On Kernel Based Variational Autoencoders

Tian Qin
Lehigh University

Wei-Min Huang
Lehigh University

Abstract

In this paper, we bridge Variational Autoencoders (VAEs) (Kingma and Welling, 2013) and kernel density estimations (KDEs) (Rosenblatt, 1956; Parzen, 1962) by approximating the posterior by the expectation of kernel density estimator and deriving a new lower bound of empirical log likelihood. The flexibility of KDEs provides a new perspective of controlling the KL-divergence term in original evidence lower bound (ELBO) which enriches the choice of the posterior and prior pairs in VAE. We show that the Epanechnikov kernel gives the tightest upper bound in controlling the KL-divergence under appropriate conditions (Epanechnikov, 1969; Bickel and Rosenblatt, 1973) in theory and develop a kernel-based VAE called Epanechnikov Variational Autoenocoder (EVAE). The implementation of EVAE is straightforward as Epanechnikov kernel lies in the "location-scale" family of distributions where reparametrization tricks can be applied directly. Compared with Gaussian kernel, Epanechnikov kernel has compact support which should make the generated sample less blurry. The flexibility of new lower bound of ELBO also enables us to employ a two-stage training strategy to treat reconstruction and generation separately, which is an analogue of the idea in VQ-VAE (Van den Oord et al., 2017). Extensive experiments illustrate the potential of EVAE in image generation and the superiority of EVAE over vanilla VAE and other baseline models in the quality of reconstructed images, as measured by the FID score and Sharpness (Tolstikhin et al., 2018). We also carried out additional experiments about the application of EVAE in downstream classification tasks.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

In variational inference, an autoencoder learns to encode the original data $\mathbf{x}$ using learnable function $f_\phi^{-1}(\mathbf{x})$ and then to reconstruct $\mathbf{x}$ using a decoder g_θ. Kingma and Welling (2013) proposed a stochastic variational inference and learning algorithm called Variational Autoencoders (VAEs) which can be further used to generate new data. According to VAE, a lower bound on the empirical likelihood, which is known as ELBO, is maximized to encourage the fitted decoder $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; g_\theta(\mathbf{x}), I_D)$ to reconstruct input $\mathbf{x}$ from latent variable $\mathbf{z}$ accurately and force the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{x}; f_\phi(\mathbf{x}), I_D)$ to be close to the prior $p(\mathbf{z})$.

The isotropic Gaussian priors and approximate posteriors in vanilla VAEs are mathematically and practically convenient since they make the KL-divergence term in the ELBO analytic, allowing for direct optimization by backpropagation. But the main drawbacks are the lack of expressibility of latent space and the possible posterior collapse. There are two popular directions for extending VAEs to address these drawbacks.

One direction is to enhance the posterior expressiveness. For example, Normalizing flow (NF) (Rezende and Mohamed, 2015) applied a sequence of invertible transformations to the simple and tractable initial distribution to achieve more expressive posteriors. β-VAE Higgins et al. (2016) introduced a new parameter β in balancing the reconstruction loss with disentangled latent representation $q_\phi(\mathbf{z}|\mathbf{x})$. Importance weighted Autoencoders (IWAE) Burda et al. (2015) improved log-likelihood lower bound by importance weighting. It approximated the posterior with multiple samples and learned richer latent space representations than VAEs. Saha et al. (2024) proposed the Aggregate Variational Autoencoder (AVAE), which estimates the posterior by Gaussian kernel density estimations (KDEs) to improve the quality of the learned latent space.

Rather than focusing on posterior approximation, another research trajectory involves replacing Gaussian prior with other flexible distributions. For example, Dilokthanakul et al. (2016) used a simple Gaussian mix-

ture prior and Tomczak and Welling (2018) introduced a mixture prior called "VampPrior" which consists of a mixture distribution with components given by variational posteriors. The possibility of non-parametric priors is explored in Nalisnick and Smyth (2017), which used a truncated stick-breaking process. Hasnat et al. (2017) and Davidson et al. (2018) attempted to replace Gaussian prior by von Mises-Fisher(vMF) distribution to cover hyperspherical latent space. For data with heavy-tailed characteristics, Kim et al. (2024) replaced Gaussian prior with heavy-tailed distributions such as the Student's t-distribution, which allows the model to capture data with higher kurtosis, leading to more robust representations. Hao and Shafto (2023) utilized optimal transport theory design priors that enforce a coupling between the prior and data distribution.

Most variants of VAE along these two directions require the closed-form KL divergence. This is essential in implementation but somehow limits the potential applications of more flexible pairs of posterior and prior, which may have no closed-form of KL divergence but could learn versatile latent space representations and alleviate model collapse. To improve the flexibility of the choice of posterior and prior in VAE, in this paper, we estimate the posterior by the expectation of kernel density estimator and derive a corresponding upper bound of KL-divergence, which has closed-form for many distributions. Such elasticity makes the derivation of the optimal functional form of kernel possible and the implementation time efficient.

It is true that we can simply employ Monte-Carlo simulation (MC) to approximate KL-divergence for complicated posteriors. But the price is the time efficiency. For example, Davidson et al. (2018) employed von Mises-Fisher (vMF) distribution instead of Gaussian to better capture latent hyperspherical structure. With uniform prior, their proposed Hyperspherical Variational Auto-Encoders(HVAE) has closed form of KL-divergence while sampling from von Mises-Fisher (vMF) distribution requires acceptance-rejection method, which can be time consuming for high-dimensional latent space. Additionally, Saha et al. (2024) directly employed a Gaussian kernel density estimator to estimate the posterior, which requires a large number of samples and is time consuming to achieve decent performance. Many VAE type methods also suffer from the trade-off between reconstruction and generation.

Our main contributions can be summarized as follows:

1. We formulate the latent space learning process in VAE as a problem of kernel density estimation. Inspired by the results in Bickel and Rosenblatt (1973) , we then model the posterior by the expectation of corresponding kernel density estimator and derive an upper bound of KL-divergence, which has closed-form for many distributions. Some asymptotic results are established as well. See details in section 2.2 and 3

2. After that, we utilize the conclusion from Bickel and Rosenblatt (1973) and show that the derived upper bound of KL-divergence is tightest when we employ Epanechnikov kernel. The derivation connects a quadratic functional with KL-divergence, which to our best knowledge, is the first attempt to bridge the concept of asymptotic distribution of KDEs with VAE.

3. We then propose Epanechnikov VAE (EVAE) which will learn the latent space representation from KDE based posterior. After fully training the encoder and decoder of EVAE, a DCGAN (Radford et al., 2015) will be fitted to learn the Epanechnikov posterior trained in the previous stage for the sake of generation. We conducted detailed comparisons between the Epanechnikov VAE (EVAE) and Gaussian VAE in many benchmark image datasets under the standard encoder-decoder structure. The extensive experiments demonstrate the superiority of EVAE over other baselines in image reconstruction and generation quality. Additionally, the EVAE shows strong performance in downstream classification tasks.

The remaining sections of this paper are organized as following schema. In section 2, we provide some preliminaries involving VAE and kernel density estimations. In section 3, we show that Epanechnikov kernel gives the tightest upper bound in bounding the KL-divergence in the functional sense under appropriate conditions. Based on results in section 3, we propose the Epanechnikov VAE (EVAE) in section 4. The comparisons between EVAE, vanilla VAE and other baselines in benchmark datasets are illustrated in section 5. Section 6 discusses some related works of EVAE. Section 7 raises few characteristics and limitations of EVAE and suggests some future directions. The pytorch codes for the implementation of EVAE and main experiments are available in supplementary materials. All experiments are performed on a laptop with 12th Gen Intel(R) Core(TM) i7-12700H (2.30 GHz) ,8.0 GB RAM and NVIDIA 4070 GPU.

2 PRELIMINARY

2.1 VAE formulation

Consider a dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$ consists of n i.i.d samples from space $\mathcal{X}$ whose dimension is d. In

VAE, we assume that every observed data $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_d^{(i)}) \in \mathcal{X}$ is generated by a latent variable $\mathbf{z}^{(i)} \in \mathcal{Z}$ whose dimension is p. Then the data generation process can be summarized in 2 steps. It first produces a variable $\mathbf{z}^{(i)}$ from some prior distributions $p(\mathbf{z})$. Then given the value of $\mathbf{z}^{(i)}$, an observed value $\mathbf{x}^{(i)}$ is generated from certain conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. We typically assume $d > p$ and likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ are differentiable distributions w.r.t θ and $\mathbf{z}$. To maximize the empirical log likelihood $\log p_\theta(\mathbf{x})$, we need to evaluate the integral in the form

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \log \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}\,,$$

which is intractable in most cases. Fortunately, we can instead maximizing its log evidence lower bound (ELBO) $\mathcal{L}$ with the help of Jensen's inequality:

$$\log p_\theta(\mathbf{x}) \geq \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\phi(\cdot|\mathbf{x})}} [\log p_\theta(\mathbf{x}|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\,, \tag{1}$$

where the RHS of inequality (1) is called evidence lower bound (ELBO) and $KL(\cdot||\cdot)$ represents the KL-divergence between two distributions.

