Negative Binomial Variational Autoencoders for Overdispersed Latent Modeling

Supplementary Material

A. Derivation of KL Term

We derive an analytical expression for the KL divergence between two negative binomial distributions under the assumption that the encoder does not modify the dispersion parameter (i.e., $\delta_r = 1$). the univariate negative binomial distribution, given dispersion r and success probability p, is defined as:

$$\text{NB}(z; r, p) = \binom{z+r-1}{z} (1-p)^z p^r.$$

Substituting this into the KL divergence for a single z yields:

$$\begin{aligned}
&\mathcal{D}_{\text{KL}}(q||p) = \mathbb{E}_{z\sim q}[\log \frac{q}{p}] \\
&= \mathbb{E}_{z\sim q}\left[\log \frac{\binom{z+r\delta_r-1}{z}(1-p\delta_p)^z(p\delta_p)^{r\delta_r}}{\binom{z+r-1}{z}(1-p)^z p^r}\right] \\
&= \mathbb{E}_{z\sim q}\left[r\log\frac{p\delta_p}{p} + z\log\left(\frac{1-p\delta_p}{1-p}\right)\right] \\
&= r\log\frac{p\delta_p}{p} + \mathbb{E}_{z\sim q}\left[z\log\left(\frac{1-p\delta_p}{1-p}\right)\right] \\
&= r\log\delta_p + \log\left(\frac{1-p\delta_p}{1-p}\right)\mathbb{E}_{z\sim q}[z] \\
&= r\log\delta_p + r\frac{1-p\delta_p}{p\delta_p}\log\left(\frac{1-p\delta_p}{1-p}\right) \\
&= r\left[\log\delta_p + \frac{1-p\delta_p}{p\delta_p}\log\left(\frac{1-p\delta_p}{1-p}\right)\right] \\
&= rg(p,\delta_p).
\end{aligned}$$

Taking the logarithm of the terms involve binomial coefficients and computing the expectation with respect to the posterior makes the KL divergence intractable. As a result, Monte Carlo sampling or variational approximation techniques are typically required, which often introduce high variance in the gradient estimates or rely on additional approximating assumptions and can lead to unstable or biased training. To make the expression tractable, we introduce a simplifying assumption: $\delta_r = 1$, i.e., the encoder does not adjust the prior parameter r, and thus the posterior and prior share the same dispersion parameter. This assumption is reasonable because the NB distribution is parameterized by both r and p. Therefore, even when r is fixed, we can still adjust the distribution (i.e., its mean and variance) by varying p. This leads to a closed-form approximation:

$$\mathcal{D}_{\text{KL}}(q||p) = r\left[\log\delta_p + \frac{1-p\delta_p}{p\delta_p}\log\left(\frac{1-p\delta_p}{1-p}\right)\right],$$

which we denote as $rg(p, \delta_p)$, where:

$$g(a,b) := \log b + \frac{1-ab}{ab}\log\left[\frac{1-ab}{1-a}\right],$$
$$a \in (0,1), \quad b > 0.$$

This expression is simple, interpretable and has useful boundary properties. When $\delta_p = 1$ (i.e., the encoder does not shift p), $g(a, 1) = 0$, and the KL divergence vanishes. As $ab \to 0$ (i.e., posterior sparsity increases), the KL grows rapidly, penalizing excessive deviation from the prior. This behavior mirrors that of $\mathcal{P}$-VAE, which strongly discourages low-rate posterior collapse. While $\mathcal{P}$-VAE already provides an elegant analysis of sparsity through its KL structure, we do not emphasize this aspect in the main text. However, our formulation shares the same desirable sparsity behavior: when δ_p approaches 0, the KL diverges, discouraging extreme posterior sparsification. Moreover, our formulation retains an analytical form even for overdispersed distributions, enabling tractable training without Monte Carlo estimation.

