Unified Generation, Reconstruction, and Representation: Generalized Diffusion with Adaptive Latent Encoding-Decoding

Guangyi Liu [*1] **Yu Wang** [*2] **Zeyu Feng** [*2] **Qiyu Wu** [3] **Liping Tang** [1] **Yuan Gao** [4]
Zhen Li [5] **Shuguang Cui** [5] **Julian McAuley** [2] **Zichao Yang** [6]
Eric P. Xing [1 6] **Zhiting Hu** [2]

Abstract

The vast applications of deep generative models are anchored in three core capabilities—*generating* new instances, *reconstructing* inputs, and learning compact *representations*—across various data types, such as discrete text/protein sequences and continuous images. Existing model families, like variational autoencoders (VAEs), generative adversarial networks (GANs), autoregressive models, and (latent) diffusion models, generally excel in specific capabilities and data types but fall short in others. We introduce *Generalized Encoding-Decoding Diffusion Probabilistic Models* (EDDPMs) which integrate the core capabilities for broad applicability and enhanced performance. EDDPMs generalize the Gaussian noising-denoising in standard diffusion by introducing parameterized encoding-decoding. Crucially, EDDPMs are compatible with the well-established diffusion model objective and training recipes, allowing effective learning of the encoder-decoder parameters *jointly* with diffusion. By choosing appropriate encoder/decoder (e.g., large language models), EDDPMs naturally apply to different data types. Extensive experiments on text, proteins, and images demonstrate the flexibility to handle diverse data and tasks and the strong improvement over various existing models. Code is available at https://github.com/guangyliu/EDDPM

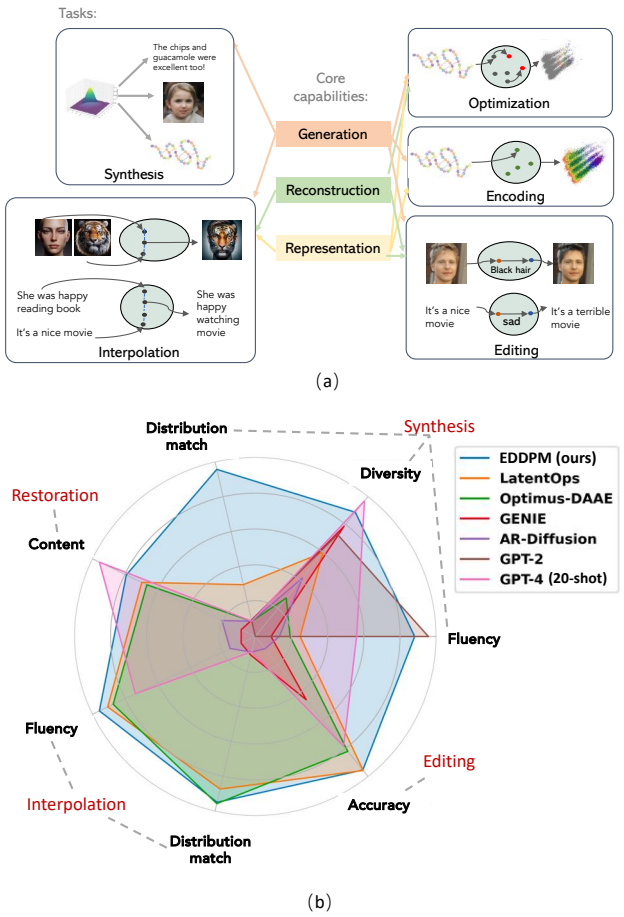

Figure 1. **(a)** Generation, reconstruction, and representation are the core capabilities for diverse applications. **(b)** EDDPM shows comprehensive abilities on different text tasks in the customer-review domain (§4.1).

1. Introduction

Numerous real-world applications involve synthesizing, modifying, restoring, and encoding data of different types, such as text, images, and biological molecules. Deep generative models are pivotal in these applications, owing to their three fundamental capabilities: (1) *Generation* of new samples from the data distribution; (2) *Reconstruction* of given instances with high fidelity; and (3) Extracting compact *representations* of raw data. Different applications can require different combinations of the capabilities (Figure 1a): for example, protein editing requires to produce a new valid protein sequence (*generation*) that retains many properties of the original sequence (*reconstruction*), while text inter-

[*]Equal contribution [1]MBZUAI [2]UC San Diego [3]University of Tokyo [4]Stanford University [5]CUHK-Shenzhen [6]CMU. Correspondence to: Guangyi Liu <guangyiliu.xx@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

polation could benefit from a latent space for continuous transitioning (*representation*) before mapping back to the discrete text space (*generation*).

Existing deep generative models typically show strengths in some, but not all, of the three capabilities, resulting in limited applicability or suboptimal performance (Figure 2). For example, variational autoencoders (VAEs, Kingma & Welling, 2014) are known for their tradeoff between realistic generation and faithful reconstruction (Chen et al., 2017; Higgins et al., 2017a), especially on text sequences (Bowman et al., 2016; Yang et al., 2017; Li et al., 2020; Liu et al., 2022; 2023a); Generative adversarial networks (GANs, Goodfellow et al., 2014) inherently lack inference of latent representations. Despite the rich subsequent research of incorporating latent inference (Zhou et al., 2019; Xia et al., 2023a; Donahue et al., 2016; Dumoulin et al., 2017; Li et al., 2017) such as GAN inversion (Xia et al., 2023b; Karras et al., 2020b), they fall short of faithfully reconstructing the inputs (Preechakul et al., 2022); Autoregressive models (e.g., modern large language models) excel in generating text, but often with limited diversity and lacking compact semantic representations (Brown et al., 2020). Similarly, recent diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) deliver new state-of-the-arts on synthesizing photo-realistic images, yet lack compact data representation and often rely on separately trained components for remediation (Rombach et al., 2021; Preechakul et al., 2022).

Moreover, these models often struggle with different forms of data. For example, it has been notoriously difficult for diffusion models and GANs to deal with text and protein data due to their discrete nature (Li et al., 2022; Qin et al., 2022; Kusner & Hernández-Lobato, 2016; Yu et al., 2017; de Rosa & Papa, 2021), and VAEs suffer from "posterior collapse" on text sequences (Bowman et al., 2016; Yang et al., 2017).

In this paper, we investigate a new deep generative approach based on *generalized* diffusion, that inherently integrates the three core capabilities and offers added flexibility to model both discrete and continuous data. Specifically, starting with the popular variational inference perspective of diffusion models (as in DDPM, Ho et al., 2020), we show that the standard formulation can be generalized by plugging in arbitrary "noising" operation at early diffusion steps, as long as the inverse "denoising" operation can be modeled. We can thus go beyond the common Gaussian noise (or other predefined image degradation, Bansal et al., 2022) and introduce *parameterized encoder-decoder* in place of noising-denoising. Crucially, the encoder-decoder parameters can naturally be learned together with other diffusion parameters using the original DDPM framework which is well-established for stable and scalable training (Rombach et al., 2021).

More specifically, we introduce encoder-decoder at the first diffusion step. That is, the diffusion process first encodes the input data into a low-dimensional latent vector, followed by common Gaussian noising steps. The resulting approach, EDDPMs (Generalized Encoding-Decoding Dffusion Probabilistic Models), combine a number of desirable properties and overcome difficulties in aforementioned deep generative models:

(1) Like VAEs and latent diffusion models (which are based on VAEs), EDDPMs offer compact semantic *representation* of data. Moreover, thanks to the unified learning of the encoder-decoder within the standard diffusion framework, EDDPMs obtain a much *improved* representation space than VAEs and latent diffusion, as shown in §4.

(2) The improved representation space also allows EDDPMs to avoid the trade-off between *generation* and *reconstruction* capabilities observed in VAEs and augmented GANs (Creswell & Bharath, 2019). Therefore, EDDPMs seamlessly integrate the three core capabilities, leading to more diverse applications and enhanced performance.

(3) The flexibility of the generalized diffusion formulation allows us to specify any desired encoder-decoder for modeling both *discrete* and *continuous* data. EDDPMs thus overcome the difficulty of standard diffusion and GANs on text and other discrete modalities.

(4) Moreover, as described in §4, we could further plug in large *pretrained* (autoregressive) language models (LMs) for initializing the encoder-decoder. This leads to greatly improved performance than previous text diffusion models (Li et al., 2022; Lin et al., 2023; Wu et al., 2023) that are inherently incompatible with off-the-shelf pretrained LMs.

We conduct extensive experiments on text, images, and protein sequences. EDDPMs demonstrate comprehensive capabilities across a wide range of tasks, such as data synthesis, reconstruction, interpolation, editing, and optimization (e.g., Figure 1b) on the different data modalities. The unified capabilities and enhanced performance of EDDPMs compared to existing generative models demonstrate its significant potential as a foundational technique for developing new, broadly applicable foundation models.

2. Background

We first give brief background of the popular diffusion model formulation from the variational inference perspective (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a). The training objective and model configurations have been well-established and are effective for stable and scalable training in practice (Rombach et al., 2021), show-

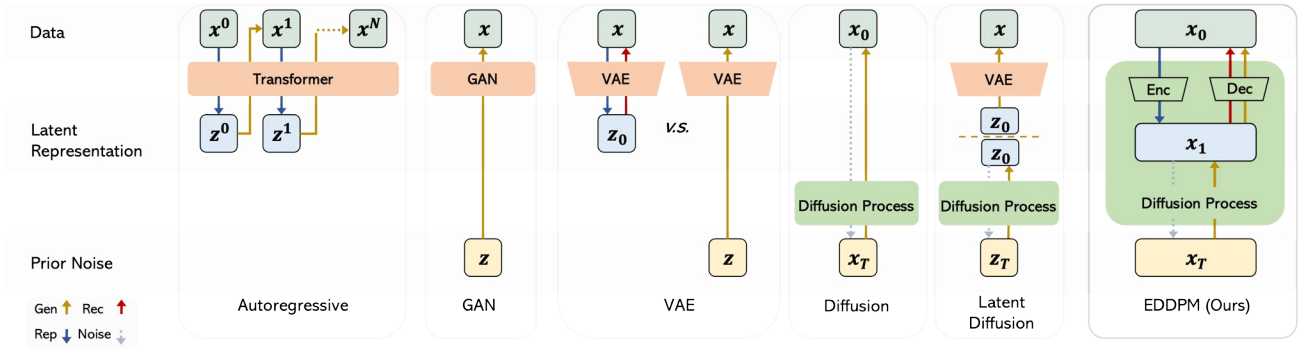

Figure 2. Different families of deep generative models.

ing advantages to alternative diffusion formalisms and other generative models. Our approach (§3) fits seamlessly in the (generalized) formulation and inherits these advantages.

A standard diffusion model consists of *noising* and *denoising* processes. Starting with a raw data point $\mathbf{x}_0$, the noising process sequentially adds Gaussian noise to the data at each diffusion step $t \in \{1, \dots, T\}$, following:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{1}$$

where β_t is the predefined variance schedule, and $\mathbf{x}_T$ becomes a standard Gaussian noise vector as $T \to \infty$. The denoising process learns to invert the above by learning a denoising operation $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ at each step t. When β_t is sufficiently small, we assume a Gaussian form:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \tag{2}$$

The parameters θ are learned with a variational lower bound on the marginal likelihood: with a Gaussian prior $p(\mathbf{x}_T)$,

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_q[-\log p_\theta(\mathbf{x}_0)] \\ &\le \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right].\end{aligned} \tag{3}$$

Recent diffusion formulation, in particular the denoising diffusion probabilistic models (DDPMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020), introduces a series of derivations to simplify the above objective, by splitting the sum and transforming and cancelling out relevant terms. The lower bound is thus rewritten as:

$$\begin{aligned}&\mathbb{E}_q\Bigg[-\log p(\mathbf{x}_T) - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \\ &\quad - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}\Bigg] \\ &= \mathbb{E}_q[\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right)}_{\mathcal{L}_T} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_0}] \\ &\quad + \sum_{t=2}^{T} \underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{\mathcal{L}_{t-1}}.\end{aligned} \tag{4}$$

Ho et al. (2020) further proposed to reparameterize $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ in Eq. (2) using $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))$, where α_t and $\bar{\alpha}_t$ are weights defined by β_t. The model is trained to directly predict the noise term $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ instead of the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$, which has demonstrated effectiveness in practice.