The conditional likelihood $p_\theta(\mathbf{x}|\mathbf{z})$, approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior distribution $p(\mathbf{z})$ can be chosen independently. For convenience, most applications of VAE employ Gaussian parametrization for all three likelihoods. Since we would like to investigate new forms of posterior and prior rather than the form of conditional likelihood $p_\theta(\mathbf{x}|\mathbf{z})$, we can assume a multivariate Bernoulli or Gaussian model w.r.t $p_\theta(\mathbf{x}|\mathbf{z})$ for simplicity. For example, under multivariate Gaussian, we have $\log p_\theta(\mathbf{x}|\mathbf{z}) = \log \mathcal{N}(\mathbf{x}; g_\theta(\mathbf{z}), I)$. As to multivariate Bernoulli model, we have $\log\, p_\theta(\mathbf{x}|\mathbf{z}) = \sum_{i=1}^d [x_i \log\, (g_\theta(\mathbf{z})_i) + (1 - x_i) \log\, (1 - g_\theta(\mathbf{z})_i)]$ where $g_\theta(\mathbf{z}) : \mathcal{Z} \to \mathcal{X}$ are typically neural-network parametrizations.

To maximize ELBO, we now need to minimize the following target function for given data $\mathbf{x}$:

$$\mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\phi(\cdot|\mathbf{x})}} [-\log p_\theta(\mathbf{x}|\mathbf{z})] + KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\,. \tag{2}$$

The two terms in equation (2) are named as "reconstruction error" and "divergence" or "regularization term", respectively. The "divergence" term regularizes the mismatch between approximate posterior and prior distribution.

2.2 Model the posterior as the expectation of kernel density estimator

A common assumption of latent space in VAE is the factorization of approximate posterior, i.e. dimensions of latent space are independent with each other. Under this assumption, we only need to consider the KDE formulation for posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and prior $p(\mathbf{z})$ in one-dimensional case. Similar arguments can be applied to multi-dimensional cases by additivity of KL-divergence under independence. For the consistency of notations, we still use bold letter $\mathbf{x}$ and $\mathbf{z}$ to denote x and z in one-dimensional KDE. Given $Y_1, ..., Y_m$ be i.i.d random variables with a continuous density function f, Parzen (1962); Rosenblatt (1956) proposed kernel density estimate $f_m(y)$ for estimating $f(y)$ at a fixed point $y \in \mathbb{R}$:

$$\begin{aligned} f_m(y) &= \frac{1}{mb(m)} \sum_{i=1}^m K\left[\frac{y - Y_i}{b(m)}\right] \\ &= \frac{1}{b(m)} \int K\left[\frac{y - t}{b(m)}\right] dF_m(t)\,, \end{aligned} \tag{3}$$

where F_m is the sample distribution function, K is an appropriate kernel function such that $\int K(y)dy = 1$ and the positive number b_m, which typically relies on the sample size m, is called bandwidth such that $b(m) \to 0, mb(m) \to \infty$ as $m \to \infty$[1].

On the other hand, with inequality $\log\,(t) \leq t - 1$ (for $t > 0$) we can bound KL-divergence as follows:

$$\begin{aligned} KL(q||p) = \int q(\mathbf{z}) \log\, \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} &\leq \int q(\mathbf{z}) (\frac{q(\mathbf{z})}{p(\mathbf{z})} - 1) d\mathbf{z} \\ &= \int \frac{(q(\mathbf{z}) - p(\mathbf{z}))^2}{p(\mathbf{z})} d\mathbf{z}\,. \end{aligned} \tag{4}$$

Let the posterior $q_\phi(\mathbf{z}|\mathbf{x}) = \mathbb{E}_0(q_{m,\phi}(\mathbf{z}))$, where

$$q_{m,\phi}(\mathbf{z}) = \frac{1}{mb(m)} \sum_{j=1}^m K_{\phi,\mathbf{x}}\left[\frac{\mathbf{z} - \mathbf{Z}_j}{b(m)}\right], \quad \mathbf{Z}_j \overset{i.i.d}{\sim} p(\tilde{\mathbf{z}})\,.$$

is a kernel density estimator of $q_\phi(\mathbf{z}|\mathbf{x})$ with kernel $K_{\phi,\mathbf{x}}$, given KDE sample size m, data point $\mathbf{x}$ and parameter ϕ. Expectation $\mathbb{E}_0$ is taken w.r.t the samples from prior distribution $p(\tilde{\mathbf{z}})$. In other words, we model the posterior as the expectation of the kernel density estimator. The subscript ϕ of kernel function implies that the parameters in kernel can be learned by neural networks.

By inequality (4) and Jensen inequality, we obtain

$$\begin{aligned} KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) &\leq \int \frac{[\mathbb{E}_0(q_{m,\phi}(\mathbf{z})) - p(\mathbf{z})]^2}{p(\mathbf{z})} d\mathbf{z} \\ &\leq \mathbb{E}_0 \left[\int \frac{(q_{m,\phi}(\mathbf{z}) - p(\mathbf{z}))^2}{p(\mathbf{z})} d\mathbf{z}\right]. \end{aligned} \tag{5}$$

[1] We omit the limits of integrals if they are $-\infty$ to ∞.

To this end, we obtain a new lower bound of log-likelihood of data:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \\ &\quad - \mathbb{E}_0\left[\int \frac{(q_{m,\phi}(\mathbf{z}) - p(\mathbf{z}))^2}{p(\mathbf{z})} d\mathbf{z}\right].\end{aligned} \tag{6}$$

Since we model the posterior as the expectation of the kernel density estimator, the number of latent variable Z generated from prior distribution, which is m, is just used for theoretical derivation and we found that sampling latent variable Z once is good enough in practice, as suggested by Kingma and Welling (2013).

In the kernel density estimation theory, Epanechnikov (1969) showed that the Epanechnikov kernel minimizes $\int \mathbb{E}_0[q_{m,\phi}(\mathbf{z}) - p(\mathbf{z})]^2 d\mathbf{z}$ asymptotically, which gives us a hint of the functional optimization of kernel K in right hand side of inequality (5). The main assumptions of deriving optimal functional form of kernel are enclosed in Appendix A.

3 CHOICE OF KERNEL

Let $f_m(t)$ be a kernel density estimate of a continuous density function f at t, as defined in equation (3), we construct a statistic T_m as follows:

$$T_m = mb(m) \int [f_m(t) - f(t)]^2 a(t) dt\,,$$

where $a(t)$ is an appropriate weight function. We now restate the main result of Bickel and Rosenblatt (1973) as Theorem 3.1:

Theorem 3.1 (Bickel and Rosenblatt (1973)) *Let assumptions **A1** − **A4** in Appendix A hold and suppose that the weight function a is integrable piecewise continuous and bounded. Suppose* $b(n) = o(n^{-\frac{2}{9}})$ *and* $o(b(n)) = n^{-\frac{1}{4}}(log(n))^{\frac{1}{2}}(loglogn)^{\frac{1}{4}}$ *as* $n \to \infty$, *then* $b^{-\frac{1}{2}}(n)(T_n - I(K)\int f(t)a(t)dt)$ *is asymptotically normally distributed with mean 0 and variance* $2J(K)\int a^2(t)f^2 dt$ *as* $n \to \infty$, *where*

$$I(K) = \int K^2(t)dt, J(K) = \int \left[\int K(t+y)K(t)dt\right]^2 dy\,. \tag{7}$$

In other words, under Theorem 3.1, we can say $\mathbb{E}[T_m]$ is approximately $I(K)\int f(t)a(t)dt$ when m is large.

Let $mb(m)/n = 1$ and $b(m)$ satisfy the conditions in Theorem 3.1, by the asymptotic result of Theorem 3.1, the inequality (5) becomes[2]:

$$\begin{aligned}&KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &\leq \frac{mb(m)}{n}\mathbb{E}_0\left[\int \frac{(q_{m,\phi}(\mathbf{z}) - p(\mathbf{z}))^2}{p(z)} d\mathbf{z}\right] \\ &= \frac{1}{n}\mathbb{E}_0\left[mb(m)\int \frac{(q_{m,\phi}(\mathbf{z}) - p(\mathbf{z}))^2}{p(\mathbf{z})} d\mathbf{z}\right] \\ &\overset{\text{n large}}{\approx} \frac{I(K_{\phi,\mathbf{x}})\int p(\mathbf{z})\frac{1}{p(\mathbf{z})}d\mathbf{z}}{n} = B(\mathbf{x})\frac{I(K_{\phi,\mathbf{x}})}{n}\,,\end{aligned} \tag{8}$$

where $B(\mathbf{x})$ is the length of support interval of $p(\mathbf{z})$ given input data $\mathbf{x}$. Note that if the prior has infinite length of support, the inequality (8) becomes theoretically useless[3]. Thus we assume finite $B(\mathbf{x})$ in implementation. We can now find the kernel which gives the tightest upper bound of KL-divergence i.e. the kernel $K^*_{\phi,\mathbf{x}}$ minimizing $I(K_{\phi,\mathbf{x}})$ given fixed parameters and data point $\mathbf{x}$. Lemma 3.2 shows that Epanechnikov kernel is the optimal choice.