Similar to the $\mathcal{P}$-VAE [46], which analyzes the behavior of its KL divergence near the prior via a Taylor expansion of the function $f(\delta_r) = 1 - \delta_r + \delta_r \log \delta_r$, we perform a similar analysis for the closed-form KL term in NegBio-VAE. To better understand the behavior of the closed-form KL divergence near the prior, we expand $g(a, b)$ at $b = 1+\epsilon$, with $\epsilon \ll 1$:

$$g(a, 1+\epsilon) \approx \frac{a}{2(1-a)}\epsilon^2 + \mathcal{O}(\epsilon^3). \tag{6}$$

Thus, when $\delta_p = 1 + \epsilon$, the KL becomes:

$$\mathcal{D}_{\text{KL}} \approx r \cdot \frac{a}{2(1-a)}\epsilon^2.$$

This reveals that, like in $\mathcal{P}$-VAE, the KL divergence grows quadratically near the prior, encouraging smooth and stable optimization. However, our formulation provides a tunable growth rate via the parameter $a = p$, allowing more flexible control over sparsity regularization. Unlike Poisson VAEs, which assume equal mean and variance, our negative binomial model accommodates overdispersion and remains analytically tractable—enabling stable training without Monte Carlo approximation. These properties make our approach better suited for modeling realistic, variable spike-based neural activity.

B. Implementation Details

We include all the implementation details in this section, including the sampling techniques and detailed experimental settings.

B.1. Sampling Techniques

We adopt two sampling techniques for our model, while we have introduced the main idea in the main text, for completeness, we include the details here:

(1) **Gumbel-Softmax Relaxation** This method approximates discrete Poisson sampling using continuous relaxation.

1. Limit the maximum count value to $Z_{\max}$.
2. Compute the log-probability for $z = 0, 1, \ldots, Z_{\max}$,

$$\log \text{Poi}(z) = z \log \lambda - \lambda - \log \Gamma(z+1).$$

3. For each z, generate noise $\epsilon_z \sim \text{Gumbel}(0, 1)$.
4. Apply the Gumbel-Softmax trick with temperature τ,

$$\tilde{z} = \sum_{z=0}^{Z_{\max}} z \cdot \text{softmax}\left(\frac{\log \text{Poi}(z) + \epsilon_z}{\tau}\right),$$

where $\tau \to 0$ recovers discrete sampling.

The proof of this reparameterization can be found in Jang et al. [18], and will not be repeated here.

(2) **Continuous-Time Simulation** This method models Poisson processes with intensity λ on $[0, 1]$ using exponentially distributed inter-arrival times.

1. Sample inter-arrival times from an exponential distribution:

$$\{s_i\}_{i=1}^{M} \sim \text{Exponential}(\lambda),$$

where M is a sufficiently large integer, the exponential distribution is easily reparameterized and PyTorch contains an implementation.

2. Accumulate inter-arrival times:

$$S_n = \sum_{i=1}^{n} s_i, \quad 1 \le n \le M.$$

3. Soft count of events:

$$\tilde{z} = \sum_{n=1}^{M} \sigma\left(\frac{1 - S_n}{\tau}\right),$$

where $\tau \to 0$ recovers discrete sampling.

This reparameterization exploits the relationship between the Poisson distribution and the Poisson process. We can generate Poisson counts from $\text{Poi}(\lambda)$ by counting events on a homogeneous Poisson process with intensity λ over the interval $[0, 1]$.

To verify that both proposed reparameterization methods can successfully generate valid count samples from the NB distribution, we generate 1000 samples from each method using $r = 20$, $p = 0.5$, and temperature $\tau = 0.1$, and the empirical distribution is shown in Fig. 5. Both methods produce plausible count distributions with unimodal structure and similar mean values, confirming that each method can successfully approximate NB samples in a differentiable manner. This validates their use as practical reparameterization techniques for NegBio-VAE.

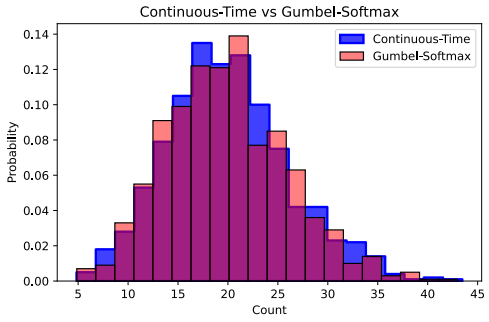


Figure 5. Empirical distributions of negative binomial samples generated using Continuous-Time Simulation and Gumbel-Softmax relaxation. Both methods successfully approximate count-valued outputs consistent with NB sampling behavior, validating their use as differentiable reparameterization strategies. Each method generates 1000 samples using parameters $r = 20$, $p = 0.5$, and temperature $\tau = 0.1$.