Despite their strengths, diffusion models have several limitations. The noisy nature of the latent variables $\mathbf{x}_t$, which represent corrupted versions of the data ($\mathbf{x}_0$), hinders their ability to capture abstract semantic meaning, making them less suitable for applications requiring such representations (Preechakul et al., 2022). Additionally, the reliance on predefined additive Gaussian noise restricts them to continuous, fixed-length data, posing challenges for handling discrete sequences of varying lengths. Recent efforts (Li et al., 2022; Lin et al., 2023; Wu et al., 2023) adapt the approach to text modeling, but their performance still greatly lags behind pretrained autoregressive models.

3. Generalized Encoding-Decoding Diffusion Probabilistic Models (EDDPMs)

We describe EDDPMs, which integrate the core capabilities of generation, reconstruction, and compact representation within one framework. EDDPMs thus combine the various advantages and applicabilities of different existing deep generative models while overcoming their limitations.

EDDPMs generalize the standard diffusion formulation described in §2. The intuition is that the leading steps ($t = 0, 1, \dots$) in the diffusion process can go beyond using only the common Gaussian noising-denoising. Instead, we can plug in more sophisticated "noising" and "denoising" operations. In particular, we use neural encoder and decoder with *learnable* parameters in place of "noising" and "denoising", respectively. The encoder allows the model to map its inputs into a low-dimensional space for a compact representation, and the decoder is to invert the encoding step and recover the inputs. Crucially, this change is fully compatible with the diffusion objective in Eqs. (3) and (4),

allowing us to retain most of the critical derivations in the well-established formulation (Sohl-Dickstein et al., 2015; Ho et al., 2020) and thus inherit the stable effective training of both the original denoising parameters and the new encoder-decoder parameters. As we discussed in §3.3, this differs crucially from latent diffusion models that train a VAE and a diffusion separately and are limited by the capability and quality of the VAE component. Our approach also shows superiority over previous joint training (Preechakul et al., 2022; Vahdat et al., 2021) which requires various approximations and can lead to unstable training and inferior performance.

3.1. The Generalized Formulation

We introduce learnable encoder-decoder to the leading n steps in the diffusion process (i.e., for all steps t, where $1 \leq t \leq n$). For simplicity and clarity, this work considers only the first step (i.e., $t = n = 1$) in the diffusion process, though all the derivation can be applied to larger n for n-layer hierarchical representations.

Let $\mathcal{E}_\lambda(\cdot)$ denote the neural encoder with free parameters λ. In the first diffusion step, the encoder transforms input $\mathbf{x}_0$ into a lower-dimensional latent vector $\mathbf{x}_1$, following:

$$q_\lambda(\mathbf{x}_1|\mathbf{x}_0) := \mathcal{N}(\mathbf{x}_1; \mathcal{E}_\lambda(\mathbf{x}_0), \beta_0\mathbf{I}) \tag{5}$$

The diffusion process then continues by adding standard Gaussian noises to $\mathbf{x}_1$ step by step until reaching $\mathbf{x}_T$ as in §2. Let $\mathcal{D}_\phi(\cdot)$ be the decoder with free parameters ϕ. At the end of the denoising process, the decoder transforms $\mathbf{x}_1$ back to $\mathbf{x}_0$ in the data space. The actual form of $\mathcal{D}_\phi(\cdot)$ and its respective conditional $p_\phi(\mathbf{x}_0|\mathbf{x}_1)$ can vary depending on the types of the data $\mathbf{x}_0$ (e.g., text, protein, image), as we detailed later. This offers added flexibility compared to the standard diffusion that is restricted to continuous data of fixed dimensionality.

As mentioned earlier, we can generalize the standard diffusion objective (Eqs. 3 and 4) to plug in the encoder and decoder seamlessly. The full derivations are provided in the appendix (§A). Briefly, we can adapt the left-hand side of Eq. (4) as:

$$\begin{aligned}&\mathcal{L}(\lambda, \phi, \theta) \leq \\ &\mathbb{E}_q\Bigg[-\log p(\mathbf{x}_T) - \sum_{t=3}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_1)} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_1)}{q(\mathbf{x}_t|\mathbf{x}_1)} \\ &\qquad - \log \frac{p_\theta(\mathbf{x}_1|\mathbf{x}_2)}{q(\mathbf{x}_2|\mathbf{x}_1)} - \log \frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}\Bigg].\end{aligned} \tag{6}$$

That is, compared to the original Eq. (4), we replace the noising-denoising $\frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}$ in step $t = 1$ with the parameterized encoder-decoder $\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}$; we also split from the sum an additional step $t = 2$, which serves to form KL divergences after rearrangement (see §A.2 for more details), resulting in the final objective:

$$\begin{aligned}\mathbb{E}_q\Bigg[&\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_1)||p(\mathbf{x}_T)\right)}_{\mathcal{L}_T} \\ &+ \sum_{t=3}^{T} \underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_1)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{\mathcal{L}_{t-1}} \\ &+ \underbrace{\mathrm{KL}\left(q_\lambda(\mathbf{x}_1|\mathbf{x}_0)||p_\theta(\mathbf{x}_1|\mathbf{x}_2)\right)}_{\mathcal{L}_{\text{align}}} \underbrace{- \log p_\phi(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_{\text{rec}}}\Bigg],\end{aligned} \tag{7}$$

where $\mathcal{L}_T$ and $\mathcal{L}_{t-1}$ match the respective terms in Eq. (4), and $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{rec}}$ are new due to the generalization.

The $\mathcal{L}_{\text{align}}$ term This term serves to align the compact representations $\mathbf{x}_1$ from the encoder $q_\lambda(\mathbf{x}_1|\mathbf{x}_0)$ and the denoising process $p_\theta(\mathbf{x}_1|\mathbf{x}_2)$, resulting in a consistent latent representation space in the model. The KL divergence between the two Gaussians is written as:

$$\mathcal{L}_{\text{align}} = \mathbb{E}_q\left[\gamma_1 \cdot \rho \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2\right] + const, \tag{8}$$

where $\boldsymbol{\mu}_\theta$ is the Gaussian mean in Eq. (2); $\rho := \frac{\alpha_1(1-\bar{\alpha}_1)}{\beta_1^2}$ and, following the approximation in Ho et al. (2020), γ_1 is set to 1 in practice.

The $\mathcal{L}_{\text{rec}}$ term This terms is the data reconstruction loss that corresponds to $\mathcal{L}_0$ in standard diffusion objective Eq. (4). However, thanks to the incorporation of the decoder $p_\phi(\mathbf{x}_0|\mathbf{x}_1)$, we can generalize the original $\mathcal{L}_0$ (which is only for continuous data like image) to model various data modalities, such as text and protein sequences of varying lengths. For instance, for text data, the decoder can be a (pretrained) language model and the reconstruction loss becomes a standard sequential cross-entropy loss.

We provide more detailed derivations of the full final objective in §A.4 and the complete training process in §B. Thanks to the compatibility with the standard diffusion formulation (Ho et al., 2020)), we can largely follow their training recipes for effective training. In practice, we introduce a hyperparameter weight w to balance the term $\mathcal{L}_{\text{rec}}$ against other terms like $\mathcal{L}_{t-1}$ and $\mathcal{L}_{\text{align}}$, as detailed in §A.4. In our experiments, using validation data, we set $w = 8$ for text and $w = 1$ for both protein and images.

3.2. Generation, Reconstruction, and Representation

The trained EDDPMs can naturally support the three core functionalities including generation, reconstruction, and representation, as well as the diverse applications built on top of the functionalities. To *generate* new samples, similar to standard diffusion, EDDPMs simulate a random noise $\mathbf{x}_T$ from

prior $p(\mathbf{x}_T)$, and go through the denoising process followed by decoding into a sample $\hat{\mathbf{x}}_0$ with the learned decoder. Due to the stochastic nature of the process (e.g., drawing random noise $\mathbf{x}_T$), EDDPMs can generate highly diverse samples (§4.1). *Representing* an input with a compact vector is done straightforwardly by applying the learned encoder. To *reconstruct* the input, we further apply the decoder on the latent vector to obtain the reconstructed sample. The compact representation space also facilitates other tasks (Figure 1), such as *interpolation* which is done by drawing an intermediate point between the representations of two given samples and decoding it into the data space, and *editing* which modifies the sample representation (e.g., with a latent classifier or latent vector arithmetic) followed by decoding. We disucss more details and analysis in experiments (§4).

3.3. Connections with Other Generative Models

We discuss the rich connections between EDDPMs to other diverse deep generative models (Goodfellow et al., 2016; Hu et al., 2018; Hu & Xing, 2022), providing insights into the integrated advantages of the new approach. EDDPMs can be viewed as **VAEs** with a learned diffusion model prior. This can be seen through Eq. (7) where $\mathcal{L}_{\text{rec}}$ corresponds to the reconstruction loss in VAEs and $\mathcal{L}_{\text{align}}$ corresponds to the KL regularization with prior. In this perspective, the vanilla VAEs use a standard Gaussian prior $p(\mathbf{x}_1)$ while EDDPMs learn the "prior" $p_\theta(\mathbf{x}_1|\mathbf{x}_2)$ through diffusion modeling. Note that previous studies have also explored learnable priors in VAEs to overcome the difficulty of vanilla VAEs training and to improve expressiveness (Dilokthanakul et al., 2016; Chen et al., 2017; Tomczak & Welling, 2018; Razavi et al., 2019; Wehenkel & Louppe, 2021; Vahdat et al., 2021; Goyal et al., 2017). Wehenkel & Louppe (2021) presented a preliminary study of diffusion priors in VAEs but without full derivations and experiments. If we replace all the noising-denoising steps in diffusion with parameterized encoders-decoders, we arrive at a **hierarchical VAEs** (e.g., NVAE, HiVAE, HVAE, Vahdat & Kautz, 2020b; Liu et al., 2023b; Bai et al., 2023). Compared to hierarchical VAEs, EDDPMs and diffusion allow for more effective training for generation thanks to the special parameterizations and training recipes (Ho et al., 2020). **Latent diffusion** (Esser et al., 2021) combines VAEs with diffusion but trains them separately. EDDPMs, inherently with unified training, offer a better semantic representation space than latent diffusion whose representation space is that of the vanilla VAEs. **Diffusion Autoencoders** (DiffAE, Preechakul et al., 2022) and **LSGM** (Vahdat et al., 2021) explore joining training strategies of diffusion and autoencoder and introduce different approximation for tractability. EDDPMs, benefiting from the well-established training objective and recipe from variational diffusion (Ho et al., 2020), achieve more effective training and better performance (§4.2). Also, the applications of DiffAE and LSGM on text/protein data have not been explored. Compared to recent **text diffusion models** (Yuan et al., 2023; He et al., 2022; Lin et al., 2023; Ye et al., 2023) that generate text non-autoregressively, EDDPMs can flexibly accommodate pretrained **autogressive language models** as the decoder for much enhanced performance.

4. Experiments

In this section, we present the main experimental results on text, image, and protein data. Additional results and related analysis can be found in §C.