Inequality (8) relaxes the form of the prior distribution, making the choice of prior more flexible. In section 4, we choose to only parameterize the support of latent prior distribution so that it could facilitate the learning of latent posterior, which also motivates us employ a two-stage training strategy, as taken by VQ-VAE (Van den Oord et al., 2017), to treat reconstruction and generation separately. See details section 4.

Lemma 3.2 *Let* $\mathcal{K}$ *be the set of all* $L^1(-\infty, \infty)$ *non-negative functions* K *satisfying*

$$\int K(t)dt = 1, \int tK(t)dt = \mu, \int (t-\mu)^2 K(t)dt = \frac{1}{5}r^2$$

where $\mu \in (-\infty, \infty), r > 0$. *Then the functional* $I(K)$ *in equation (7) is minimized on* $\mathcal{K}$ *uniquely by*

$$K^*(t) = \begin{cases} \frac{3}{4r}\left(1 - \left(\frac{t-\mu}{r}\right)^2\right) & t \in [\mu - r, \mu + r] \\ 0 & otherwise \end{cases}$$

and $min_K I(K) = I(K^*) = \frac{3}{5r}$. *The optimal kernel* K^* *is named as Epanechnikov kernel (Epanechnikov, 1969).*

The proof is based on the idea of Lagrange multiplier. See Appendix B.1.

4 EPANECHNIKOV VAE

Getting inspired by the functional optimality of Epanechnikov kernel in controlling KL-divergence, we

[2]For the consistency of notations, we still use bold $\mathbf{x}$ and $\mathbf{z}$ to denote one-dimensional variable x and z in inequality ((8)) under the context of Theorem 3.1.

[3]However, we also found that replacing the uniform prior with Gaussian prior can still make kernel based VAEs work. See extra experiments in section 5.6.

propose the Epanechnikov Variational Autoencoder (EVAE) whose resampling step is based on Epanechnikov kernel. There are two main differences between EVAE and VAE. The latent distribution in EVAE is assumed to be estimated by the Epanechnikov kernel rather than multivariate isotropic Gaussian. And EVAE is trained to minimize a different target function (9):

$$\mathop{\mathbb{E}}_{z\sim q_{\phi(\cdot|\mathbf{x})}}[-\log p_\theta(\mathbf{x}|\mathbf{z})] + B(\mathbf{x})\frac{I(K^*_{\phi,\mathbf{x}})}{n}\,, \tag{9}$$

where n can be the sample size or minibatch size, ϕ are outputs of the encoding network, $K^*_{\phi,\mathbf{x}}$ is Epanechnikov kernel given data point $\mathbf{x}$ and trainable neural network parameter ϕ, and support parameter $B(\mathbf{x})$ relies on input $\mathbf{x}$. The choice of $B(\mathbf{x})$ is flexible and we find that setting $B(\mathbf{x}^{(i)}) = \mathbf{r}^{(i)}$ simplifies the target function and inspires us to split the training part of EVAE into two phases, whose idea is analogous to VQ-VAE (Van den Oord et al., 2017).

In the first stage, we optimize the sample version of equation (9) at i-th data point $\mathbf{x}^{(i)}$:

$$\tilde{\mathcal{L}}(\theta,\phi,\mathbf{B};\mathbf{x}^{(i)}) \approx -\frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\mathbf{x}^{(i)})|\mathbf{z}^{(i,l)}) + \frac{3}{5M}\,, \tag{10}$$

where $i \in \{1,2,...,M\}$, p is the dimension for latent space, M is is the minibatch size. Since $\frac{3}{5M}$ is a constant, we can safely ignore this constant term in optimizing equation (9). In vanilla VAE, Kingma and Welling (2013) suggested that the number of regenerated samples L can be set to 1 as long as the minibatch size is relatively large. Based on our experiment experience, this is also the case for EVAE. During stage 1 training, the encoder learns the latent posterior $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$. By definition, the density of $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ can be derived as follows:

$$\begin{aligned} q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) &= \mathbb{E}_0\left[\frac{1}{mb(m)}\sum_{j=1}^{m} K^*_{\phi,\mathbf{x}^{(i)}}\left(\frac{\mathbf{z}-\mathbf{Z}_j}{b(m)}\right)\right] \\ &= \mathbb{E}_0\left[\frac{1}{b(m)}K^*_{\phi,\mathbf{x}^{(i)}}\left(\frac{\mathbf{z}-\mathbf{Z}_1}{b(m)}\right)\right] \\ &= \int \frac{1}{b(m)}K^*_{\phi,\mathbf{x}^{(i)}}\left(\frac{\mathbf{z}-\tilde{\mathbf{z}}}{b(m)}\right)p_{\mathbf{z}}(\tilde{\mathbf{z}})d\tilde{\mathbf{z}}\,, \end{aligned} \tag{11}$$

where $p_{\mathbf{z}}$ is the density of prior distribution.

Thus the latent posterior constructed from stage 1 can be viewed a convolution between two random variables, i.e.

$$Z(\mathbf{x}^{(i)}) = b(m)K^*(\mathbf{x}^{(i)}) + \mathbf{r}^{(i)}(\mathbf{x})\cdot U\,, \tag{12}$$

where U represents a uniform distribution in $[-1,1]$ and we assume that the prior has the same support length as the posterior, which is modeled by the Epanechnikov kernel. Once we can construct the latent space with a trained encoder, a DCGAN will be trained to learn the encoded latent space distribution $Z(\mathbf{x}^{(i)})$ in the second stage to generate a dense representation of the latent space from random noise. In other words, we train a DCGAN generator $G_\theta(\tilde{Z})$ in stage 2 so that $G_\theta(\tilde{Z})$ could generate a similar distribution of encoded space, where θ represents the parameter for the generator and random noise $\tilde{Z}$ follows the standard Gaussian for simplicity.

Essentially, identity (12) decomposes the posterior into two parts. One is the prior distribution information, the other represents the incremental updates from new information induced by data $\mathbf{x}^{(i)}$. The Epanechnikov kernel can be viewed as the "optimal" direction of perturbing the prior distribution and the coefficient $b(m)$ can be interpreted as "step size", which controls the deviation of posterior from the prior. The resampling step in EVAE can also be divided into two parts, as described in Algorithm 1 where we assume that the mean parameters of the Epanechnikov kernel are absorbed in the encoder network. For the sake of finite support and simplicity, we assume that the step size $b(m)$ is the same for all dimensions. The theoretical conditions for $b(m)$ is demanding. In practice we found that setting coefficient $b(m)$ as $b(100) = 100^{-2/9} \approx 0.3594$ is good enough.

A DCGAN will be trained to approximate the latent posterior $q_\phi(\mathbf{z}|\mathbf{x})$ for the sake of image generation. The detailed structure of DCGAN in learning latent space can be found in Appendix C. We follow the main idea in Radford et al. (2015) but replace the convolution layers with linear layers. The intuition is that there is no spatial relationship between latent feature dimensions.

Algorithm 1 Resampling step in a minibatch of EVAE (Stage 1 training)

Require: Latent space dimension d_z; Minibatch size M. Step size $b(m)$. Spread $\mathbf{r}_\phi(\mathbf{x})$ is learned by encoder networks given input $\mathbf{x}$ and have dimension $M \times d_z$.

1: Encode a input $\mathbf{x}$ into $Enc(\mathbf{x})$.
2: Sample a $M \times d_z$ matrix $\mathbf{K}$ where each (i,j) entry of the random matrix $\mathbf{K}$ is sampled from an standard Epanechnikov kernel supported on $[-1,1]$.
3: Scale sampled $\mathbf{K}$ by the location-scale formula: $\mathbf{Z} = \mathbf{r}_\phi(\mathbf{x}) \odot \mathbf{K}$.
4: **Return:** $b(m)\odot\mathbf{Z} + \mathbf{r}_\phi(\mathbf{x})\odot\mathbf{U}$, $\mathbf{U} \overset{i.i.d}{\sim} \text{Unif}[-1,1]$.

In Algorithm 1, we apply reparametrization trick to sample $\mathbf{z}$ from general Epanechnikov kernel, i.e $\mathbf{z}^{(i,l)} = \mathbf{r}^{(i)} \odot \mathbf{k}^{(l)}$, where $\mathbf{k}^{(l)}$ is sampled from standard Epanechnikov kernel supported on $[-1,1]$ as it

Algorithm 2 Sampling from centered Epanechnikov kernel supported on $[-1, 1]$

1: Sample $U_1, U_2, U_3 \overset{i.i.d}{\sim} \text{Unif}[-1, 1]$.
2: Set $U = \text{Median}(U_1, U_2, U_3)$
3: **Return:** U

lies in the "location-scale" family. We use $\odot$ to signify the element-wise product. There are many ways to sample from standard Epanechnikov kernel, such as accept-rejection method. For efficiency, we provide a faster sampling procedure in Algorithm 2. The theoretical proof is given in Appendix B.2. Once the encoder and decoder are fully trained, we can get the encoded space from Algorithm 1 and fit a DCGAN with respect to the actual encoded space, where the generator will generate fake encoded space from random noise and the discriminator will judge the quality of generated encoded space. Essentially, we train a DCGAN for the dense encoded space from EVAE.