B.2. Experimental Implementation

In this section, we provide additional implementation details that complement the experiments described in the main text.

B.2.1. Datasets

We implement all data loading pipelines using PyTorch Lightning's LightningDataModule interface, ensuring consistent structure across datasets. For each dataset, we apply preprocessing transformations tailored to its modality and input requirements. All preprocessing logic is encapsulated in a shared function `get_transform()`, which dynamically composes transformations based on dataset type, grayscale conversion, data flattening, and augmentation flags.

- MNIST: Grayscale images are normalized to $[0, 1]$, using `transforms.ToTensor()` without additional augmentation. Images are optionally flattened if required by the encoder structure.
- Fashion-MNIST: Following the same preprocessing as MNIST, grayscale images are normalized to the $[0, 1]$ range using `transforms.ToTensor()` without any additional augmentation. Images are optionally flattened when required by the encoder architecture.
- CIFAR$_{16\times16}$: We use CIFAR-10 as a base dataset and

uniformly downsample all images to 16×16 resolution. Color images are normalized to $[-1, 1]$ using mean and standard deviation $(0.5, 0.5, 0.5)$ and are optionally augmented via horizontal flipping.

- CelebA-64: For CelebA-64, RGB face images are resized to 64×64 and normalized to the $[-1, 1]$ range using `transforms.ToTensor()` followed by `transforms.Resize(64)`. To ensure consistency across samples, no additional augmentation is applied. When required by the encoder architecture, images are optionally flattened or converted to latent representations.

B.2.2. Encoder and Decoder Architectures

Our model supports interchangeable encoder and decoder architectures to accommodate various data modalities and representation structures. Following the same setting in [46], we include linear, MLP, and convolutional-based designs.

- Linear Encoder: A single fully connected layer that maps flattened inputs to the latent space. Optionally applies weight normalization.
- MLP Encoder: A two-layer perceptron with an intermediate residual dense layer followed by a liner projection. This encoder supports flexible nonlinearity and is used when richer transformations are required from vectorized inputs.
- Convolutional Encoder: A two-stage convolutional network with ReLU activations, followed by flattening and a fully connected projection. It adapts the input channel size (1 for graysacle datasets, 3 for RGB), and autocomputes the flattening shape based on the dataset resolution. LayerNorm is optionally applied to the final latent layer.
- Linear Decoder: Mirrors the linear encoder with a fully connected layer projecting latent vectors to pixel space. Output is passed through either a Sigmoid or Tanh nonlinearity depending on the expected pixel scale.
- MLP Decoder: A three-layer feedforward network with residual blocks and configurable nonlinearity (e.g., Swish), designed for richer reconstructions from compact latent codes. The final layer uses Sigmoid or Tanh.
- Convolutional Decoder: Used in image-based settings, this decoder first expands latent vectors through a fully connected layer into a low-resolution feature map, then applies transposed convolutions to upscale to the desired image size. The initial size is determined by dataset type (e.g., 7×7 for MNIST, 4×4 for $\text{CIFAR}_{16\times16}$).

B.2.3. Shattering Dimensionality

To quantitatively evaluate the geometry of the learned latent space, we compute the shattering dimensionality following prior work. Specifically, we measure how well the latent space supports linear separation across all balanced binary label partitions. Concretely, given a label set such as digits 0-9, we enumerate all disjoint splits into two non-overlapping and balanced class groups, where each partition defines a binary classification task. For each split, we relabel samples in one group as class 0 and those in the other group as class 1, producing binary-labeled data. For a dataset with 10 classes (e.g., MNIST), we generate all possible balanced, disjoint 5-vs-5 class splits. This results in a total of 252 unique binary classification tasks, corresponding to all combinations of 5 classes out of 10 without regard to class order or labeling symmetry. A linear classifier (e.g., logistic regression) is then trained on latent representations from a subset of the training data. The classification accuracy is computed on the validation set, and the final shattering dimensionality is taken as the average accuracy across all 252 tasks. The implementation uses the `itertools.combinations` function to enumerate all unique 5-class subsets, and constructs their complementary partitions to define the 5-vs-5 classification groups.