4.1. Text

Setup We use BERT-small (Devlin et al., 2019; Bhargava et al., 2021) as the encoder and GPT2-xl (Radford et al., 2019) as the decoder. A warmup training is done to align the encoder and decoder using the bookcorpus dataset (Zhu et al., 2015) without diffusion, followed by the proposed unified EDDPM training on the Yelp review dataset (Shen et al., 2017; Li et al., 2018). We compare EDDPM with LatentOps (Liu et al., 2023a) and Optimus-DAAE (Shen et al., 2020; Li et al., 2020), using the same architecture and training procedure. LatentOps is a form of latent diffusion model with successful applications on text data. It uses ordinary differential equations (ODEs), instead of discrete diffusion process as in standard latent diffusion models, on top of a separately trained VAE latent space. Optimus-DAAE is an improved variant of large-scale text VAEs (Li et al., 2020) inspired by (Shen et al., 2020). We also compare with latest text diffusion models, GENIE (Lin et al., 2023) and AR-Diffusion (Wu et al., 2023). In addition, we compare with fine-tuned GPT2-xl and 20-shot GPT4 (Achiam et al., 2023). For comprehensive evaluation, we study tasks of generation, reconstruction, interpolation, and editing. More details are in §C.1.1. We use BLEU score to measure **content** preservation, MAUVE (Pillutla et al., 2021) to assess the **distribution match** between the generated samples and the original data, and perplexity to evaluate text **fluency**. For text editing with latent arithmetic, transfer **accuracy** is measured as the geometric mean (Liu et al., 2023a) of the accuracy (by BERT classifiers) and the BLEU score (relative to the input text).

4.1.1. Text Generation and Reconstruction

Figure 1(b) and Table 1 show the results. Autoencoding-based models (LatentOps and Optimus-DAAE) exhibit proficiency in reconstruction tasks, primarily because their objectives rely heavily on reconstruction quality. However, these models struggle to generate fluent text that aligns with the real data distribution. This limitation stems from the regularization in latent space designed to adhere to a specific prior distribution, such as a standard Gaussian. The regular-

	Reconstruction	Generation			Latent Arithmetic	Interpolation	
	Content↑	Fluency↓	Distr. Match↑	Diversity↑	Accuracy↑	Fluency↓	Distr. Match↑
LatentOps	87.6	68.1	<u>0.240</u>	0.50	**57.3**	<u>32.5</u>	0.697
Optimus-DAAE	86.1	94.1	0.006	0.17	51.0	33.7	**0.770**
GENIE	58.5	337.6	0.013	0.69	33.9	258.2	0.029
AR-Diffusion	64.1	157.8	0.007	0.32	17.0	163.9	0.012
GPT2	-	**15.0**	0.015	0.65	-	-	-
GPT4	**100**	25.7	0.007	**0.87**	49.3	39.7	0.010
EDDPM	<u>92.1</u>	<u>16.4</u>	**0.977**	<u>0.79</u>	<u>57.1</u>	**30.8**	<u>0.763</u>

Table 1. Evaluation of Text Reconstruction, Generation (§4.1.1), Latent Vector Arithmetic (§4.1.3), and Interpolation (§4.1.2). The best results are highlighted in **bold**, and the second-best results are <u>underlined</u>.

ization, while mathematically sound, often results in a significant disparity between the model's latent space and the actual latent distribution observed in real-world scenarios (Yang et al., 2017; Li et al., 2020). The text diffusion models GENIE and AR-Diffusion apply a diffusion process over the token space. Despite the reconstruction-based training objective, these models fall short of both reconstruction and generation. The results highlight the difficulty of standard diffusion modeling for discrete text data. The fine-tuned GPT2 autoregressive model excels in generation fluency but performs poorly in terms of generation diversity and domain match. It also fails for reconstruction. Despite the general capability of GPT4, it fails to capture the domain-specific text characteristics via in-context demonstrations. In comparison, EDDPM achieves strong performance across all metrics for both reconstruction and generation. The learned autoencoding ensures effective reconstruction, while the diffusion process introduces dynamic regularization to the latent space, ensuring superior quality in text generation.

4.1.2. Sentence Interpolation

In this experiment, we randomly selected 200 samples from the test set, dividing them into two groups of 100 each. We performed interpolation between these groups, encoding the first 100 samples as $\mathbf{x}_1^1$ and the second 100 as $\mathbf{x}_1^2$. Given that both sets of latent distributions adhere to a Gaussian distribution, we utilized spherical linear interpolation (Slerp) as per Shoemake's method (Shoemake, 1985), employing the formula $\text{Slerp}(\mathbf{x}_1^1, \mathbf{x}_1^2; \alpha)$ where $0 \leq \alpha \leq 1$. Our aim was to generate a series of sentences that smoothly transition in semantics, achieving a fluent blend.

We set the number of interpolation steps to 10 and evaluated the quality at each step to gauge performance. Particularly, the interpolation results at the midpoint are shown in Table 1. EDDPM demonstrates a consistent ability to produce fluent sentences that closely align with the original data distribution. In contrast, baseline models such as GENIE and AR-Diffusion struggled, primarily due to their lack of a semantically coherent latent space for whole sentences. Additionally, we assessed the outcomes using MAUVE and BLEU metrics, with detailed interpolation examples provided in §C.1.2.

4.1.3. Latent Vector Arithmetic

Text editing (e.g., changing the text sentiment) necessitates proficiency in all three fundamental abilities. Previous research (Shen et al., 2020; Hu et al., 2017) demonstrated that sentence representations can capture linguistic relationships through simple arithmetic operations. We use this approach as a proxy for evaluating the quality of the learned latent space. Specifically, we use the sentiment attribute and evaluate how well the learned latent space can support the inference of text sentiment change with simple arithmetic operations in the latent space. We first obtain a positive-sentiment latent vector by averaging the latent vectors of 100 positive sentences randomly sampled from the dataset. A negative-sentiment latent vector is obtained similarly with 100 negative sentences. We then compute the difference between the two sentiment vectors, denoted as $\mathbf{v}$. Given a new sentence, it is first encoded to obtain its latent vector, $\mathbf{x}_1 = \mathcal{E}_\lambda(\mathbf{x}_0)$. The sentiment-transferred sentence is then acquired via $\mathcal{D}_\phi(\mathbf{x}_1 \pm k\mathbf{v})$, where k serves as a weight modulating the degree of transfer. We tried different $k \in \text{range}(1, 5, .5)$ and show the best one in Table 1. EDDPM is on par with LatentOps and outperforms others, validating a well-structured latent space.

4.1.4. Training Efficiency

To show the training efficiency of EDDPMs, we measured the training cost on text data. The results shown in Table 2 validate that training EDDPMs is as efficient as training VAEs and variants (e.g., Optimus-DAAE, LatentOps) with the same encoder-decoder architecture. Due to the added diffusion process, the training time of EDDPM for each epoch is 1.3x that of LatentOps and Optimus-DAAE. However, thanks to the stable training process inherited from the well-established DDPM formulation, EDDPMs avoid the different training tricks necessary in VAEs and variants, such as beta-annealing (Bowman et al., 2016), free bits (Kingma et al., 2017), and cyclic annealing schedule (Fu et al., 2019).

Models	Time / Epoch	Overall Training Time
LatentOps	13.98 mins	6.98 hrs
Optimus-DAAE	14.49 mins	7.02 hrs
EDDPM	19.35 mins	7.41 hrs

Table 2. Training cost.

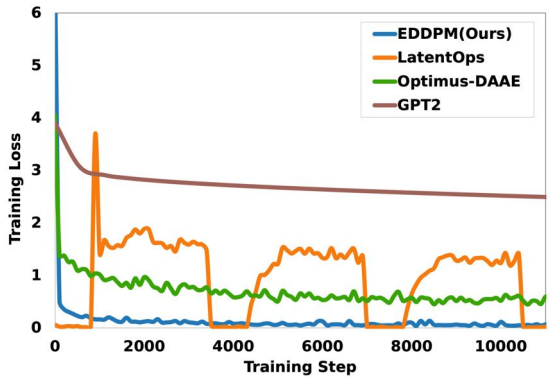

Figure 3. Training curves. EDDPM converges fast in a stable way. LatentOps, which depends on text VAE, relies on periodic scheduling of the weight of the prior regularization. The scheduling is common for training text VAE models (Li et al., 2020), and leads to the ups-and-downs in the loss curve, indicating a difficult trade-off between generation and reconstruction capabilities.

This allows EDDPMs to converge in fewer steps, leading to similar overall training cost. Figure 3 shows the training curves.

4.2. Image

Setup We adopt UNet (Long et al., 2015) as the encoder and the diffusion-based model in DiffAE (Preechakul et al., 2022)) as the decoder. Following DiffAE, we train our model and then evaluate the reconstruction and generation abilities on the popular FFHQ (Karras et al., 2019), CelebA (Karras et al., 2018), LSUN-Bedroom, and LSUN-Horse (Yu et al., 2015) datasets. We use the CelebA-HQ dataset to perform image manipulation tasks. In addition to DiffAE, we compare our model with a number of latest models, including the Latent Diffusion Model (LDM, Rombach et al., 2021), DDIM (Song et al., 2021a), StyleGAN-XL (Sauer et al., 2022), NVAE (Vahdat & Kautz, 2020b), and Consistency Models (Song et al., 2023). We compare EDDPM with these models where existing corresponding model checkpoints exist. Following the common practice, we use FID and reconstruction-FID (rFID) to evaluate the generation quality and reconstruction quality, respectively. To systematically evaluate EDDPM, besides the individual evaluations of reconstruction and generation, we also perform image *interpolation* and *manipulation* (§4.2.2) tasks, which can reflect the integrated ability. We perform interpolation within the latent space and then reconstruct images

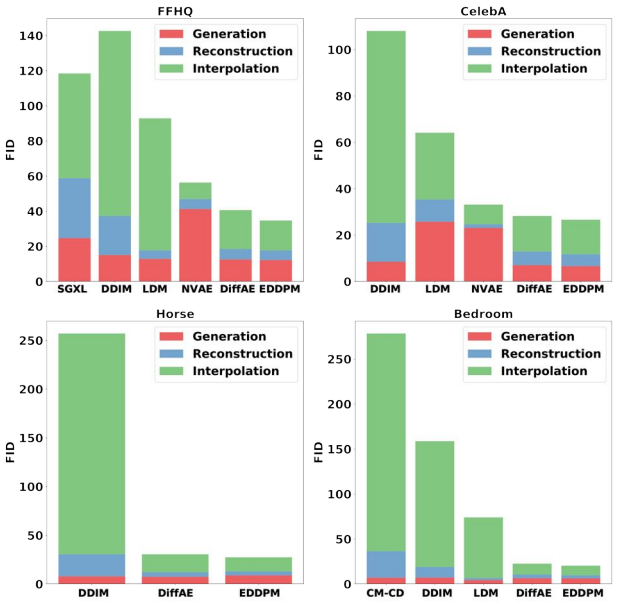

Figure 4. Overall performance comparison on images. SGXL stands for StyleGAN-XL. We stack the FIDs of generation, reconstruction, and interpolation together to show the overall performances of different models.

from the resulting interpolated representations. Given two latent vectors, $\mathbf{x}_1^1$ and $\mathbf{x}_1^2$, we employ linear interpolation using the formula: $\alpha\mathbf{x}_1^1 + (1-\alpha)\mathbf{x}_1^2$, where α represents the interpolation ratio and we use $\alpha = 0.2$ and 0.4. This interpolated latent vector is subsequently fed into the decoder to produce the interpolated image. In total, 50k images are used for interpolation and the obtained images are measured against 50k images in the training set to obtain FID score. The detailed image experimental setup is in §C.2.1.

4.2.1. Generation, Reconstruct., Interpolation

The overall performances for generation, reconstruction and interpolation are shown in Figure 4. For all experiments requiring diffusion process, we fix the inference steps to be $T = 50$. From the figure, We could observe: (1) EDDPM consistently achieves superior or at least comparable performance across all three evaluated tasks, and is the best in terms of aggregate performance across the datasets examined; (2) NVAE demonstrates robust capabilities in the tasks of reconstruction and interpolation. However, its performance in the generative task is notably less competitive compared with other models; (3) LDM, while demonstrating commendable performance in reconstruction, falls short in maintaining quality during interpolation. The detailed experimental results are shown in Table 7 (§C.2.2).