5 EXPERIMENTS

In this section, we will compare the proposed EVAE model with (vanilla) VAE whose posterior and prior are modelled by isotropic multivariate Gaussian in real datasets. To assess the quality of reconstructed images, we employ the Frechet Inception Distance (FID) score Heusel et al. (2017) to measure the distribution of reconstructed images with the distribution of real images. The lower, the better. Inception_v3 (Szegedy et al., 2016) is employed as default model to generate features of input images, which is a standard implementation in generative models. Unlike the unbounded support of Gaussian distribution, the compact support for Epanechnikov kernel could help EVAE generate less blurry images. To check this claim empirically, we calculated sharpness (Tolstikhin et al., 2018) for reconstructed images by a 3×3 Laplace filter. See more details in Appendix E.

5.1 Benchmark datasets

We trained EVAE and VAE on four benchmark datasets: MNIST Deng (2012), Fashion-MNIST Xiao et al. (2017), CIFAR-10 Krizhevsky (2009) and CelebA Liu et al. (2015). The detailed information for training parameters are attached in Section 3 of Appendix. In terms of FID score, we observe that EVAE has an edge in high dimensions ($d_z = 32, 64$), indicating the better quality of reconstructed images from EVAE. Additionally, when $d_z = 64$, EVAE generates higher sharpness of reconstructed images for all datasets, as illustrated in Table 5 and Table 6 in Appendix I.1. Those results empirically justify the positive effect of having compact support in posteriors and priors. As to CIFAR-10, EVAE has larger sharpness in high dimensions ($d_z = 32, 64$) while it is relatively mediocre in low dimensions. Possible reasons include low-resolution(blurriness) of original images and deficient expressibility of simple CNN models in low dimensions. But the validation reconstruction loss curves in Appendix G indeed authenticate the superiority of EVAE over benchmark datasets, including CelebA.

5.2 Reconstruction samples

Figure 1 presents some test reconstructed samples from trained VAE and EVAE with $d_z = 64$ of CIFAR10, respectively. We can see many images reconstructed from EVAE are able to pick up local features better than the VAE. And reconstructed samples from CIFAR-10 are clearer and closer to the original images, as indicated by large gap between FID scores. See more examples of MINST, Fashion-MNIST and CelebA-64 datasets in Appendix F. The statistical analysis from binomial test based on total experiments (Appendix E) shows that EVAE significantly outperforms VAE in FID and sharpness.

5.3 Unconditional samples

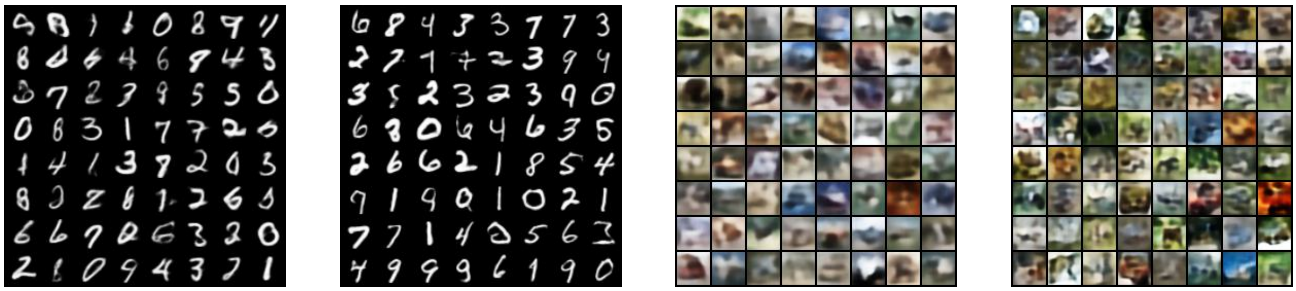
Figure 2 demonstrates the unconditional samples from MINST dataset with EVAE and VAE. To generate images with EVAE, we directly sample from random noise , and feed them into the trained DCGAN generator to obtain the latent vector sample, which is then used as input of the trained decoder. We observe that unconditional samples from EVAE also have a higher quality than those of VAE, as measured by the FID score, which implies the potential of EVAE in learning dense latent spaces. One intuition is that the generation step in EVAE is more flexible as we learn the kernel based posterior separately. Unlike VQ-VAE, EVAE doesn't require the codebook in connecting discrete latent space and dense embedding space, providing a new insight of latent space representation.

5.4 Comparisons with baselines

In this section we conducted extra experiments comparing the reconstruction ability of EVAE with open-source variants of VAE that improve the representational capacity of the encoder such as $\beta-$VAE (Higgins et al., 2016), IWAE (Burda et al., 2015), WAE (Tolstikhin et al., 2018) and HVAE (Davidson et al., 2018) on three datasets. Note that we did not involve Normalizing Flow/GAN/Diffusion in the baselines as they belong to different types of image generation model. We used uniform prior in EVAE and we ran all experiments three times. We reported mean and standard

(a) Real images (b) VAE reconstructed images (c) EVAE reconstructed images

Figure 1: (a) Sampled real images from hold-out samples in CIFAR-10 (b) Reconstructed images by VAE. (c) Reconstructed images by EVAE. Dimension $d_z = 64$ for both models.



(a) VAE(FID=12.57) (b) EVAE(FID=7.1) (c) EVAE(FID=155.6) (d) EVAE(FID=91.1)

Figure 2: (a) Unconditional samples generated from VAE (MNIST) (b) Unconditional samples generated from EVAE (MNIST) (c) Unconditional samples generated from VAE (CIFAR10)(d) Unconditional samples generated from EVAE (CIFAR10). Latent dimension d_z is 32 for (a) and (b). Latent dimension d_z is 64 for (c) and (d).

deviation of FID score (for testing set reconstruction quality). Same for sharpness. For fair comparisons, we used same encoder and decoder CNN architecture for all baselines. The only difference comes from posterior modeling and prior modeling. All latent dimensions d_z are set to 64. Other hyperparameters and training details are the same as in the Appendices C and E.

According to Table 2, the EVAE model consistently achieves strong performance across the board, particularly in terms of FID and sharpness metrics. The WAE(MMD) model also performs well in many scenarios, which is not surprising since it extends the KL divergence used in standard VAEs to the more general Wasserstein distance. The IWAE model shows decent results by using importance-weighted sampling to close the gap between the ELBO and the true log-likelihood. In our experiments, we used k=5 samples for this importance sampling step. In contrast, the Hyperspherical VAE (HVAE) performed poorly, possibly due to the complexity of the von-Mises Fisher prior and the choice of the hypersphere manifold's dimension. Our experiments also found that the VQ-VAE performed worse than other models. This can be attributed to its use of only simple Convolutional Neural Networks (CNNs). When we incorporated residual layers, as in the original implementation (Van den Oord et al., 2017), the VQ-VAE's FID score on the CIFAR10 dataset improved significantly to approximately 49. This suggests that the high-quality image reconstruction seen in VQ-VAE is not solely a result of its vector quantization component.

In terms of efficiency, all models had similar running times, with the exception of HVAE. Its use of an acceptance-rejection method for sampling, especially in high-dimensional settings, made it relatively more time-consuming.

5.5 Downstream classification

Latent variables are known to be useful for downstream tasks because they capture the underlying structure of the data distribution (He et al., 2019; Chen et al., 2020). To demonstrate the potential of the latent space representation learned from EVAE, we compared our method against VAE with respecto the downstream classification tasks.

Table 2: EVAE and other baselines

Models	Fashion-MNIST		CIFAR10	
	FID	Sharpness	FID	Sharpness
VAE	39.1(0.3)	0.016(3E-4)	142.9(1.7)	0.035(7E-4)
EVAE	**12.2(0.2)**	**0.027(2.1E-4)**	**79.7(0.1)**	**0.036(7E-4)**
β-VAE($\beta = 0.1$)	19.5(0.5)	0.023(5E-4)	84.4(0.8)	0.035(5E-4)
β-VAE($\beta = 5$)	74.5(2.0)	0.01(2E-4)	239.8(1.9)	0.034(1.4E-3)
IWAE($k = 5$)	26.8(0.1)	0.02(3E-4)	125.4(1.6)	0.036(7E-4)
WAE(MMD)	22.1(0.5)	0.021(6E-4)	140.1(7.2)	0.034(7E-4)
HVAE	36.8(0.3)	0.016(1E-4)	195.2(0.8)	0.035(1E-3)
VQ-VAE	42.86(1.9)	0.015(2E-4)	218.92(2.2)	0.034(5E-4)

Generally, EVAE's latent space representations lead to better downstream classification accuracy on the MNIST dataset than a standard VAE's. This suggests EVAE learns a more effective and informative data representation, which improves performance on a related task. The results are listed in the Table 3 and the detailed training strategies are described in Appendix H.