B.2.4. SSIM Computation Details

To further assess the perceptual quality of reconstructed images, we employ the structural similarity index (SSIM) as a complementary metric. SSIM evaluates image similarity by considering changes in luminance, contrast, and structural information between two images. Given two images x and y, SSIM is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)},$$

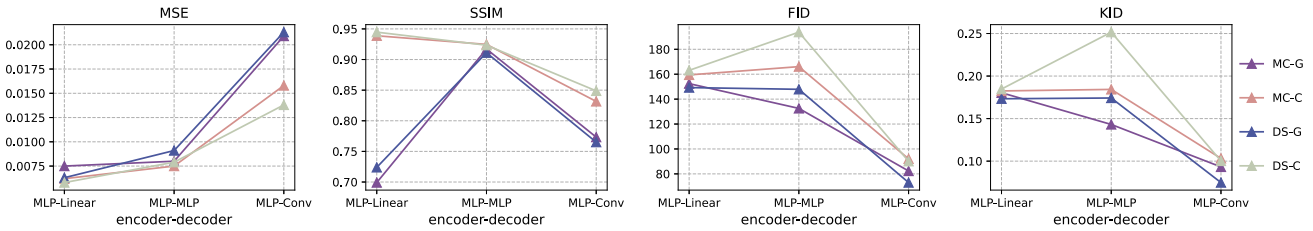
where μ_x and μ_y denote the mean intensities, σ_x^2 and σ_y^2 are the variances, and σ_{xy} represents the covariance between x and y. Constants C_1 and C_2 are used to stabilize the division. A higher SSIM value indicated greater perceptual similarity between the reconstructed and ground-truth images.

B.2.5. FID Computation Details

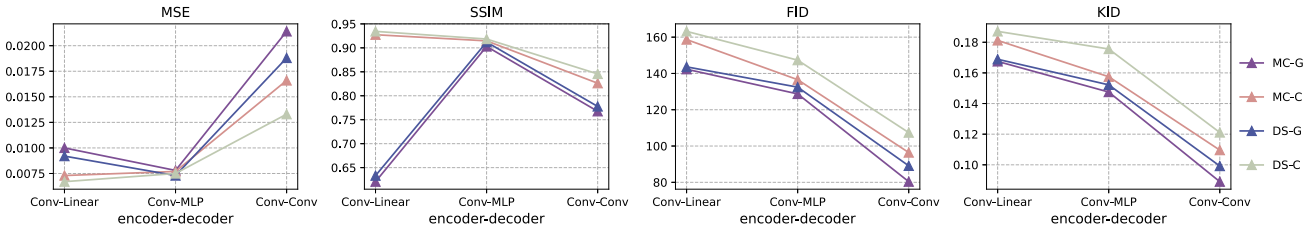
To quantitatively evaluate the visual fidelity and distributional similarity of generated images, we adopt the Fréchet Inception Distance (FID) as a standard evaluation metric. FID measures the distance between the real and generated image distributions in the feature space of a pretrained Inception network. Specifically, it assumes both distributions are Gaussian, and computes the Fréchet distance between them as:

$$\text{FID} = \|\mu_{\text{real}} - \mu_{\text{gen}}\|^2 + \text{Tr}\left(\Sigma_{\text{real}} + \Sigma_{\text{gen}} - 2(\Sigma_{\text{real}}\Sigma_{\text{gen}})^{1/2}\right),$$

where μ_{real}, Σ_{real} and μ_{gen}, Σ_{gen} denote the mean and covariance of the real and generated feature activations, respectively. A lower FID score indicates that the generated samples are more similar to the real data in terms of both image quality and diversity.



(a) MLP encoder



(b) Convolutional encoder

Figure 6. Ablation study of encoder-decoder architectures on MNIST with four variants of NegBio-VAE.