4.2.2. Image Manipulation

We deploy our model trained on FFHQ to the CelebA-HQ domain in a zero-shot fashion. We select the CelebA-HQ

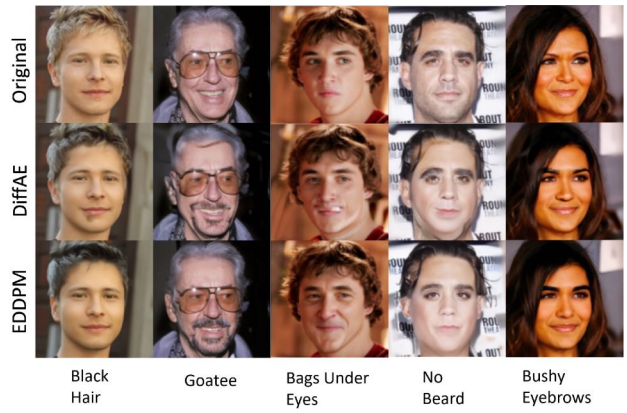

Figure 5. **Image Manipulation:** The procedure for manipulation is detailed in Section 4.2.2. The class names provided at the bottom indicate the target class towards which the images are being manipulated.

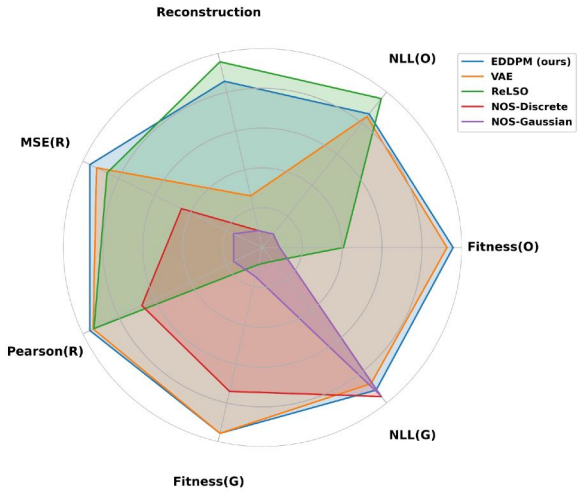

Figure 6. EDDPM shows robust performance on fundamental tasks about protein sequences. R, G, and O denote **R**epresentation, **G**eneration, and **O**ptimization respectively. Results are normalized for better visualization. Detailed results in §C.3

dataset for this task due to the availability of 40 binary classification category labels.

Following DiffAE, we train a linear classifier, $\mathbf{y} = \mathbf{w}^\top \mathbf{x}_1 + b$, on 70% of the training data for each attribute. This classifier predicts the attribute based on the representation $\mathbf{x}_1$. To construct the manipulated image representation, we use $\mathbf{x}_1' = \mathbf{x}_1 + \epsilon \mathbf{w}$, where ϵ is a scalar determining the manipulation magnitude. The manipulated representation $\mathbf{x}_1'$ is then used to generate the corresponding image. We show the examples in Figure 5. We compare ours with DiffAE as we follow its setting and it is also one of the state-of-the-art attribute-based image manipulation methods. Our evaluation focuses on two aspects: 1) *image quality post-manipulation*: as shown in Table 3, EDDPM consistently yields high-quality images after manipulation. 2) *alignment with target class*: we test the linear classifier on the remaining 30% of the dataset. In terms of weighted AUC, EDDPM's representation achieves 0.915, closely matching DiffAE's 0.917. Notably, EDDPM surpasses in accuracy, registering 0.893 against DiffAE's 0.795. A detailed AUC comparison can be found in §C.2.

Model	$\epsilon = 0.1$	$\epsilon = 0.3$
DiffAE	19.62	23.69
EDDPM	**13.48**	**18.43**

Table 3. Manipulation (FID).

4.3. Protein Sequences

Setup Following the setup of ReLSO (Castro et al., 2022), a protein autoencoder, we adopt a simple transformer as the encoder and convolutional layers as the decoder. In line with the settings of ReLSO, we jointly train a simple regressor to predict the fitness values from the latent embeddings of protein sequences. This fitness value, representing some desired properties of proteins, serves as a performance metric with higher values indicating superiority. Our models are trained and evaluated on the Gifford (Liu et al., 2019) dataset and GFP (Sarkisyan et al., 2016) dataset separately. In addition to ReLSO, we provided quantitative comparisons between our models and several baseline models, namely NOS (Gruver et al., 2023) which utilized transformer-based discrete diffusion and Gaussian diffusion model for protein design, as well as vanilla VAEs (Kingma & Welling, 2014). Detailed experimental setups are in §C.3.1. We evaluate EDDPM's representation ability in §4.3.1 and its generation ability through both protein optimization (§4.3.2) and protein generation (§C.3.4). The evaluation on EDDPM's reconstruction ability is provided in the appendix at §C.3.3.

4.3.1. Protein Representation

After training, each protein sequence in the test set is transformed into a latent representation, upon which the fitness value is predicted using the regressor in the latent space. Evaluation metrics, including MSE, L1 Norm, Pearson and Spearman correlation coefficients, are computed between the predicted values and the ground truth. The results are presented in Table 9. The regressor trained in the latent space of EDDPM demonstrates superior performance over the regressors from the baseline models on all four metrics on the Gifford Dataset. This suggests that EDDPM obtained more refined representations for the protein sequences, leading to more accurate predictions by the regressor. This is also evident from the visualization of the latent space. In Figure 7, we display the latent spaces for both ReLSO (left) and EDDPM (right), coloring proteins by their respective fitness value intervals. In the latent space of ReLSO in Figure 7(left), proteins with fitness values less than 0 exhibits an overlap (inside the red box). While some overlap persists across intervals inside the red box in EDDPM's latent space

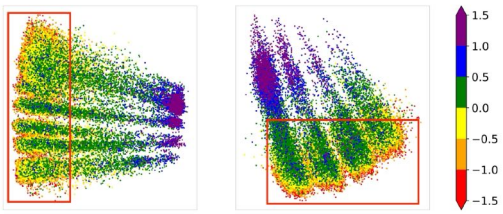

Figure 7. Protein Latent Space of ReLSO (left) and EDDPM (right). Different colors represent different fitness values of the corresponding protein sequence.

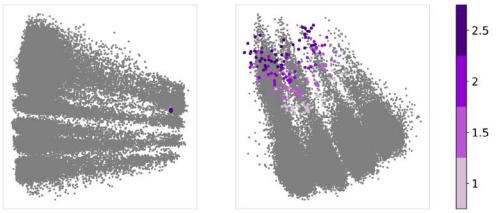

Figure 8. Optimized Protein Sequence of ReLSO (left) and EDDPM (right) in the latent space.

in Figure 7 (right), the delineation between each interval is much clearer.

4.3.2. Protein Optimization

We optimize a protein sequence by optimizing its corresponding representation in the latent space. We adopt the sampling algorithm introduced in LatentOps (Liu et al., 2023a) that solves an ODE involving the regressor (Liu et al., 2023a). This approach requires a target fitness value to guide the optimization. We set this value to 1, 1.5, 2, and 2.5 for Gifford proteins; 3, 4 for GFP proteins, all of which represent reasonably high fitness values within the respective dataset. Visualizations of the optimized sequence with ReLSO and EDDPM algorithms are shown in Figure 8. The grey dots represent proteins from the Gifford dataset, while colored dots represents optimized proteins with varying target fitness values. As shown in Figure 8 (left), the pseudo-convex nature of ReLSO leads to convergence of optimized sequences to a singular point, revealing a lack of diversity. In contrast, as depicted in Figure 8 (right), our model not only achieves superior fitness values but also fosters a broader protein variety. For a detailed quantitative analysis, refer to §C.3.2.

5. Other Related Works

To tackle the limitations of current generative models, past research has ventured into developing hybrid models by combining VAEs (Razavi et al., 2019; Vahdat & Kautz, 2020a; Dai & Wipf, 2019) and GANs (Karras et al., 2020a; Esser et al., 2021; Xia et al., 2023a) to create VAE-GAN hybrids (Larsen et al., 2016; Hu et al., 2018; Wu et al., 2020; Xu et al., 2021). Recently, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have been noted for their generative ability, yet they fall short in providing a robust semantic latent space. Efforts like Wehenkel & Louppe (2021) and Vahdat et al. (2021) explored the parameterization of the variational prior with diffusion models. Latent diffusion models (e.g., stable diffusion) (Rombach et al., 2021) merge autoencoder and diffusion model architectures, optimizing diffusion training with the VAE's pivotal role in image dimension compression. A deep dive can be found in §D.1. Despite these advancements, it remains elusive to achieve all three capabilities of representation, generation, and reconstruction effectively. EDDPMs aim to tackle these challenges.

6. Conclusion

This work generalized the diffusion model to introduce EDDPMs, an innovative generative framework designed to seamlessly integrate the three core functionalities for generative models: generation, reconstruction and representation. To this end, we incorporated parameterized encoder and decoder transformations into the conventional diffusion process. We derived an end-to-end training objective from the data likelihood inspired from the widely-used DDPM training framework. Experimental results across image, text, and protein sequence data demonstrate that EDDPMs consistently outperform strong baselines across all three core functionalities. Building a compact high-quality representation space with the versatile utilities (generation, reconstruction, encoding, interpolation, editing) can be significant for building new foundation models, such as world models (Hu & Shu, 2023; LeCun, 2022; Assran et al., 2023), for more robust machine reasoning (Hao et al., 2023; Wong et al., 2023).

Limitations In this work, EDDPMs have not been verified on other data modalities such as videos, audios, and time series. It would be interesting to see how EDDPMs will perform on all the diverse forms of data, and even multimodal data (such as text-image). In addition, we have only parameterized the first diffusion step with learnable encoding/decoding. A single latent vector representation could be limited to capture relevant information of data for different tasks. We would like to investigate our approach with more encoding/decoding-based diffusion steps for hierarchical latent representations.

Impact Statement

There are potential societal consequences associated with diffusion modeling and deep generative models in general regarding content creation, privacy, and others.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report, 2023.

Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., and Ballas, N. Self-supervised learning from images with a joint-embedding predictive architecture. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15619–15629, 2023.

Bai, R., Huang, R., Qin, Y., Chen, Y., and Lin, C. Hvae: A deep generative model via hierarchical variational auto-encoder for multi-view document modeling. Information Sciences, 623:40–55, 2023.

Bansal, A., Borgnia, E., Chu, H., Li, J. S., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., and Goldstein, T. Cold diffusion: Inverting arbitrary image transforms without noise. CoRR, abs/2208.09392, 2022. doi: 10.48550/arXiv.2208.09392.

Bhargava, P., Drozd, A., and Rogers, A. Generalization in nli: Ways (not) to go beyond simple heuristics, 2021.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. Generating sentences from a continuous space. In Goldberg, Y. and Riezler, S. (eds.), Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016, pp. 10–21. ACL, 2016. doi: 10.18653/v1/k16-1002. URL https://doi.org/10.18653/v1/k16-1002.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. ArXiv, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

Castro, E., Godavarthi, A., Rubinfien, J., Givechian, K. B., Bhaskar, D., and Krishnaswamy, S. Relso: A transformer-based model for latent space optimization and generation of proteins, 2022.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. URL https://openreview.net/forum?id=BysvGP5ee.

Creswell, A. and Bharath, A. A. Inverting the generator of a generative adversarial network. IEEE Transactions on Neural Networks and Learning Systems, 30(7):1967–1974, 2019. doi: 10.1109/TNNLS.2018.2875194.

Dai, B. and Wipf, D. Diagnosing and enhancing VAE models. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=B1e0X3C9tQ.

de Rosa, G. H. and Papa, J. P. A survey on text generation using generative adversarial networks. Pattern Recognition, 119:108098, 2021.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep unsupervised clustering with gaussian mixture variational autoencoders. CoRR, abs/1611.02648, 2016. URL http://arxiv.org/abs/1611.02648.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. CoRR, abs/1605.09782, 2016. URL http://arxiv.org/abs/1605.09782.