Table 3: Downstream classification testing accuracy on MNIST dataset under different latent space dimensions

Models	$d_z = 8$	$d_z = 16$	$d_z = 32$
VAE	97.41 ± 0.05	98.38 ± 0.02	98.76 ± 0.14
EVAE	**97.87±0.03**	**98.83± 0.03**	98.76± 0.06

5.6 Extra experiments

We further replace the uniform prior with Gaussian prior in training the EVAE. Table I.2 in Appendix shows that the superior performance of EVAE comes from the kernel based posterior with compact support and not rely on the choice of prior. In addition, Tabular diffusion model (TabDiff) (Kotelnikov et al., 2023) is another choice in learning the latent space if we view the latent space as continuous tabular data. The results in Table 4 illustrate that DCGAN is a better latent distribution learner compared to tabular diffusion model. Another drawback of diffusion based model is the sequential denoising process is relatively time consuming. Thus DCGAN is a more efficient choice. The implementation of TabDiff follows the structure in Kotelnikov et al. (2023).

In addition, Table 9 in Appendix I.3 demonstrates that EVAE has the comparable time efficiency to VAE. The slight increase in training time for EVAE results from the sampling process of the Epanechnikov kernel, which requires a few more samples to achieve the Epanechnikov density, as described in Algorithm 2. The difference in training time is negligible when the latent space dimension is large, which facilitates the application of EVAE in high-resolution datasets.

We also compare the PSNR (Peak Signal-to-Noise Ratio) of different baselines on CIFAR10 to contextualize the overall reconstruction fidelity of EVAE. All training details follow the same setting in section E. From Table 10 in Appendix I.4, we can see that EVAE still outperforms other baselines under PSNR, implying EVAE has higher reconstruction quality in general.

Table 4: FID (on unconditional generated samples) on MNIST dataset under different latent space learners (TabDiff and DCGAN)

Models	$d_z = 8$	$d_z = 16$	$d_z = 32$
EVAE with DCGAN	**14.27**	**9.63**	**7.86**
EVAE with TabDiff	21.97	13.02	15.05

6 RELATED WORKS

Normalizing flow(NF) (Mohamed and Rezende, 2015) transforms a simple prior distribution (like a standard $N(0, I)$) into a complex target distribution via a series of diffeomorphisms, which enhances the posterior approximations. If we replace the simple Gaussian prior with kernel-based posterior developed in our work, it's possible we can let NF start from an adaptive data manifold, reducing the flow complexity and improving the tail modeling. Extending the implementation of Epanechnikov kernel to more complicated generative models such as diffusion models(Ho et al., 2020), normalizing flows(Mohamed and Rezende, 2015; Horvat and Pfister, 2021) and other variants of VAEs is one of our future works.

Casale et al. (2018) proposed the Gaussian Process (GP) prior to address the issue that vanilla VAE assumes the latent sample representations are independent and identically distributed, which may be too strong for dependent data, e.g. time series. Since the

derivation of EVAE also assumes the i.i.d latent distribution, it's possible the idea in Casale et al. (2018) could improve the performance in EVAE. We will leave this as one of our future works.

Sun et al. (2024) proposed weighted standard deviation of logit and performed a plug-and-play Z-score pre-process of logit standardization before applying softmax and Kullback-Leibler divergence in matching logits of teacher and student models. This method reminds us the downstream application (classification) of EVAE where the KL is instead approximated by asymptotic results from KDE. It's possible that the logit standardization could boost the downstream classification performance of EVAE if we can mode this process as a special case of knowledge distillation.

7 DISCUSSION AND LIMITATION

A more precise approximation of KL-divergence. In section 2.2, we bounded the KL-term by a simple inequality $\log t \leq 1 - t$, which limits the potential of EVAE in more complicated cases such as high resolution images. It's possible to derive sharper bounds with higher order approximation where the Epanechnikov kernel may not be the optimal one.

Optimal kernel under different criteria. In this paper, we mainly focused on the L_2 deviation of KDEs, which is measured by the functional $I(K)$ proposed in Theorem 3.1. However, in the general theory of KDEs, different criteria lead to different optimal kernels. For example, if we want to minimize the asymptotic variance of T_m, which is related to the convolution functional $J(K) = \int \left[\int K(t+y)K(t)dt\right]^2 dy$ defined in section 3, the optimal kernel is just the uniform kernel, as derived by Ghosh and Huang (1991).

In addition, the Bickel-Rosenblatt statistic T_m we defined in section 3 is proposed by Bickel and Rosenblatt (1973), which can be viewed as a "continuous version" of the ordinary k-cell Chi-Square statistic and the KL-divergence is also a type of Chi-Square function Berkson (1980). Replacing KL-divergence with expectation of another type of Chi-Square statistic makes the kernel optimization (minimum Chi-Square estimation) in section 3 possible, which supports the claim in Berkson (1980) that minimum Chi-Square estimate is more general and flexible.

The factorization of approximate posterior. Our derivation of the new target function (10) is based on the independence among hidden dimensions, which is a fairly strong condition in practice. Exploring the theory of KDEs with dependent latent dimensions would be another compelling direction. For instance, we can utilize the copula theory (Hofert et al., 2018; Tagasovska et al., 2019; Joe, 2014) to deal with the non mean-field posterior family.

References

Joseph Berkson. Minimum Chi-Square, not Maximum Likelihood! *The Annals of Statistics*, 8(3):457 – 487, 1980. doi: 10.1214/aos/1176345003. URL https://doi.org/10.1214/aos/1176345003.

P. J. Bickel and M. Rosenblatt. On Some Global Measures of the Deviations of Density Function Estimates. *The Annals of Statistics*, 1(6):1071 – 1095, 1973. doi: 10.1214/aos/1176342558. URL https://doi.org/10.1214/aos/1176342558.

Yuri Burda, Roger Baker Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015. URL https://api.semanticscholar.org/CorpusID:11383178.

Francesco Paolo Casale, Adrian V Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 10390–10401, Red Hook, NY, USA, 2018. Curran Associates Inc.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. *ArXiv*, abs/1804.00891, 2018.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, M. J. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *ArXiv*, abs/1611.02648, 2016. URL https://api.semanticscholar.org/CorpusID:1327363.

V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969. doi: 10.1137/1114019. URL https://doi.org/10.1137/1114019.

B. K. Ghosh and Wei-Min Huang. The Power and Optimal Kernel of the Bickel-Rosenblatt Test for Goodness of Fit. *The Annals of Statistics*, 19(2):999 – 1009, 1991. doi: 10.1214/aos/1176348133. URL https://doi.org/10.1214/aos/1176348133.

Xiaoran Hao and Patrick Shafto. Coupled variational autoencoder. 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 12546–12555. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/hao23b.html.

Abul Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, and Liming Chen. von mises-fisher mixture model-based deep learning: Application to face verification. *ArXiv*, abs/1706.04264, 2017. URL https://api.semanticscholar.org/CorpusID:22888520.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2019. URL https://api.semanticscholar.org/CorpusID:207930212.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016. URL https://api.semanticscholar.org/CorpusID:46798026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Marius Hofert, Ivan Kojadinovic, Martin Machler, and Jun Yan. *Elements of Copula Modeling with R*. Use R! Springer International Publishing AG, Cham, Switzerland, 2018. ISBN 9783319896342.

Christian Horvat and Jean-Pascal Pfister. Denoising normalizing flow. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:245018141.

H. Joe. *Dependence Modeling with Copulas*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2014. ISBN 9781466583221. URL https://books.google.com/books?id=09ThAwAAQBAJ.

Juno Kim, Jaehyuk Kwon, Mincheol Cho, Hyunjong Lee, and Joong-Ho Won. t^3-variational autoencoder: Learning heavy-tailed data with student's t and power divergence. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RzNlECeoOB.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with 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 17564–17579. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kotelnikov23a.html.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738. IEEE Computer Society, 2015. ISBN 978-1-4673-8391-2. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2015.html#LiuLWT15.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *ArXiv*, abs/1509.08731, 2015. URL https://api.semanticscholar.org/CorpusID:12860852.

Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1jmAotxg.

Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065 – 1076, 1962. doi: 10.1214/aoms/1177704472. URL https://doi.org/10.1214/aoms/1177704472.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL https://api.semanticscholar.org/CorpusID:11758569.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows.

ArXiv, abs/1505.05770, 2015. URL https://api.semanticscholar.org/CorpusID:12554042.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. doi: 10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

Surojit Saha, Sarang C. Joshi, and Ross T. Whitaker. Matching aggregate posteriors in the variational autoencoder. In *ICPR (6)*, pages 428–444, 2024. URL https://doi.org/10.1007/978-3-031-78172-8_28.

Shangquan Sun, Wenqi Ren, Jingzhi Li, Rui Wang, and Xiaochun Cao. Logit standardization in knowledge distillation. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15731–15740, 2024. URL https://api.semanticscholar.org/CorpusID:268247468.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

Natasa Tagasovska, Damien Ackerer, and Thibault Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/15e122e839dfdaa7ce969536f94aecf6-Paper.pdf.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkL7n1-0b.