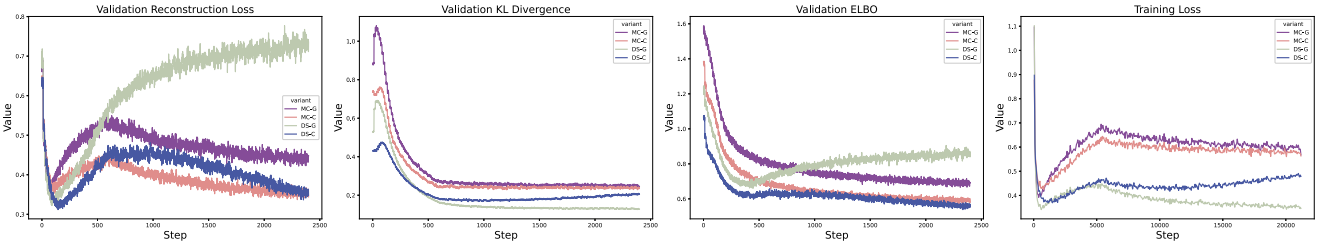


Figure 7. Comparison of four NegBio-VAE variants (MC-G, MC-C, DS-C, DS-G) across validation reconstruction loss, validation KL divergence, validation ELBO and training loss.

B.2.6. KID Computation Details

We also report the Kernel Inception Distance (KID) to evaluate the distributional alignment between real and generated images. Unlike FID, KID computes the squared Maximum Mean Discrepancy (MMD) between Inception representations of real and generated samples using a polynomial kernel. Formally, it is defined as:

$$\mathrm{KID} = \left\| \frac{1}{n_r} \sum_{i=1}^{n_r} \phi(x_i) - \frac{1}{n_g} \sum_{j=1}^{n_g} \phi(y_j) \right\|^2 ,$$

where $\phi(\cdot)$ denotes the feature embedding from a pretrained Inception network, and n_r, n_g are the numbers of real and generated samples, respectively. A smaller KID score indicated that the generated distribution is closer to the real one. Compare with FID, KID is unbiased and more reliable when computed with limited sample sizes.

C. Visualization of Image Reconstruction Results

To further evaluate the reconstruction performance of NegBio-VAE, we present reconstructed samples from multiple datasets in Fig. 8, Fig. 9, Fig. 10, and Fig. 11. As observed, our method accurately preserves fine-grained structural details across different datasets. For example, it retains the subtle gaps surrounding digits in the MNIST dataset and effectively reconstructs the contours and texture details of clothing in the Fashion-MNIST dataset. These observations demonstrate the strong capability of NegBio-VAE in modeling discrete structural information.

D. Visualization of Image Generation Results

We further provide qualitative image generation results in Fig. 12, Fig. 13, Fig. 14, and Fig. 15. The generated samples demonstrate that NegBio-VAE produces diverse and visually coherent outputs, effectively capturing meaningful variations within the data distribution. These results further confirm the strong generative ability of the model and the well-structured organization of its learned latent space.

Table 4. Evaluation of latent representations on MNIST. Higher accuracy and shattering dimensionality indicate more structured and generalizable latent.