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. C. Adversarially learned inference. In ICLR (Poster). OpenReview.net, 2017.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, pp. 12873–12883. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01268.

Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., and Carin, L. Cyclical annealing schedule: A simple approach to mitigating kl vanishing, 2019.

Goodfellow, I., Bengio, Y., and Courville, A. Deep Generative Models, chapter 20. MIT Press, 2016.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 2672–2680, 2014.

Goyal, P., Hu, Z., Liang, X., Wang, C., and Xing, E. P. Nonparametric variational auto-encoders for hierarchical representation learning. In Proceedings of the IEEE International Conference on Computer Vision, pp. 5094–5102, 2017.

Gruver, N., Stanton, S., Frey, N. C., Rudner, T. G. J., Hotzel, I., Lafrance-Vanasse, J., Rajpal, A., Cho, K., and Wilson, A. G. Protein design with guided discrete diffusion, 2023.

Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z. Reasoning with language model is planning with world model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 8154–8173, 2023.

He, Z., Sun, T., Wang, K., Huang, X., and Qiu, X. Diffusionbert: Improving generative masked language models with diffusion models, 2022.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 6626–6637, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In International Conference on Learning Representations, 2017a. URL https://openreview.net/forum?id=Sy2fzU9gl.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017b.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

Hu, Z. and Shu, T. Language models, agent models, and world models: The law for machine reasoning and planning. arXiv preprint arXiv:2312.05230, 2023.

Hu, Z. and Xing, E. P. Toward a 'Standard Model' of Machine Learning. Harvard Data Science Review, 4(4), oct 27 2022. https://hdsr.mitpress.mit.edu/pub/zkib7xth.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. Toward controlled generation of text. In International conference on machine learning, pp. 1587–1596. PMLR, 2017.

Hu, Z., Yang, Z., Salakhutdinov, R., and Xing, E. P. On unifying deep generative models. In International Conference on Learning Representations, 2018.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. In ICLR. OpenReview.net, 2018.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4401–4410, 2019.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pp. 8107–8116. Computer Vision Foundation / IEEE, 2020a. doi: 10.1109/CVPR42600.2020.00813.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In CVPR, pp. 8107–8116. Computer Vision Foundation / IEEE, 2020b.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving variational inference with inverse autoregressive flow, 2017.

Kusner, M. J. and Hernández-Lobato, J. M. Gans for sequences of discrete elements with the gumbel-softmax distribution, 2016.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. In Balcan, M. and Weinberger, K. Q. (eds.), Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, volume 48 of JMLR Workshop and Conference Proceedings, pp. 1558–1566. JMLR.org, 2016.

LeCun, Y. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. Open Review, 62(1), 2022.

Li, C., Liu, H., Chen, C., Pu, Y., Chen, L., Henao, R., and Carin, L. ALICE: towards understanding adversarial learning for joint distribution matching. In NIPS, pp. 5495–5503, 2017.

Li, C., Gao, X., Li, Y., Peng, B., Li, X., Zhang, Y., and Gao, J. Optimus: Organizing sentences via pre-trained modeling of a latent space. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, pp. 4678–4699. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.378. URL https://doi.org/10.18653/v1/2020.emnlp-main.378.

Li, J., Jia, R., He, H., and Liang, P. Delete, retrieve, generate: A simple approach to sentiment and style transfer. 2018.

Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., and Hashimoto, T. B. Diffusion-lm improves controllable text generation, 2022.

Lin, Z., Gong, Y., Shen, Y., Wu, T., Fan, Z., Lin, C., Duan, N., and Chen, W. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise, 2023.

Liu, G., Zeng, H., Mueller, J., Carter, B., Wang, Z., Schilz, J., Horny, G., Birnbaum, M. E., Ewert, S., and Gifford, D. K. Antibody complementarity determining region design using high-capacity machine learning. Bioinformatics, 36(7):2126–2133, 11 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz895.

Liu, G., Yang, Z., Tao, T., Liang, X., Bao, J., Li, Z., He, X., Cui, S., and Hu, Z. Don't take it literally: An edit-invariant sequence loss for text generation, 2022.

Liu, G., Feng, Z., Gao, Y., Yang, Z., Liang, X., Bao, J., He, X., Cui, S., Li, Z., and Hu, Z. Composable text controls in latent space with ODEs. In Bouamor, H., Pino, J., and Bali, K. (eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 16543–16570, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.1030.

Liu, S., Zhang, Y., Peng, J., and Shang, X. An improved hierarchical variational autoencoder for cell–cell communication estimation using single-cell rna-seq data. Briefings in Functional Genomics, pp. elac056, 2023b.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pp. 3431–3440. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298965. URL https://doi.org/10.1109/CVPR.2015.7298965.

Pillutla, K., Swayamdipta, S., Zellers, R., Thickstun, J., Welleck, S., Choi, Y., and Harchaoui, Z. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In NeurIPS, 2021.

Preechakul, K., Chatthee, N., Wizadwongsa, S., and Suwajanakorn, S. Diffusion autoencoders: Toward a meaningful and decodable representation. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 10609–10619. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01036.

Qin, L., Welleck, S., Khashabi, D., and Choi, Y. Cold decoding: Energy-based constrained text generation with langevin dynamics. Advances in Neural Information Processing Systems, 35:9538–9551, 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

Razavi, A., van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with VQ-VAE-2. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 14837–14847, 2019.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10674–10685, 2021. URL https://api.semanticscholar.org/CorpusID:245335280.

Sarkisyan, K. S., Bolotin, D. A., Meer, M. V., Usmanova, D. R., Mishin, A. S., Sharonov, G. V., Ivankov, D. N.,

Bozhanova, N. G., Baranov, M. S., Soylemez, O., Bogatyreva, N. S., Vlasov, P. K., Egorov, E. S., Logacheva, M. D., Kondrashov, A. S., Chudakov, D. M., Putintseva, E. V., Mamedov, I. Z., Tawfik, D. S., Lukyanov, K. A., and Kondrashov, F. A. Local fitness landscape of the green fluorescent protein. Nature, 533:397–401, 2016.

Sauer, A., Schwarz, K., and Geiger, A. Stylegan-xl: Scaling stylegan to large diverse datasets. In SIGGRAPH (Conference Paper Track), pp. 49:1–49:10. ACM, 2022.

Shen, T., Lei, T., Barzilay, R., and Jaakkola, T. Style transfer from non-parallel text by cross-alignment. 2017.

Shen, T., Mueller, J., Barzilay, R., and Jaakkola, T. S. Educating text autoencoders: Latent representation guidance via denoising. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pp. 8719–8729. PMLR, 2020.

Shoemake, K. Animating rotation with quaternion curves. In Cole, P., Heilman, R., and Barsky, B. A. (eds.), Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1985, San Francisco, California, USA, July 22-26, 1985, pp. 245–254. ACM, 1985. doi: 10.1145/325334.325242. URL https://doi.org/10.1145/325334.325242.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. R. and Blei, D. M. (eds.), Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of JMLR Workshop and Conference Proceedings, pp. 2256–2265. JMLR.org, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In ICLR. OpenReview.net, 2021a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations, 2021b.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. arXiv preprint arXiv:2303.01469, 2023.

Tomczak, J. M. and Welling, M. VAE with a vampprior. In Storkey, A. J. and Pérez-Cruz, F. (eds.), International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain, volume 84 of Proceedings of Machine Learning Research, pp. 1214–1223. PMLR, 2018. URL http://proceedings.mlr.press/v84/tomczak18a.html.

Vahdat, A. and Kautz, J. NVAE: A deep hierarchical variational autoencoder. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020a.

Vahdat, A. and Kautz, J. NVAE: A deep hierarchical variational autoencoder. In NeurIPS, 2020b.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pp. 11287–11302, 2021.

Wehenkel, A. and Louppe, G. Diffusion priors in variational autoencoders. CoRR, abs/2106.15671, 2021. URL https://arxiv.org/abs/2106.15671.

Wong, L., Grand, G., Lew, A. K., Goodman, N. D., Mansinghka, V. K., Andreas, J., and Tenenbaum, J. B. From word models to world models: Translating from natural language to the probabilistic language of thought. arXiv preprint arXiv:2306.12672, 2023.

Wu, T., Fan, Z., Liu, X., Gong, Y., Shen, Y., Jiao, J., Zheng, H.-T., Li, J., Wei, Z., Guo, J., Duan, N., and Chen, W. Ar-diffusion: Auto-regressive diffusion model for text generation, 2023.

Wu, Y., Zhou, P., Wilson, A. G., Xing, E., and Hu, Z. Improving GAN training with probability ratio clipping and sample reweighting. Advances in Neural Information Processing Systems, 33:5729–5740, 2020.

Xia, W., Zhang, Y., Yang, Y., Xue, J., Zhou, B., and Yang, M. GAN inversion: A survey. IEEE Trans. Pattern Anal. Mach. Intell., 45(3):3121–3138, 2023a. doi: 10.1109/TPAMI.2022.3181070. URL https://doi.org/10.1109/TPAMI.2022.3181070.

Xia, W., Zhang, Y., Yang, Y., Xue, J., Zhou, B., and Yang, M. GAN inversion: A survey. IEEE Trans. Pattern Anal. Mach. Intell., 45(3):3121–3138, 2023b.

Xu, L., Zheng, L., Li, W., Chen, Z., Song, W., Deng, Y., Chang, Y., Xiao, J., and Yuan, B. NVAE-GAN based approach for unsupervised time series anomaly detection. CoRR, abs/2101.02908, 2021.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved variational autoencoders for text modeling using dilated convolutions. In International conference on machine learning, pp. 3881–3890. PMLR, 2017.

Ye, J., Zheng, Z., Bao, Y., Qian, L., and Wang, M. Dinoiser: Diffused conditional sequence learning by manipulating noises, 2023.

Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. CoRR, abs/1506.03365, 2015.

Yu, L., Zhang, W., Wang, J., and Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient, 2017.

Yuan, H., Yuan, Z., Tan, C., Huang, F., and Huang, S. Seqdiffuseq: Text diffusion with encoder-decoder transformers, 2023.

Zhou, Y., Gu, K., and Huang, T. S. Unsupervised representation adversarial learning network: from reconstruction to generation. In International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019, pp. 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8852395. URL https://doi.org/10.1109/IJCNN.2019.8852395.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In The IEEE International Conference on Computer Vision (ICCV), December 2015.