Jakub Tomczak and Max Welling. Vae with a vampprior. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/tomczak18a.html.

Aaron Van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] See Section 2 and Section 3 and Appendix A.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] See Supplementary Material.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] See Appendix A.
   (b) Complete proofs of all theoretical results. [Yes] See Appendix B.1
   (c) Clear explanations of any assumptions. [Yes] Appendix A.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] See Supplementary Material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] See Appendix E
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] See Appendix E
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] See Section 1

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

A Assumptions for Theorem 3.1

We mainly borrow the terminologies and assumptions in Rosenblatt (1956) which is the pioneering work in measuring deviations of density function estimates. Note that Rosenblatt (1956) studied the asymptotic distribution of the quadratic functional $\int [f_n(t) - f(t)]^2 a(t)dt$ under appropriate weight function a and conditions as sampling size n approaches to infinity. In inequality (5), we just saw that the weight function is the reciprical of prior density in our case. Assumptions $\mathbf{A1} - \mathbf{A4}$ are listed as follows:

A1: The kernel function K is bounded, integrable, symmetric (about 0) and $\int K(t)dt = 1, \int t^2 K(t)dt < \infty, \int K^2(t)dt < \infty$. Also, K either (a) is supported on an closed and bounded interval $[-B, B]$ and is absolutely continuous on $[-B, B]$ with derivative K' or (b) is absolutely continuous on the whole real line with derivative K' satisfying $\int |K'(t)|^k dt < \infty, k = 1, 2$. Moreover,

$$\int_{t \geq 3} |t|^{\frac{3}{2}} [\log (\log |t|)]^{\frac{1}{2}} [|K'(t)| + |K(t)|]dt < \infty$$

A2: The underlying density f is continuous, positive and bounded.

A3: Squared density $f^{1/2}$ is absolutely continuous and its derivative is bounded in absolute value.

A4: The second derivative f'' exists and is bounded.

We can see that the Gaussian kernel satisfies those assumptions, indicating that the kernel density estimation theory also works for Gaussian VAE. Similarly, most priors and posteriors we listed in section 1 lie in conditions $\mathbf{A1} - \mathbf{A4}$, demonstrating the promising applications of kernel estimate theory in variants of VAE.

B Proofs

B.1 Proof of Lemma 3.2

For simplicity, we first consider the following constraints

$$K \geq 0, \quad \int K(x)dx = 1, \quad \int xK(x)dx = 0, \int x^2 K(x)dx = \frac{1}{5}$$

By the method of undermined multipliers, it's equivalent to minimize the following target functional without constraints:

$$\int K^2(x) + aK(x) + cz^2 K(x)dz$$

with simplified constraints above and a, c are undermined real coefficients. We ignore the term $xK(x)$ as it does not contribute to the unconstrained target function now.

For fixed x, denote $y(K) = K^2 + aK + cx^2 K, (K \geq 0)$. Note that the quadratic function $y(K)$ achieves minimum when $K = -\frac{c}{2}x^2 - \frac{a}{2}$.

It follows that $y(K)$ is minimized subject to $K \geq 0$ by

$$K(x) = \begin{cases} -\frac{c}{2}x^2 - \frac{a}{2} & -\frac{c}{2}x^2 - \frac{a}{2} \geq 0 \\ 0 & -\frac{c}{2}x^2 - \frac{a}{2} < 0 \end{cases}$$

We can rewrite it as

$$K(x) = \begin{cases} A(B^2 - x^2) & |x| \leq B \\ 0 & \text{otherwise} \end{cases}$$

for some number A, B. By simplified assumptions $\int xK(x)dx = 0, \int x^2K(x)dx = \frac{1}{5}$, we can find that $A = \frac{3}{4}, B = 1$.

Under general moment conditions in Lemma 3.2, optimal K^* can be written as

$$K^*(x) = \begin{cases} \frac{3}{4r}\left(1 - \left(\frac{x-\mu}{r}\right)^2\right) & |x - \mu| \leq r \\ 0 & \text{otherwise} \end{cases}$$

by location-scale formula. In literature Epanechnikov (1969), this kernel is called Epanechnikov kernel. We put $1/5$ in front of the constraint of second moment in order to make the resulting support interval cleaner, which won't change the optimal kernel. The corresponding optimal value of $I(K)$ is $I(K^*) = \frac{3}{5r}$.

Plots of Standard Epanechnikov kernel and Guassian kernel

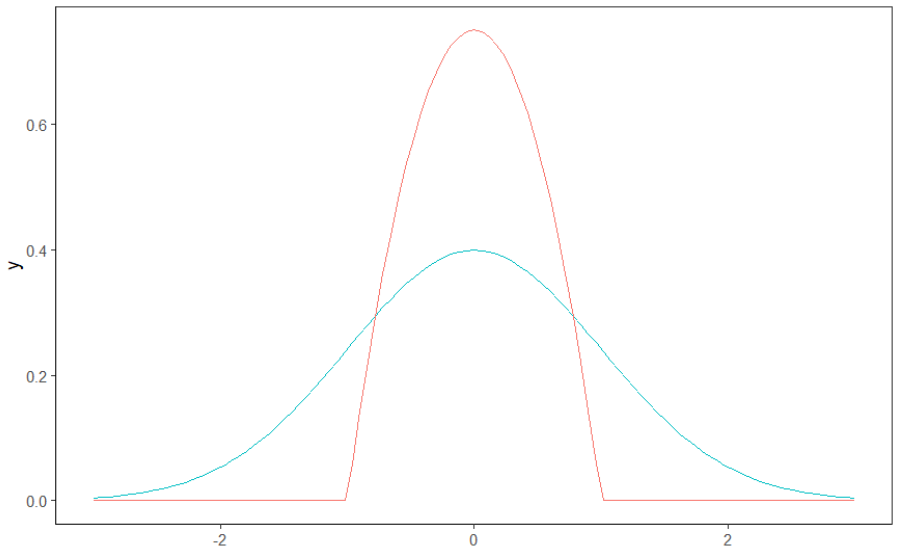


Figure S3: Red curve: Standard Epanechnikov kernel. Green curve: Standard Gaussian kernel.

B.2 Theoretical support for Algorithm 2

Given $U_1, U_2, U_3 \overset{i.i.d.}{\sim} \text{Unif}[-1, 1]$, we only need to show the density of median(U_1, U_2, U_3) is standard Epanechnikov kernel.

Denote $Y = \text{Median}(U_1, U_2, U_3)$, we have

$$\begin{aligned} P(Y \leq t) &= P(\text{Median}(U_1, U_2, U_3) \leq t) \\ &= P(\text{Exactly two of } U_1, U_2, U_3 \text{ are less than } t) + P(\text{All of } U_1, U_2, U_3 \text{ are less than } t) \\ &= \binom{3}{2}\left(\frac{1+t}{2}\right)^2\left(\frac{1-t}{2}\right) + \binom{3}{3}\left(\frac{1+t}{2}\right)^3 \\ &= \frac{1}{2} + \frac{3}{4}t - \frac{t^3}{4} \end{aligned}$$

Then the density $f_Y(t)$ of Y is

$$f_Y(t) = \frac{3}{4} - \frac{3}{4}t^2 = \frac{3}{4}(1 - t^2)$$

which is essentially the standard Epanechnikov kernel.

C Model architecture of EVAE

C.1 MNIST and Fashion-MNIST

We used fully convolutional architectures with 4×4 convolutional filters for both encoder and decoder in EVAE and VAE, as described following. All convolutions in the encoder and decoder employed SAME padding.

We resized images in MNIST and Fashion-MNIST from 28×28 to 32×32 at beginning. In the last conv layer, the sigmoid activation function was used to restrict the range of output as we assumed Bernoulli type model of $p_\theta(\mathbf{x}|\mathbf{z})$ and the binary cross entropy loss employed used (reduction to sum). Dimensions d_z for latent space : $\{8, 16, 32, 64\}$

Encoder q_ϕ:

$$
\begin{aligned}
x \in \mathbb{R}^{32\times 32} &\to \text{32 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{64 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{128 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{256 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{Fully connected } (1 * 1 * 256 \times d_z) \text{ for each parameters}
\end{aligned}
$$

Decoder p_θ:

$$
\begin{aligned}
z \in \mathbb{R}^{d_z \times d_z} &\to \text{Fully connected } (d_z \times 1 * 1 * 256) \\
&\to \text{128 ConvTran, Stride 1} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{64 ConvTran, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{32 ConvTran, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{1 ConvTran, Stride 2} \to \text{Sigmoid}
\end{aligned}
$$

C.2 CIFAR-10

Again, we used fully convolutional architectures with 4×4 convolutional filters for both encoder and decoder in EVAE and VAE for CIFAR-10 model. In encoder, we employed a layer of Adaptive Average pool filter. Other settings are the same with MNIST and Fashion-MNIST.