Latent Dim	Model	Acc↑(N=200)	Acc↑(N=1000)	Acc↑(N=5000)	Acc↑(Shat. Dim.)
10	$\mathcal{G}$-VAE	$0.726_{\pm 0.0015}$	$0.798_{\pm 0.0020}$	$0.844_{\pm 0.0040}$	$0.851_{\pm 0.0050}$
	$\mathcal{L}$-VAE	$0.647_{\pm 0.0160}$	$0.733_{\pm 0.0080}$	$0.781_{\pm 0.0040}$	$0.811_{\pm 0.0070}$
	$\mathcal{C}$-VAE	$0.728_{\pm 0.0190}$	$0.812_{\pm 0.0060}$	$0.855_{\pm 0.0020}$	$0.856_{\pm 0.0090}$
	$\mathcal{P}$-VAE	$\underline{0.747}_{\pm 0.0180}$	$\mathbf{0.836}_{\pm 0.0030}$	$\mathbf{0.883}_{\pm 0.0040}$	$\mathbf{0.865}_{\pm 0.0080}$
	NegBio-VAE	$\mathbf{0.749}_{\pm 0.0150}$	$\underline{0.830}_{\pm 0.0010}$	$\underline{0.878}_{\pm 0.0020}$	$\underline{0.862}_{\pm 0.0080}$
50	$\mathcal{G}$-VAE	$0.819_{\pm 0.0090}$	$\mathbf{0.922}_{\pm 0.0030}$	$\mathbf{0.960}_{\pm 0.0020}$	$\underline{0.903}_{\pm 0.0070}$
	$\mathcal{L}$-VAE	$\underline{0.822}_{\pm 0.0080}$	$\underline{0.921}_{\pm 0.0020}$	$\mathbf{0.960}_{\pm 0.0030}$	$\underline{0.903}_{\pm 0.0060}$
	$\mathcal{C}$-VAE	$0.784_{\pm 0.0090}$	$0.888_{\pm 0.0040}$	$0.936_{\pm 0.0030}$	$0.887_{\pm 0.0060}$
	$\mathcal{P}$-VAE	$0.760_{\pm 0.0130}$	$0.897_{\pm 0.0030}$	$0.951_{\pm 0.0020}$	$0.872_{\pm 0.0060}$
	NegBio-VAE	$\mathbf{0.826}_{\pm 0.0070}$	$0.914_{\pm 0.0020}$	$\underline{0.952}_{\pm 0.0030}$	$\mathbf{0.904}_{\pm 0.0070}$
100	$\mathcal{G}$-VAE	$0.790_{\pm 0.0070}$	$\mathbf{0.914}_{\pm 0.0020}$	$\mathbf{0.958}_{\pm 0.0020}$	$0.890_{\pm 0.0050}$
	$\mathcal{L}$-VAE	$\underline{0.798}_{\pm 0.0090}$	$\underline{0.912}_{\pm 0.0020}$	$\mathbf{0.958}_{\pm 0.0020}$	$\underline{0.892}_{\pm 0.0070}$
	$\mathcal{C}$-VAE	$0.783_{\pm 0.0070}$	$0.896_{\pm 0.0030}$	$0.941_{\pm 0.0040}$	$0.886_{\pm 0.0070}$
	$\mathcal{P}$-VAE	$0.736_{\pm 0.0110}$	$0.888_{\pm 0.0020}$	$0.947_{\pm 0.0030}$	$0.862_{\pm 0.0070}$
	NegBio-VAE	$\mathbf{0.811}_{\pm 0.0050}$	$\underline{0.912}_{\pm 0.0010}$	$\underline{0.955}_{\pm 0.0030}$	$\mathbf{0.898}_{\pm 0.0060}$

E. Additional Experiments

This section presents additional experimental results, including the latent representation analysis, further evaluations on NegBio-VAE architectures variants, and a detailed analysis of the loss evolution during training.

E.1. Additional Results on Latent Analysis

Tab. 4 extends the shattering test results to different latent dimensions (10, 50, and 100) on MNIST. Across all configurations, NegBio-VAE consistently achieves the best performance, particularly under limited data ($N = 200$), demonstrating its superior sample efficiency and robustness. Notably, NegBio-VAE attains the highest shattering dimensionalities (0.862, 0.904, and 0.898 for latent dimensions 10, 50, and 100, respectively), indicating a more structured and stable latent space under randomized supervision. As the latent dimension increases, all models exhibit performance gains; however, NegBio-VAE maintains smoother improvement trends, suggesting stronger regularization and more biologically consistent representation learning.

E.2. Additional Results on VAE Architecture Variants

Fig. 6 presents an ablation study on different encoder-decoder architectures using the MNIST dataset. In this study, the latent dimension of all variants is fixed at 256, and both MLP and convolutional architectures are used as encoders. Experimental results show that for MC-based methods, the MLP encoder generally achieves the lowest reconstruction error (MSE), while the convolutional architecture achieves higher generation quality (i.e., lower FID). Notably, the combination of an MLP encoder and a convolutional decoder achieves the best balance between reconstruction and generation, outperforming purely linear or convolutional designs. Similar trends are observed for DS-based methods: the MLP encoder consistently achieves the lowest MSE, while the convolutional decoder achieves competitive or even superior FID scores.