A. Derivation

A.1. Derivation of our ELBO loss

Below is a derivation of the error bound of the log-likelihood loss.

$$\mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}\left[-\log p_{\phi,\theta}(\mathbf{x}_0)\right] \tag{9}$$

$$\leq \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}\left[-\log p_{\phi,\theta}(\mathbf{x}_0)+\mathrm{KL}(q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_{\phi,\theta}(\mathbf{x}_{1:T}|\mathbf{x}_0))\right] \tag{10}$$

$$= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}\left[-\log p_{\phi,\theta}(\mathbf{x}_0)+\mathbb{E}_{\mathbf{x}_{1:T}\sim q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}[\log\frac{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_{\phi,\theta}(\mathbf{x}_{1:T}|\mathbf{x}_0)}]\right] \tag{11}$$

$$= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}\left[-\log p_{\phi,\theta}(\mathbf{x}_0)+\mathbb{E}_{\mathbf{x}_{1:T}\sim q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}[\log\frac{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_{\phi,\theta}(\mathbf{x}_{0:T})/p_{\phi,\theta}(\mathbf{x}_0)}]\right] \tag{12}$$

$$= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}\left[-\log p_{\phi,\theta}(\mathbf{x}_0)+\mathbb{E}_{\mathbf{x}_{1:T}\sim q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}[\log\frac{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_{\phi,\theta}(\mathbf{x}_{0:T})}]+\log p_{\phi,\theta}(\mathbf{x}_0)\right] \tag{13}$$

$$= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\mathbf{x}_{1:T}\sim q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log\frac{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_{\phi,\theta}(\mathbf{x}_{0:T})}\right] \tag{14}$$

$$= \mathbb{E}_q\left[-\log\frac{p_{\phi,\theta}(\mathbf{x}_{0:T})}{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \tag{15}$$

A.2. Derivation of our KL loss

Below is a derivation of Eq. (7).

$$\mathcal{L}(\lambda,\phi,\theta) = \mathbb{E}_q\left[-\log\frac{p_{\phi,\theta}(\mathbf{x}_{0:T})}{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \tag{16}$$

$$= \mathbb{E}_q\left[-\log\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)p_\theta(\mathbf{x}_{1:T})}{q(\mathbf{x}_{2:T}|\mathbf{x}_1)q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}\right] \tag{17}$$

$$= \mathbb{E}_q\left[-\log\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)p(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}\right] \tag{18}$$

$$= \mathbb{E}_q\left[-\log p(\mathbf{x}_T)-\sum_{t=3}^T\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}-\log\frac{p_\theta(\mathbf{x}_1|\mathbf{x}_2)}{q(\mathbf{x}_2|\mathbf{x}_1)}-\log\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}\right] \tag{19}$$

$$= \mathbb{E}_q\left[-\log p(\mathbf{x}_T)-\sum_{t=3}^T\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_1)}\cdot\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_1)}{q(\mathbf{x}_t|\mathbf{x}_1)}-\log\frac{p_\theta(\mathbf{x}_1|\mathbf{x}_2)}{q(\mathbf{x}_2|\mathbf{x}_1)}-\log\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}\right] \tag{20}$$

$$= \mathbb{E}_q\left[-\log\frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_1)}-\sum_{t=3}^T\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_1)}-\log\frac{p_\theta(\mathbf{x}_1|\mathbf{x}_2)}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}-\log p_\phi(\mathbf{x}_0|\mathbf{x}_1)\right] \tag{21}$$

$$= \mathbb{E}_q\Bigg[\underbrace{\mathrm{KL}\left(q(\mathbf{x}_T|\mathbf{x}_1)||p(\mathbf{x}_T)\right)}_{\mathcal{L}_T}+\sum_{t=3}^T\underbrace{\mathrm{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_1)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{\mathcal{L}_{t-1}}$$
$$+\underbrace{\mathrm{KL}\left(q_\lambda(\mathbf{x}_1|\mathbf{x}_0)||p_\theta(\mathbf{x}_1|\mathbf{x}_2)\right)}_{\mathcal{L}_{\text{align}}}-\underbrace{\log p_\phi(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_{\text{rec}}}\Bigg]. \tag{22}$$

A.3. Derivation of each KL term

In line with (Ho et al., 2020), our objective exclusively involves KL divergences between Gaussians, thus facilitating closed-form evaluations.

For $\mathcal{L}_T$: The posterior distribution q lacks learnable parameters due to the deterministic forward mapping from $\mathbf{x}_1$ to $\mathbf{x}_T$. Specifically, we have $q(\mathbf{x}_T|\mathbf{x}_1)=\mathcal{N}(\mathbf{x}_T;\sqrt{\bar{\alpha}_T}\mathbf{x}_1,(1-\bar{\alpha}_T)\mathbf{I})$. When $\bar{\alpha}_T\approx 1$, this simplifies to $q(\mathbf{x}_T|\mathbf{x}_1)=\mathcal{N}(\mathbf{x}_T;\mathbf{0},\mathbf{I})$. Given this property, $\mathcal{L}_T$ remains constant during training and can be excluded from optimization.

For $\mathcal{L}_{1:T-1}$: This term aligns with the conventional diffusion model. Given the two conditional Gaussian distributions, $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_1)$ is derived as

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_1) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_1), \tilde{\beta}_t), \tag{23}$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_1) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_1 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t, \tag{24}$$

$\mathcal{L}_{t-1}$ can be compactly represented as:

$$\mathcal{L}_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_1) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C, \tag{25}$$

where C is a constant, independent with θ. We can expand Eq. (25) further by reparameterizing $q(\mathbf{x}_t|\mathbf{x}_1)$ as $\mathbf{x}_t(\mathbf{x}_1, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_1 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and reparemetrizing $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ as $\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))$:

$$\mathcal{L}_{t-1} = \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\gamma_t \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t}\mathbf{x}_1 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t \right) \right\|^2 \right], \tag{26}$$

where $\gamma_t = \frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}$.

For $\mathcal{L}_{\text{align}}$: This loss serves to align the latent representations derived from both the encoder and the diffusion model, ensuring consistent latent spaces across the framework. Both $p_\theta(\mathbf{x}_1|\mathbf{x}_2)$ and $q_\lambda(\mathbf{x}_1|\mathbf{x}_0)$ are Gaussian distributions. Therefore, the KL divergence has the similar form as Eq. (25), while the $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_1)$ becomes the mean of $q_\lambda(\mathbf{x}_1|\mathbf{x}_0)$, which corresponds to $\mathcal{E}_\lambda(\mathbf{x}_0)$. So the loss function becomes:

$$\mathcal{L}_{\text{align}} = \mathbb{E}_q \left[\frac{1}{2\sigma_1^2} \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2 \right] \tag{27}$$

$$= \mathbb{E}_q \left[\frac{\beta_1^2}{2\sigma_1^2\alpha_t(1-\bar{\alpha}_1)} \frac{\alpha_1(1-\bar{\alpha}_1)}{\beta_1^2} \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2 \right] \tag{28}$$

$$= \mathbb{E}_q \left[\gamma_1 \cdot \rho \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2 \right] \tag{29}$$

where γ_1 is consistent with Eq. (26), $\rho := \frac{\alpha_1(1-\bar{\alpha}_1)}{\beta_1^2}$ is a constant, and $\boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)$ is reparemeterized under the same reparemetrization trick used in $\mathcal{L}_{1:T-1}$, i.e., $\boldsymbol{\mu}_\theta(\mathbf{x}_2, 1) = \frac{1}{\sqrt{\bar{\alpha}_1}}(\mathbf{x}_2 - \sqrt{1-\bar{\alpha}_1}\boldsymbol{\epsilon}_\theta(\mathbf{x}_2, 1))$.

For L_{rec} : This is the reconstruction loss corresponding to the one in VAE. Depending on the data type, it can be formulated using different loss functions. In our model, we employ MSE loss for continuous data like images and cross-entropy for discrete data, including texts and proteins.

A.4. Derivation of our Final loss

The final objective in Eq. (7) in can be reformulated as:

$$\mathbb{E}_{q,\boldsymbol{\epsilon}} \left[\sum_{t=3}^{T} \gamma_t \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right\|^2 + \gamma_1 \cdot \rho \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2 + \mathcal{L}_{\text{rec}} \right] \tag{30}$$

$$= \mathbb{E}_{q,\boldsymbol{\epsilon}} \left[\sum_{t=3}^{T} \gamma_t \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right\|^2 + \gamma_1 \cdot \rho \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2 + T \cdot \frac{1}{T}\mathcal{L}_{\text{rec}} \right] \tag{31}$$

$$= \mathbb{E}_{q,\boldsymbol{\epsilon}} \left[\sum_{t=3}^{T} \gamma_t \left(\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right\|^2 + \frac{1}{\gamma_t T}\mathcal{L}_{\text{rec}} \right) + \gamma_1 \left(\rho \|\mathcal{E}_\lambda(\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_2, 1)\|^2 + \frac{1}{\gamma_1 T}\mathcal{L}_{\text{rec}} \right) \right] \tag{32}$$

We further introduce a hyperparameter w to balance the diffusion/align loss with the reconstruction loss, which gives us the following loss:

$$\mathcal{L}^{\text{final}} = \mathbb{E}_{q,\boldsymbol{\epsilon}}\left[\sum_{t=3}^{T} \underbrace{\gamma_t\left(w\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t,t\right)\right\|^2+\frac{1}{\gamma_t T}\mathcal{L}_{\text{rec}}\right)}_{\mathcal{L}_{t-1}^{\text{final}}} + \underbrace{\gamma_1\left(w\left(\rho\|\mathcal{E}_\lambda(\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_2,1)\|^2\right)+\frac{1}{\gamma_1 T}\mathcal{L}_{\text{rec}}\right)}_{\mathcal{L}_1^{\text{final}}}\right]. \tag{33}$$

A.5. Derivation of our loss from a learnable prior perspective

The following is a derivation of our loss from the VAE perspective as shown in §3.3, the objective from a learnable prior perspective.

$$\begin{aligned}
\mathcal{L}(\lambda,\phi,\theta;\mathbf{x}_0) &= \mathbb{E}_q\left[-\log\frac{p_{\phi,\theta}(\mathbf{x}_{0:T})}{q_\lambda(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] && (34)\\
&= \mathbb{E}_q\left[-\log\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)p_\theta(\mathbf{x}_{1:T})}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)q(\mathbf{x}_{2:T}|\mathbf{x}_1)}\right] && (35)\\
&= \mathbb{E}_q\left[-\log\frac{p_\phi(\mathbf{x}_0|\mathbf{x}_1)p_\theta(\mathbf{x}_1)}{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}\right] && (36)\\
&= -\mathbb{E}_{q_\lambda(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\phi(\mathbf{x}_0|\mathbf{x}_1)] + \text{KL}(q_\lambda(\mathbf{x}_1|\mathbf{x}_0)||p_\theta(\mathbf{x}_1)) && (37)
\end{aligned}$$

B. Algorithm

Below shows the complete training algorithm of EDDPMs.

Algorithm 1 Training

repeat
  $\mathbf{x}\sim q(\mathbf{x})$
  $\boldsymbol{\epsilon}_0\sim\mathcal{N}(0,\mathbf{I})$
  $\mathbf{x}_1=\mathcal{E}_\lambda(\mathbf{x}_0)+\beta_0\boldsymbol{\epsilon}_0$
  $t\sim$ Uniform$(\{1,\dots,T-1\})$
  $\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})$
  $\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\mathbf{x}_1+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
  if t == 1 **then**
    $\boldsymbol{\mu}_\theta(\mathbf{x}_2,1)=\frac{1}{\sqrt{\bar{\alpha}_1}}(\mathbf{x}_2-\sqrt{1-\bar{\alpha}_1}\boldsymbol{\epsilon}_\theta(\mathbf{x}_2,1))$
    Take gradient descent step on $\mathcal{L}_1^{\text{final}}$
  else
    Take gradient descent step on $\mathcal{L}_t^{\text{final}}$
  end if
until converged

C. Details of Experiments

This section is the counterpart of the experiment section §4.

C.1. Text

C.1.1. Setup

Model Architecture We closely adhere to the experimental setup presented in LatentOps (Liu et al., 2023a). For our encoder, denoted as $\mathcal{E}_\lambda$, we utilize the BERT-small model[1](Devlin et al., 2019; Bhargava et al., 2021). As for the decoder,

[1]The BERT model follows the Apache 2.0 License.

represented as $\mathcal{D}_\phi$, we employ the GPT2-xl architecture[2](Radford et al., 2019). Our diffusion model is constructed using a straightforward MLP with skip connections, as inspired by (Preechakul et al., 2022). The latent dimension is set to 128.

Building upon the methodologies of Li et al. (2020); Liu et al. (2023a), we equip the pretrained language model (LM) with a linear layer that precedes the LM, facilitating the passage of $\mathbf{x}_1$ to the decoder. To maintain generative capabilities and acclimate the LM to the latent space, we incorporate an additional transformer layer between the original first layer and the embedding layer of the LM, fostering adaptability. During the training phase, our optimization is confined to the MLP layers, the embedding layer, the newly inserted transformer layer, the encoder, and the diffusion model with all other parameters remaining frozen.