Encoder q_ϕ:

$$
\begin{aligned}
x \in \mathbb{R}^{32\times 32} &\to \text{32 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{64 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{128 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{256 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{AdaptiveAvgPool2d} \\
&\to \text{Fully connected } (1 * 1 * 256 \times d_z) \text{ for each parameters}
\end{aligned}
$$

Decoder p_θ:

$$
\begin{aligned}
z \in \mathbb{R}^{d_z \times d_z} &\to \text{Fully connected } (d_z \times 1 * 1 * 256) \\
&\to \text{128 ConvTran, Stride 1} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{64 ConvTran, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{32 ConvTran, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{3 ConvTran, Stride 2} \to \text{Sigmoid}
\end{aligned}
$$

C.3 CelebA

For CelebA dataset, we used 5×5 convolutional filters for both encoder and decoder in EVAE and VAE. Simiar to CIFAR-10, we employed a layer of Adaptive Average pool filter before the fully connected layer in encoder. We first scaled images with Center Crop to 140×140 and resized them to 64×64.

Encoder q_ϕ:

$$
\begin{aligned}
x \in \mathbb{R}^{64 \times 64} &\to \text{64 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{128 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{256 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{512 Conv, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{AdaptiveAvgPool2d} \\
&\to \text{Fully connected } (1 * 1 * 512 \times d_z) \text{ for each parameters}
\end{aligned}
$$

Decoder p_θ:

$$
\begin{aligned}
z \in \mathbb{R}^{d_z \times d_z} &\to \text{Fully connected } (d_z \times 8 * 8 * 512) \\
&\to \text{256 ConvTran, Stride 1} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{128 ConvTran, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{64 ConvTran, Stride 2} \to \text{BatchNorm} \to \text{ReLU} \\
&\to \text{3 ConvTran, Stride 2} \to \text{Sigmoid}
\end{aligned}
$$

D DCGAN structure (latent distribution learning) for MNIST and CIFAR-10

Generator G:

$$
\begin{aligned}
z \in \mathbb{R}^{B \times d_z} &\to \text{Linear}(d_z, \text{ 64 * 2}) \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear}(\text{64 * 2, 64 * 4}) \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear}(\text{64 * 4, 64 * 8}) \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear}(\text{64 * 8}, d_z) \to \text{BatchNorm} \to \text{Generated latent vector}
\end{aligned}
$$

Discriminator D:

$$
\begin{aligned}
\text{Latent vector } \tilde{z} \in \mathbb{R}^{B \times d_z} &\to \text{Linear}(d_z, 64) \to \text{GeLU} \\
&\to \text{Linear(64, 64 * 2)} \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear(64 * 2, 64 * 4)} \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear(64 * 4, 64 * 8)} \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear(64 * 8, 1)} \to \text{Sigmoid}
\end{aligned}
$$

Note that the real latent vector is the encoded mean parameter of posterior and the fakte latent vector is generated by the generator G. The discriminator in our case will be used to judge whether or not the latent vector is the actual mean parameter.

E Datasets and Training details

We list details for each benchmark dataset in following table

Datasets	# Training samples	# Hold-out samples	Original image size
MNIST	60000	10000	28*28
Fashion-MNIST	60000	10000	28*28
CIFAR-10	50000	10000	32*32
CelebA	162770	19867	178*218

Note that for MNIST,Fashion-MNIST and CIFAR-10, we used default splittings of training sets and testing sets provided in Pytorch (torchvision.datasets). For CelebA, we used default validation set as hold-out samples.

As to the training details, we used same training parameters for all algorithms and datasets, as described in following table

Latent space dimensions d_z	8,16,32,64
Optimizer	Adam with learning rate 3e-4
Batch size	100
Epochs	50

Calculation of Sharpness

We follow the way in Tolstikhin et al. (2018) in calculating the sharpness of an image. For each generated image, we first transformed it into grayscale and convolved it with the Laplace filter $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$, computed the variance of the resulting activations and took the average of all variances. The resulting number is denoted as sharpness (larger is better). The blurrier image will have less edges. As a result, the variance of activations will be small as most activations will be close to zero. Note that we averaged the sharpness of all reconstructed images from hold-out samples for each dataset.

Binomial test for two models

If EVAE and VAE have similar performance in FID score, the probability that EVAE has lower FID score should be 0.5 in each independent experiment. (Same hypothesis for sharpness). However, according to Table 5 and 6, EVAE wins 15 experiments for FID score among all datasets. The p-value of winning 15 experiments under null hypothesis is

$$P(X \geq 15) = \binom{16}{16}(0.5)^{16} + \binom{16}{15}(0.5)^{16} \approx 2.6 \times 10^{-4} < 0.05.$$

P-value is smaller than 0.05 significance level thus EVAE significantly outperforms VAE in FID. Similar calculation can be applied to sharpness, whose p-value of winning 12 experiments is $P(X \geq 15) = \sum_{i=15}^{16} \binom{16}{i} 0.5^{16} \approx 2.6 \times 10^{-4} < 0.05$ and we achieved the same conclusion for the significance of EVAE's superiority in sharpness.

Training details for Figure 2

To generate unconditional images, we need to train a DCGAN for learned posterior from stage 1. For MNIST, we set the latent dimension be 32. As to CIFAR10, we set the latent dimention be 64 to cater to the more complicated latent space structure of RGB images. Other hyper parameters are the same in table above. The network structures are described in Appendix D. GeLU represents the Gaussian Error Linear Unit activation function (Hendrycks and Gimpel, 2016).

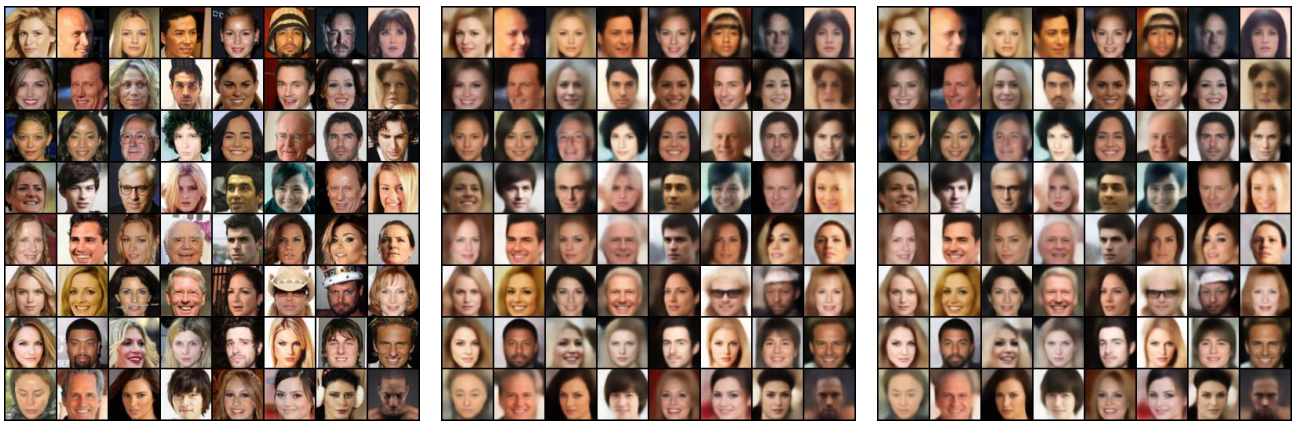
F Sampled reconstructed images

(a) Real images (b) VAE reconstructed images (c) EVAE reconstructed images

Figure S4: (a) Sampled real images from hold-out samples in MNIST (b) Reconstructed images by VAE. (c) Reconstructed images by EVAE. Dimension $d_z = 64$ for both models. See section C for Model architectures.

(a) Real images (b) VAE reconstructed images (c) EVAE reconstructed images

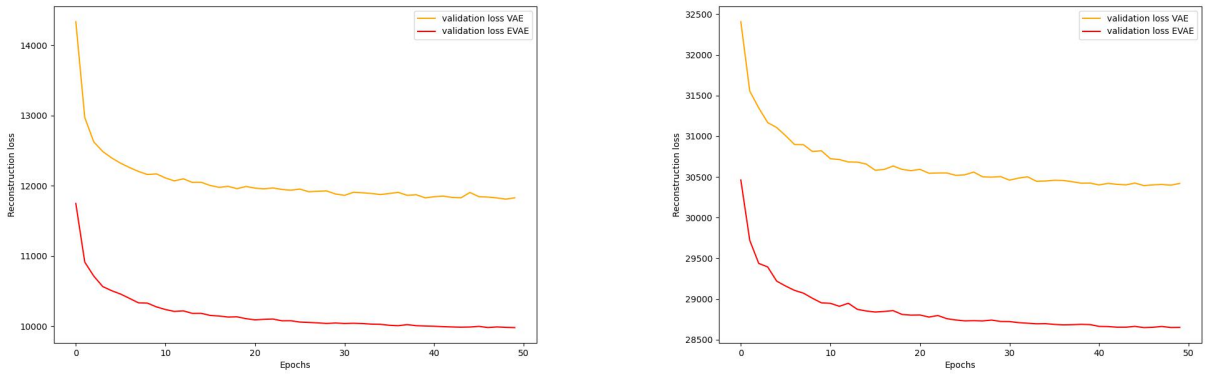
Figure S5: (a) Sampled real images from hold-out samples in Fashion-MNIST (b) Reconstructed images by VAE. (c) Reconstructed images by EVAE. Dimension $d_z = 64$ for both models.



