06$ |
| DDVI | $\mathbf{-1.38\pm0.44}$ | $1.75\pm0.53$ | $\mathbf{-3.05\pm0.65}$ | $5.66\pm2.63$ | $-2.47\pm0.30$ | $\mathbf{1.58\pm0.09}$ |

*Table 7.* Unsupervised learning on CIFAR-10. We report ELBO and latent negative log-likelihood (Latent NLL) with pinwheel, swiss roll, and square priors.

| Method | Pinwheel | | | Swiss Roll | | | Square | | |
|---|---|---|---|---|---|---|---|---|---|
| | ELBO | Acc | Latent NLL | ELBO | Acc | Latent NLL | ELBO | Acc | Latent NLL |
| AEVB | $-17.14\pm1.46$ | $0.30\pm0.05$ | $2.32\pm0.27$ | $-17.89\pm5.21$ | $0.20\pm0.07$ | $6.56\pm2.25$ | $-13.30\pm1.50$ | $0.30\pm0.05$ | $1.95\pm0.28$ |
| AEVB-IAF | $-5.70\pm0.07$ | $0.47\pm0.01$ | $1.62\pm0.05$ | $-5.53\pm2.82$ | $0.28\pm0.08$ | $6.82\pm1.90$ | $-4.41\pm0.53$ | $0.36\pm0.01$ | $1.58\pm0.15$ |
| AAEB | $N/A$ | $0.25\pm0.01$ | $1.77\pm0.14$ | $N/A$ | $0.23\pm0.01$ | $3.38\pm0.30$ | $N/A$ | $0.23\pm0.04$ | $1.74\pm0.15$ |
| DDVI | $\mathbf{-1.60\pm0.29}$ | $\mathbf{0.49\pm0.01}$ | $\mathbf{1.09\pm0.05}$ | $\mathbf{-4.13\pm1.51}$ | $\mathbf{0.47\pm0.09}$ | $\mathbf{2.29\pm0.08}$ | $\mathbf{-1.73\pm0.64}$ | $\mathbf{0.49\pm0.01}$ | $\mathbf{1.48\pm0.02}$ |

*Table 8.* Semi-supervised learning on CIFAR-10 (10,000 labels). We report ELBO, accuracy using KNN (K=20) classifier (Acc), and latent negative log-likelihood (Latent NLL) with pinwheel, swiss roll, and square priors.

| Method | Latent NLL - Pinwheel | Latent NLL - Swiss Roll | Latent NLL - Square |
|---|---|---|---|
| AEVB | 1.68±0.31 | 5.65±1.58 | 2.78±0.61 |
| AEVB-IAF | 1.64±0.73 | 4.43±1.09 | 1.68±0.27 |
| AAEB | — | — | — |
| H-AEVB | 2.25±3.02 | 4.04±4.62 | 2.22±2.03 |
| DDVI | **1.27±0.21** | **3.86±1.17** | **1.56±0.09** |
| *DDVI (w/o sleep term)* | 2.12 | 5.25 | 2.97 |

*Table 9.* Unsupervised learning on MNIST, including the results of DDVI without the sleep term.