Dataset Regarding our dataset selection, we commence with the bookcorpus dataset (Zhu et al., 2015) to train the autoencoder in the absence of the diffusion model. Subsequently, we engage in joint training of the model with diffusion, utilizing the Yelp review dataset[3] (Shen et al., 2017), which has been preprocessed by Li et al. (2018). It's noteworthy that Yelp serves as a sentiment dataset, encompassing approximately 179K negative and 268K positive sentences.

Baselines To ensure a comprehensive and fair comparison, our approach is primarily benchmarked against two key models: LatentOps (Liu et al., 2023a) and Optimus-DAAE (Shen et al., 2020; Li et al., 2020). We have maintained consistency in both architecture and training procedures across these comparisons. The primary difference lies in the training objectives. Optimus-DAAE represents a modified, large-scale autoencoder. It enhances text generation capabilities by reconstructing sentences from their slightly altered versions and integrates adversarial training with a denoising objective.

In addition, we have conducted comparisons with the latest advancements in text diffusion models, namely GENIE (Lin et al., 2023) and AR-Diffusion (Wu et al., 2023). GENIE stands out as a non-autoregressive, large-scale pre-training text diffusion framework designed for text generation. AR-Diffusion, on the other hand, is an auto-regressive text diffusion model characterized by its multi-level diffusion strategy, which operates at both sentence and token levels. To evaluate these models comprehensively, we trained both GENIE and AR-Diffusion using the Yelp dataset and applied them to various downstream tasks. In addition to the text diffusion models, our comparative analysis extends to include two state-of-the-art purely auto-regressive models: GPT2 and GPT4 (Achiam et al., 2023). Specifically, we adapted GPT2 for our experiments by fine-tuning it on the Yelp dataset; however, its application was limited to text generation tasks due to its model architecture and capabilities. Conversely, GPT4 was evaluated across a broader range of tasks, utilizing a diverse set of prompts designed to explore its extensive generative potential and versatility.

Tasks In our study, we aim to comprehensively assess the holistic performance of EDDPMs by evaluating it across three foundational capabilities: generation, reconstruction, and latent vector arithmetic. To achieve this, we focus on four specific tasks: generation, reconstruction, interpolation, and text style transfer using latent vector arithmetic. Each of these tasks is selected for the following reasons:

- **Good Reconstruction:** A key aspect of our model's performance is its ability to accurately preserve content. This is particularly evident in its reconstruction capability, which is critical for tasks like text style transfer and interpolation, where maintaining the original content's integrity is essential.

- **Robust Representation:** Our model aims to do more than just preserve content; it strives to capture the intrinsic meaning, nuances, and essential features of the input text. This ability is vital for tasks requiring a deep understanding of the source material, such as text style transfer.

- **Fluent Generation:** The ultimate test of our model's effectiveness lies in the quality of its output. For EDDPM, the fluency and coherence of the generated text are crucial indicators of its robust generative capabilities. This is demonstrated in generation tasks, text style transfer, and interpolation.

By evaluating EDDPM on these four tasks, we can effectively reflect and analyze its performance in terms of the three core functionalities.

[2]The GPT2 model follows the MIT License.

[3]The datasets are distributed under CC BY-SA 4.0 license.

C.1.2. Detailed Results

We present detailed results of our text experiments. The results for reconstruction, generation, and style transfer are displayed in Table 4. Additionally, results of interpolation are separately provided in two tables: Table 5 focuses on content preservation, while Table 6 addresses fluency and divergence.

	Reconstruction	Generation		Style Transfer (Arithmetic)		
	Content↑	Fluency↓	Divergence↑	Accuracy↑	Attribute↑	Content↑
LatentOps (Liu et al., 2023a)	87.6	68.1	0.240	57.3	71.5	45.9
Optimus-DAAE (Shen et al., 2020)	86.1	94.1	0.006	51.0	74.7	34.8
GENIE (Lin et al., 2023)	58.5	337.6	0.013	33.9	20.8	55.2
AR-Diffusion (Wu et al., 2023)	64.1	157.8	0.007	17.0	4.6	62.7
GPT2 (Radford et al., 2019)	-	15.04	0.015	-	-	-
GPT4 (Achiam et al., 2023)	100	25.65	0.007	49.3	80.5	30.1
EDDPM	92.1	16.4	0.977	57.1	77.6	42.1

Table 4. Evaluation of Text Reconstruction, Generation and Style Transfer. Accuracy of style transfer is the geometric mean of attribute score (classifier) and content score (BLEU).

Models	$\alpha = 0.0/1.0$	$\alpha = 0.1/0.9$	$\alpha = 0.2/0.8$	$\alpha = 0.3/0.7$	$\alpha = 0.4/0.6$	$\alpha = 0.5$
LatentOps (Liu et al., 2023a)	0.0 / 92.0	0.0 / 90.1	0.0 / 83.7	0.8 / 63.2	0.9 / 35.2	9.5
Optimus-DAAE (Shen et al., 2020)	0.0 / 91.4	0.0 / 87.2	0.0 / 73.5	0.5 / 46.5	2.2 / 21.0	7.8
GENIE (Lin et al., 2023)	0.8 / 66.2	0.6 / 67.8	0.8 / 65.6	0.8 / 66.6	0.5 / 52.1	10.1
AR-Diffusion (Wu et al., 2023)	0.7 / 60.9	0.9 / 87.3	0.9 / 86.9	1.0 / 84.8	0.6 / 62.2	15.9
EDDPM	0.0 / 97.2	0.0 / 95.6	0.3/ 91.7	0.4 / 69.9	3.0 / 30.3	11.5

Table 5. Evaluation of Text Interpolation Content Consistency. This table quantifies the consistency of interpolated text between two input sentences using BLEU scores. The degree of interpolation is denoted by α. A higher BLEU score indicates greater text consistency with the corresponding input sentence.

Models	$\alpha = 0.0/1.0$	$\alpha = 0.1/0.9$	$\alpha = 0.2/0.8$	$\alpha = 0.3/0.7$	$\alpha = 0.4/0.6$	$\alpha = 0.5$
LatentOps (Liu et al., 2023a)	19.5 / **0.871**	19.5 / **0.886**	19.7 / 0.867	21.3 / 0.798	26.9 / 0.856	32.5 / 0.697
Optimus-DAAE (Shen et al., 2020)	20.2 / 0.848	20.4 / 0.847	20.9 / 0.849	26.0 / 0.799	31.6 / 0.655	33.7 / 0.623
GENIE (Lin et al., 2023)	49.8 / 0.013	45.4 / 0.010	46.9 / 0.015	45.0 / 0.008	77.4 / 0.027	300.0 / 0.029
AR-Diffusion (Wu et al., 2023)	63.1 / 0.008	25.9 / 0.007	26.3 / 0.006	29.1 / 0.008	60.4 / 0.011	156.0 / 0.012
GPT4(Achiam et al., 2023)	-	-	-	-	-	39.7 / 0.010
EDDPM	**17.8** / 0.867	**17.9** / 0.884	**18.1 / 0.868**	**19.9 / 0.910**	**26.5 / 0.909**	**30.8 /0.763**

Table 6. Evaluation of Text Interpolation Fluency and Divergence. The table presents perplexity scores / MAUVE scores. Lower perplexity scores signify higher fluency and higher MAUVE scores represent smaller gap with the training data.

C.2. Image

C.2.1. Setup

Model Architecture In line with the architecture presented by Diffusion Autoencoders (DiffAE) (Preechakul et al., 2022), our model is structured with an encoder designed as a UNet and a decoder functioning as a conditional diffusion-based model at the pixel level. Given the latent semantic representation $\mathbf{x}_1$ and a random Gaussian sample $\mathbf{x}_T$ that shares the same dimensionality as the raw data $\mathbf{x}$, the decoder employs reverse diffusion transitions to produce the output $\hat{\mathbf{x}}$. Complementing this, we integrate an additional standard diffusion process, with transitions realized through a straightforward MLP fortified with skip connections.

Dataset Following the approach of DiffAE, we train our model and subsequently evaluate its reconstruction and generation capabilities on FFHQ (Karras et al., 2019), CelebA (Karras et al., 2018), LSUN-Bedroom and LSUN-Horses (Yu et al.,

2015). To assess the representation ability, we employ the CelebA-HQ dataset (Karras et al., 2018).

Hyperparameter settings We trained our model on two Nvidia A100-SXM4-40GB GPUs with a batch size of 100. For evaluation purposes, we sampled 50,000 images to compute the FID, setting total steps $T = 100$ for both the diffusion process and the decoder at every 500,000 training steps interval. The optimization was carried out using the Adam Optimizer, with a learning rate of 1×10^{-4} and no weight decay. The image dimensions inputted into the model were consistently set at 128×128 for FFHQ and 64×64 for CelebA.

Evaluation Metrics For the assessment of the generated images' quality, we resort to the Fréchet Inception Distance (FID) (Heusel et al., 2017), a widely-accepted metric in the field. To assess the fidelity of our reconstruction, we employ the reconstruction-FID (rFID) metric.

Baselines In our experiments, we compare our method against the following prominent baselines:

1. **DDIM** (Song et al., 2021a): Following the implementation by (Preechakul et al., 2022), we ensure a fair comparison by adopting the DDIM approach described therein.

2. **DiffAE** (Preechakul et al., 2022): In this baseline, the encoder-decoder structure is trained in isolation, separate from the latent diffusion model.

3. **StyleGAN-XL** (Sauer et al., 2022): StyleGAN-XL is the scaled version of StyleGAN3 generator on Imagenet, achieving state-of-the-art results on large-scale image synthesis. The provided checkpoint on FFHQ is of resolution 256, thus we perform all the experiments with 256×256 but resize the images into 128×128 for FID calculation.

4. **NVAE** (Vahdat & Kautz, 2020b): Nouveau VAE (NVAE) is a deep hierarchical VAE built for image generation.

5. **Consistency Models**: Consistency Models (CM) are proposed to overcome the limitation that diffusion models are too slow during generation. It could achieve new state-of-the-art results with one-step generation. In our experiments, we adopt the best setting which is run by two steps. Specifically, we use the model `openai/diffusers-cd_bedroom256_lpips` in our experiments as it performs better than `openai/diffusers-cd_bedroom256_l2` (Generation FID of the former is lower). Similar to StyleGAN-XL, we conduct all the experiments in the resolution 256×256 and calculate FID with resolution 128×128.

6. **Latent Diffusion Model (LDM)** (Rombach et al., 2021): For this method, the Variational AutoEncoder (VAE) is pretrained prior to training the latent diffusion model. We leverage the pre-trained weights from (Rombach et al., 2021) to generate images with a resolution of 224×224. Subsequent to generation, these images are downscaled to a 64×64 resolution for Fréchet Inception Distance (FID) computation. For tasks including reconstruction, interpolation, and manipulation, the VAE from the LDM approach serves as the benchmark.

C.2.2. Detailed Experimental Results

Detailed Overall Performance Comparison for Generation, Reconstruction, and Interpolation. The results are shown in Table 7. Note that we draw Figure 4 according to the generation, reconstruction performances, and interpolation results with $\alpha = 0.4$. As for the results of CM-CD, the reported FID for generation in the original paper (Song et al., 2023) is 5.22, whereas we obtained an FID of 7.01. This discrepancy arises due to two main reasons: (1) We calculate the reference statistics (mean and variance) using the original images, resulting in slightly different statistics compared to those provided here[4]. (2) The original model was trained on images with a resolution of 256x256. Consequently, we need to convert the generated 256x256 images to 128x128 for evaluation, which may introduce differences in FID. When using the pre-calculated statistics and evaluating the model at a resolution of 256x256, we obtain an FID of 5.82, which is close to the 5.22 reported in the original paper. Using the newly calculated statistics, the FID is 6.56. Upon converting the images to 128x128, the FID becomes 7.01.