(a) Real images (b) VAE reconstructed images (c) EVAE reconstructed images

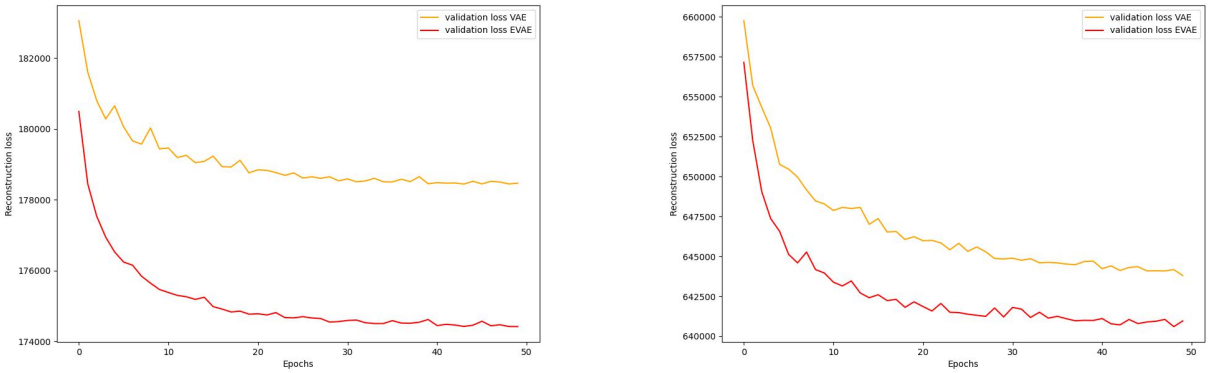
Figure S6: (a) Sampled real images from hold-out samples in CelebA (b) Reconstructed images by VAE. (c) Reconstructed images by EVAE. Dimension $d_z = 64$ for both models.

G Validation curves



(a) MNIST (b) Fashion-MNIST

Figure S7: Reconstruction validation loss curves as function of Epochs. ($d_z = 64$). Red curve is for EVAE and yellow one represents VAE. Binary cross entropy loss is reduced to sum for each batch (a) MNIST (b) Fashion-MNIST.



(a) CIFAR-10 (b) CelebA

Figure S8: Reconstruction validation loss curves as function of Epochs. ($d_z = 64$). Red curve is for EVAE and yellow one represents VAE. Binary cross entropy loss is reduced to sum for each batch (a) CIFAR-10 (b) CelebA.

H Downstream classification

In the downstream classification task, we first encode the input image into latent representation by the learned encoder from VAE or EVAE. The mean output from encoder part is used as latent representation. After that, a classifier network is fit to classify the digit in MNIST dataset. The learning rate, epochs, training/ testing splitting follow the details in Appendix E. The crossentropy loss is used since this is a multiclass classification problem. The classifier network structure follows as:

Classifier:

$$
\begin{aligned}
\text{Latent representation } \tilde{z} \in \mathbb{R}^{B\times d_z} &\to \text{Linear}(d_z, 64) \to \text{GeLU} \\
&\to \text{Linear(64, 64 * 2)} \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear(64 * 2, 64 * 4)} \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear(64 * 4, 64 * 8)} \to \text{BatchNorm1d} \to \text{GeLU} \\
&\to \text{Linear(64 * 8, 10)}
\end{aligned}
$$

The results in Table 3 are based on three times running.

I More experiment details on EVAE

I.1 Benchmark datasets

For comparison, we applied a classical CNN architecture (Appendix C) in encoding and decoding part for all datasets. Consequently, the main differences between EVAE and VAE in implementation stem from the resampling step and target function in training procedure. All FID scores and sharpness are based on hold-out samples. We also evaluated VAE and EVAE with different dimensions d_z of latent spaces. Table 6 summarize the results on CIFAR10 adn CelebA datasets with uniform prior in EVAE. More results for MNIST and Fashion-MNIST can be found in Table 5

Table 5: VAE and EVAE results on MNIST and Fashion-MNIST datasets

Datasets	MNIST				Fashion-MNIST			
d_z	VAE		EVAE		VAE		EVAE	
	FID	Sharpness	FID	Sharpness	FID	Sharpness	FID	Sharpness
8	15.70	0.0211	14.74	0.0243	41.47	0.0157	36.18	0.0176
16	11.93	0.0252	10.49	0.0305	38.85	0.0159	26.71	0.0216
32	11.97	0.0249	7.92	0.0338	38.74	0.0163	18.47	0.0251
64	11.97	0.0246	5.13	0.0360	39.31	0.0161	12.02	0.0267

Table 6: VAE and EVAE results on CIFAR10 and CelebA datasets

Datasets	CIFAR-10				CelebA			
d_z	VAE		EVAE		VAE		EVAE	
	FID	Sharpness	FID	Sharpness	FID	Sharpness	FID	Sharpness
8	230.1	0.0353	226.6	0.0337	198.7	0.0147	198.7	0.0154
16	181.4	0.0350	164.2	0.0355	152.5	0.0148	149.7	0.0154
32	149.7	0.0348	123.5	0.0359	110.1	0.0156	97.5	0.0162
64	144.7	0.0355	79.7	0.0369	77.2	0.0163	65.3	0.0167

I.2 EVAE with guassian prior

The experiment details are the saem with section C. The only difference is we replaced uniform prior with gaussian prior.

Table 7: VAE and EVAE results on MNIST and Fashion-MNIST datasets

Datasets	MNIST				Fashion-MNIST			
d_z	VAE		EVAE		VAE		EVAE	
	FID	Sharpness	FID	Sharpness	FID	Sharpness	FID	Sharpness
8	16.14	0.0217	15.22	0.0246	42.22	0.0157	35.89	0.0181
16	12.11	0.0248	10.25	0.0308	38.88	0.0167	26.65	0.0218
32	12.43	0.0247	8.22	0.0335	39.14	0.0157	18.52	0.0246
64	11.73	0.0248	5.74	0.0364	38.96	0.0159	12.18	0.0271

Table 8: VAE and EVAE results on CIFAR10 and CelebA datasets

Datasets	CIFAR-10				CelebA			
d_z	VAE		EVAE		VAE		EVAE	
	FID	Sharpness	FID	Sharpness	FID	Sharpness	FID	Sharpness
8	228.92	0.0353	218.84	0.0355	194.36	0.0149	194.62	0.0145
16	183.03	0.0360	163.84	0.0355	144.88	0.0155	147.78	0.0157
32	146.80	0.0359	122.17	0.0364	106.02	0.0160	102.73	0.0163
64	141.22	0.0353	79.06	0.0356	80.60	0.0161	65.79	0.0164

I.3 Time efficiency of EVAE

One advantage of kernel posterior proposed in equation (11) is that the quadratic functional $I(K)$ has a closed form for many distributions, while the closed form of KL divergence in standard ELBO can be hard to derive when we want to use complicated posterior and prior. For example, the KL divergence becomes piecewise for the uniform prior and posterior. In those cases, time-consuming Monte Carlo simulations might be needed.

To explore the time efficiency of EVAE, we performed additional experiments in Table I.3 to compare the average and standard error of the training time (in seconds) per epoch (10 epochs total for each experiment) for VAE and EVAE in CIFAR10 and CelebA-64.

Table 9: Training time of EVAE and VAE per epoch in seconds

Latent space dimension d_z	CIFAR10		CelebA-64	
	VAE(std)	EVAE(std)	VAE(std)	EVAE(std)
$d_z = 8$	7.21(0.28)	7.9(0.19)	208.94(5.95)	208.42(4.33)
$d_z = 16$	7.22(0.43)	7.99(0.19)	209.10(5.94)	210.88(5.35)
$d_z = 32$	7.9(0.6)	8.87(0.31)	208.02(3.22)	210.23(2.81)
$d_z = 64$	7.56(0.4)	8.56(0.63)	206.85(5.24)	207.72(5.66)

In general, we found that EVAE has the comparable time efficiency to VAE. The slight increase in training time for EVAE results from the sampling process of the Epanechnikov kernel, which requires a few more samples to achieve the Epanechnikov density, as described in Algorithm 2. The difference in training time is negligible when the latent space dimension is large, which facilitates the application of EVAE in high-resolution datasets.

I.4 PSNR comparison for reconstruction quality

Table 10: PSNR comparison for EVAE and other baselines on CIFAR10

Models	PSNR (dB)
PSNR (dB)	18.75
EVAE	21.08
β-VAE($\beta = 0.1$)	20.87
β-VAE($\beta = 5$)	16.57
IWAE($k = 5$)	9.88
WAE-MMD	9.8
HVAE	17.15
VQVAE	16.86

For illustration, we conducted extra experiments comparing PSNR (the higher the better) of different baselines on CIFAR10. We can see that EVAE still outperforms other baselines under PSNR, implying EVAE has higher reconstruction quality in general