E.3. Loss Dynamics Across NegBio-VAE Variants

Fig. 7 presents a comparison of the training dynamics for four NegBio-VAE variants across different loss terms (validation reconstruction loss, validation KL, validation ELBO and train loss). We observe notable differences in both the convergence rate and the smoothness of the trajectories, which reflect the influence of the KL estimation method and the reparametrization strategy. Overall, models with MC KL estimation (MC-G and MC-C) exhibit higher variance in the loss curves, with visible oscillations due to the stochastic nature of the Monte Carlo method (which introduces noise into the gradient updates as we have discussed in Sec. 4.1). In contrast, DS-based variants (DS-C and DS-G), which leverage closed-form KL computation via dispersion sharing, show smoother and more stable curves, suggesting better optimization stability. Comparing reparameterization strategies, models using continuous-time simulation (C) (DS-C and MC-C) tend to achieve lower reconstruction loss and faster ELBO convergence than their Gumbel-softmax (G). This suggests that the continuous-time approach offers a more expressive and stable mechanism for modeling spike-like latent representations. In particular, DS-C demonstrates the most stable and efficient convergence across all loss types, with consistently smooth

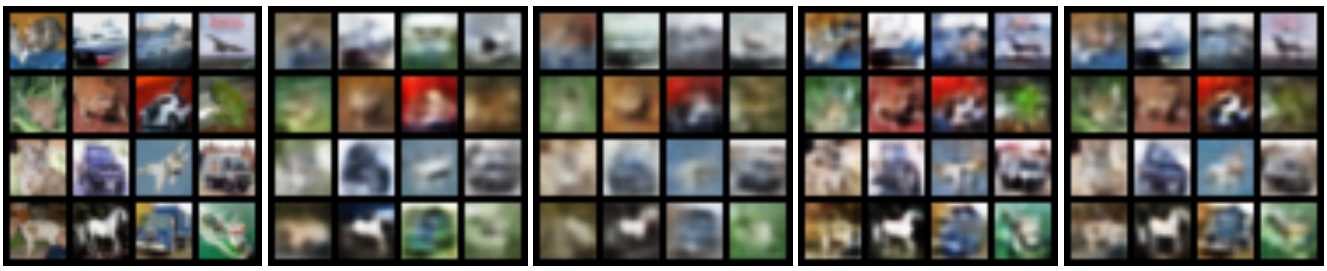
trajectories and lower final values.

(a) Input images (b) MC-G (c) MC-C (d) DS-G (e) DS-C

Figure 8. Reconstruction results on the MNIST dataset. The leftmost column shows the original images, while the remaining columns display the reconstructed images generated by NegBio-VAE.

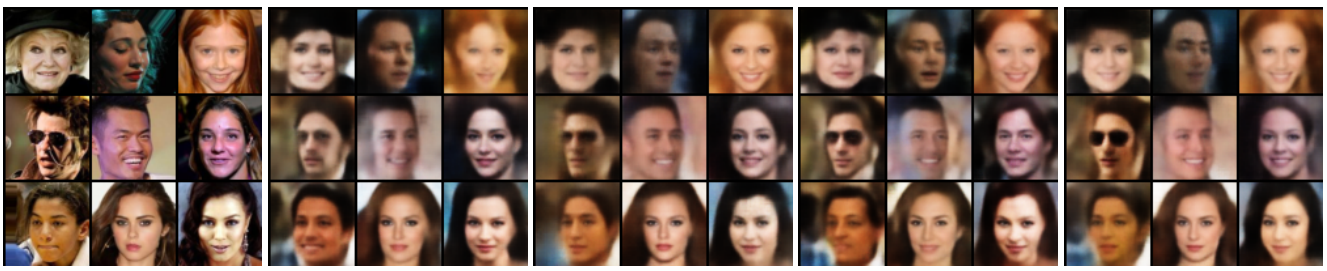
(a) Input images (b) MC-G (c) MC-C (d) DS-G (e) DS-C

Figure 9. Reconstruction results on the Fashion-MNIST dataset. The leftmost column shows the original images, while the remaining columns display the reconstructed images generated by NegBio-VAE.



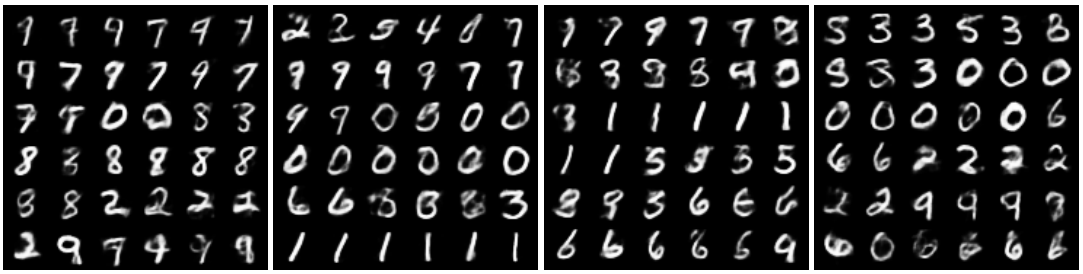
(a) Input images (b) MC-G (c) MC-C (d) DS-G (e) DS-C

Figure 10. Reconstruction results on the CIFAR$_{16\times16}$ dataset. The leftmost column shows the original images, while the remaining columns display the reconstructed images generated by NegBio-VAE.



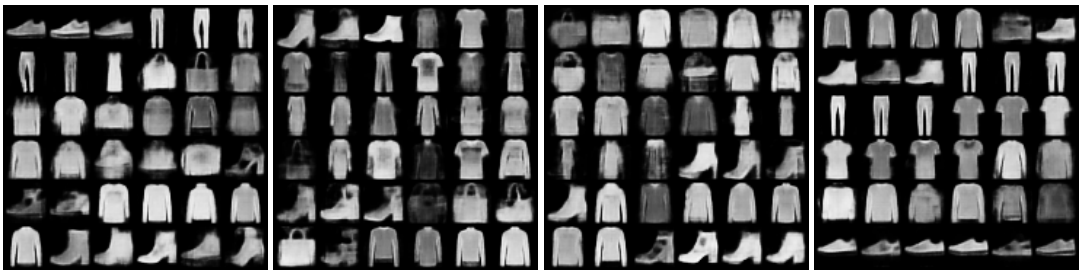
(a) Input images (b) MC-G (c) MC-C (d) DS-G (e) DS-C

Figure 11. Reconstruction results on the CelebA-64 dataset. The leftmost column shows the original images, while the remaining columns display the reconstructed images generated by NegBio-VAE.



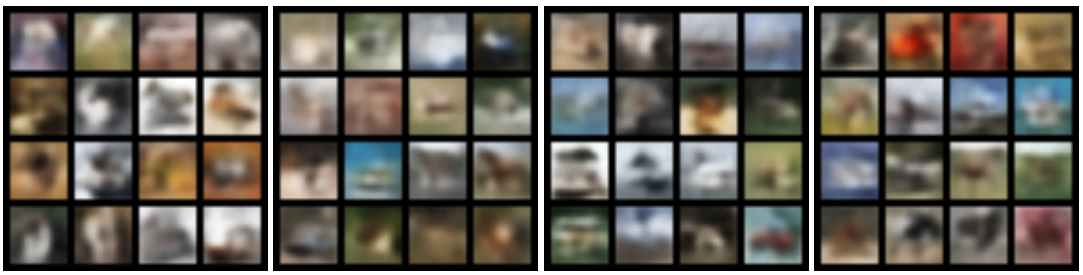
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 12. Randomly generated samples on the MNIST dataset using NegBio-VAE. Each image is generated from a different random latent variable z under identical model settings, illustrating the NegBio-VAE's ability to produce diverse and realistic samples.



(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 13. Randomly generated samples on the Fashion-MNIST dataset using NegBio-VAE. Each image is generated from a different random latent variable z under identical model settings, illustrating the NegBio-VAE's ability to produce diverse and realistic samples.



(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 14. Randomly generated samples on the CIFAR$_{16\times16}$ dataset using NegBio-VAE. Each image is generated from a different random latent variable z under identical model settings, illustrating the NegBio-VAE's ability to produce diverse and realistic samples.

(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 15. Randomly generated samples on the CelebA-64 dataset using NegBio-VAE. Each image is generated from a different random latent variable z under identical model settings, illustrating the NegBio-VAE's ability to produce diverse and realistic samples.