[4] https://github.com/openai/guided-diffusion/tree/main/evaluations

Dataset	Model	Generation FID			Reconstruction rFID			$\alpha = 0.2$			$\alpha = 0.4$		
		T=10	T=20	T=50	T=10	T=20	T=50	T=10	T=20	T=50	T=10	T=20	T=50
FFHQ 128	LDM	67.78	30.43	12.90	—	<u>4.87</u>	—	—	21.29	—	—	75.13	—
	NVAE	—	41.26	—	—	5.73	—	—	<u>6.28</u>	—	—	<u>9.34</u>	—
	StyleGAN-XL	—	24.72	—	—	34.09	—	—	46.82	—	—	59.64	—
	DDIM	29.56	21.45	15.08	88.22	45.30	22.23	144.40	104.91	75.81	181.07	131.80	105.31
	DiffAE	20.80	16.70	12.57	12.59	9.23	5.93	13.25	11.33	9.38	22.93	23.01	22.17
	EDDPM	**18.41**	**14.38**	**12.26**	**11.50**	**8.17**	**5.48**	**12.57**	**9.80**	**6.66**	**19.11**	**18.36**	**16.98**
CelebA 64	LDM	41.87	31.40	25.80	—	9.58	—	—	9.95	—	—	28.78	—
	NVAE	—	23.11	—	—	<u>1.55</u>	—	—	<u>3.03</u>	—	—	<u>8.49</u>	—
	DDIM	16.38	12.70	8.52	78.44	20.20	16.76	148.22	80.06	51.84	163.26	103.01	82.77
	DiffAE	12.92	10.18	7.05	14.14	10.09	5.87	11.09	9.30	6.90	17.13	16.82	15.35
	EDDPM	**12.35**	**9.49**	**6.65**	**11.84**	**8.60**	**5.15**	**9.75**	**8.68**	**6.23**	**16.12**	**16.56**	**14.85**
Horse	DDIM	22.17	12.92	7.92	157.24	73.50	22.59	266.54	243.47	181.69	316.98	301.88	226.54
	DiffAE	11.97	9.37	7.44	9.62	6.83	4.71	12.41	10.07	7.88	24.04	21.19	18.34
	EDDPM	12.53	9.80	8.90	**9.15**	**6.36**	**3.98**	**10.45**	**8.31**	**6.19**	**19.64**	**17.18**	**14.55**
Bedroom	LDM	41.67	10.86	**4.39**	—	<u>2.28</u>	—	—	5.80	—	—	67.42	—
	CM-CD	—	7.01	—	—	29.53	—	—	169.57	—	—	241.90	—
	DDIM	13.70	9.23	7.14	108.72	59.11	11.81	190.04	157.81	76.28	250.50	224.96	139.81
	DiffAE	**10.69**	8.19	6.50	7.10	5.26	4.13	8.42	7.08	6.00	15.15	13.32	12.01
	EDDPM	11.01	**8.03**	6.35	**6.21**	**4.77**	**3.49**	**7.26**	**6.31**	**5.39**	**13.20**	**11.83**	**10.58**

Table 7. Overall performance comparison in various tasks. We use FID and rFID to evaluate the performances of the generation and reconstruction, respectively. For the interpolation tasks, we test two settings $\alpha = 0.2$ and $\alpha = 0.4$, where FID is also used to evaluate the performance. We highlight the best performance within the performances of methods that need to set up the parameter T with **bold**. The performances of the methods with only one step such as NVAE, StyleGAN-XL, and CM-CD are <u>underlined</u> if their performances are better than the best performances of other methods with $T = 50$.

C.2.3. Image Generation

For FFHQ128, we present the generated images in Figure 9, 10, 11). Then for CelebA64, the generated images are shown in Figure 12, 13, 14. As depicted in the figures, images generated with $T = 50$ typically exhibit finer granularity when contrasted against those produced with $T = 10$ or $T = 20$.

C.2.4. Image Reconstruction

For a comparative analysis, we present reconstructed images from various models, each utilizing a distinct total step T in the decoder. The results for FFHQ128 and CelebA64 are depicted in Figure 15 and Figure 16, respectively. From the figures, it's clear that the VAE in LDM is effective at reconstruction. Our model EDDPM also produces strong results.

C.2.5. Image Representation

Interpolation The interpolation results for FFHQ128 and CelebA64 are illustrated in Figure 17 and Figure 18, respectively. A close examination of the figures reveals that while the VAE in LDM is adept at reconstruction, its interpolation with $\alpha = 0.4$ appears akin to a superposition of two images. This aligns with the inherent nature of their VAE, where the representation predominantly encodes spatial rather than semantic information. In contrast, both our approach and DiffAE yield superior results at $\alpha = 0.4$. Specifically, our method demonstrates fewer visual artifacts compared with DiffAE, underscoring the better representation space of our model.

Manipulation The comprehensive comparisons are presented in Table 8. As indicated by the table, our model consistently delivers comparable AUC values across all classes.

C.3. Protein

Protein design plays a crucial role in drug discovery, protein therapeutics and various other applications in biotechnology. However, due to the complex and large search space of protein sequences, traditional empirical methods that demands intensive and thorough experiments and screening for validation are expensive and time-consuming. Recent advances in

Figure 9. FFHQ128, T=10

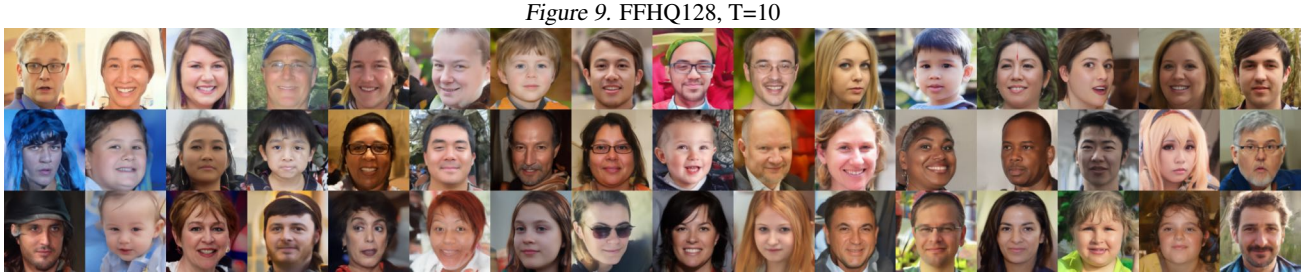

Figure 10. FFHQ128, T=20

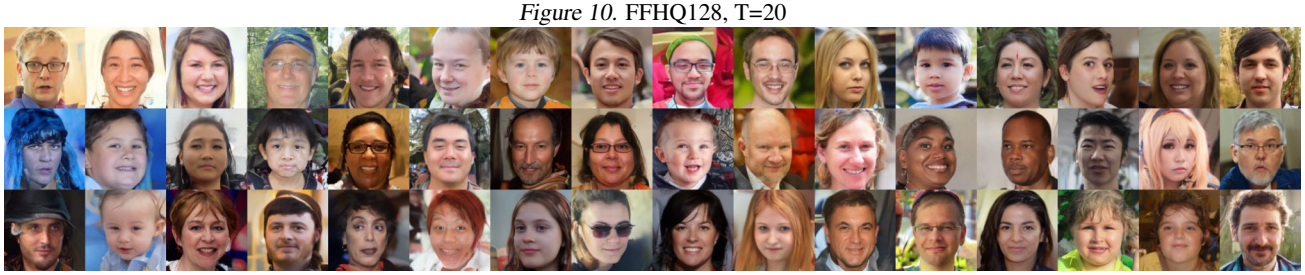

Figure 11. FFHQ128, T=50

machine learning and computational methods introduces new approaches for protein optimization, however, most of these methods focuses on protein design in the discrete text space and lacks a meaningful latent space. Our work combines the Autoencoder and Diffusion Model, enabling effective protein generation/optimization and establishing a robust latent space for protein representation.

C.3.1. SETUP

Architecture We adopt the set up of ReLSO (Castro et al., 2022) which consists of a simple transformer as the encoder and convolutional layers as the decoder. While ReLSO relies on Negative Sampling Loss which augments the datasets with synthetic samples with negative labels and Interpolative Regularization which penalizes differences between interpolated points and nearest neighbors to achieve a smooth pseudo-convex latent space, we leverage the same MLP with skip connections in our text model as the backbone of the diffusion model to act as a learnable prior. In line with ReLSO, we jointly train a simple regressor consists of linear layers and ReLU that predicts the fitness value with the latent embedding of a protein sequence. The regressor is trained with Mean Squared Error and added to final objective derived in §A.4 :

$$\mathcal{L}_t^{\text{protein}} = w_{\text{protein}} * \mathcal{L}^{\text{final}} + \mathcal{L}^{\text{regressor}}, \tag{38}$$

The dimension of the latent space is set to 30. And we have trained 2 separate models with w_{protein} set to 1 and 5.

Dataset The models are trained and evaluaed on the Gifford Liu et al. (2019) dataset and the GFP(Sarkisyan et al., 2016) dataset. The Gifford dataset was generated from directed evolution of 10^{10} mutants of an antibody against a single target(Ranibizumab) through three rounds of phage display panning. Fitness value is defined as the log of the round-to-round

Figure 12. CelebA64, T=10

Figure 13. CelebA64, T=20

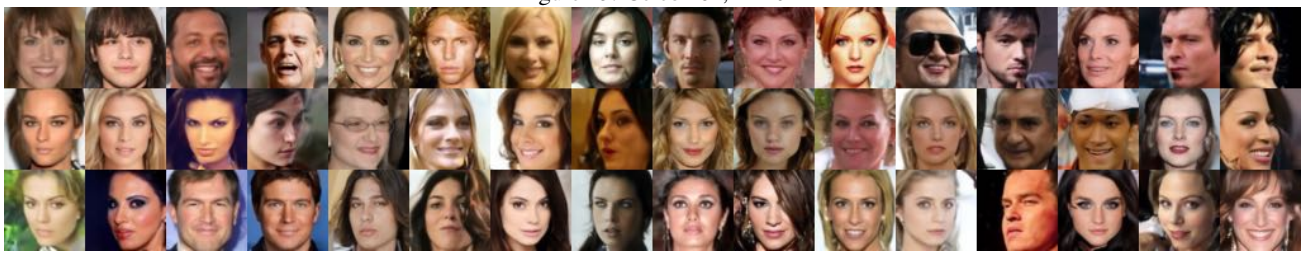

Figure 14. CelebA64, T=50

ratio of sequence frequencies and a higher fitness value indicate better performance. In this dataset, each protein sequence has a length of 20 with a vocabulary of 20 amino acids which are represented by 20 letters. The resulting dataset consists of 57603 sequences in the training set, 10166 sequences in the validation set, and 22690 sequences in the test set. The GFP dataset was sampled from a fluorescent protein(avGFP) through mutagenesis. The dataset consists of 51175 sequences with length of 237 and an average mutation of 3.7. The fitness is defined as the fluorescence of the proteins assayed with fluorescence-activated cell sorting of the sequences.

Baseline In addition to ReLSO, an autoencoder-based model introduced above, we trained a vanilla Variational Autoencoder(Kingma & Welling, 2014) consists of the same transformer as encoder and convolutional layers as decoder. The dimension of the latent space of ReLSO and VAE are all set to 30. Furthermore, we have conducted comparisons with latest advancements in protein sequence diffusion models, namely NOS (Gruver et al., 2023). NOS proposes a Discrete Diffusion Model that based on BERT and utilized [MASK] as noise, and a Gaussian Diffusion Model that jointly learned a word embedding function and used BERT as the backbone of the diffusion model. While ReLSO, VAE and EDDPM utilized a regressor of linear layers and ReLU, the regressor of NOS required an additional BertPooler to aggregate a sequence of latent embeddings. We have conducted experiments on Protein Representation with EDDPM, ReLSO, VAE and NOS; Protein Optimization with EDDPM, ReLSO and VAE; Protein Reconstruction with EDDPM, ReLSO and VAE; and Protein Generation with EDDPM, VAE and NOS.

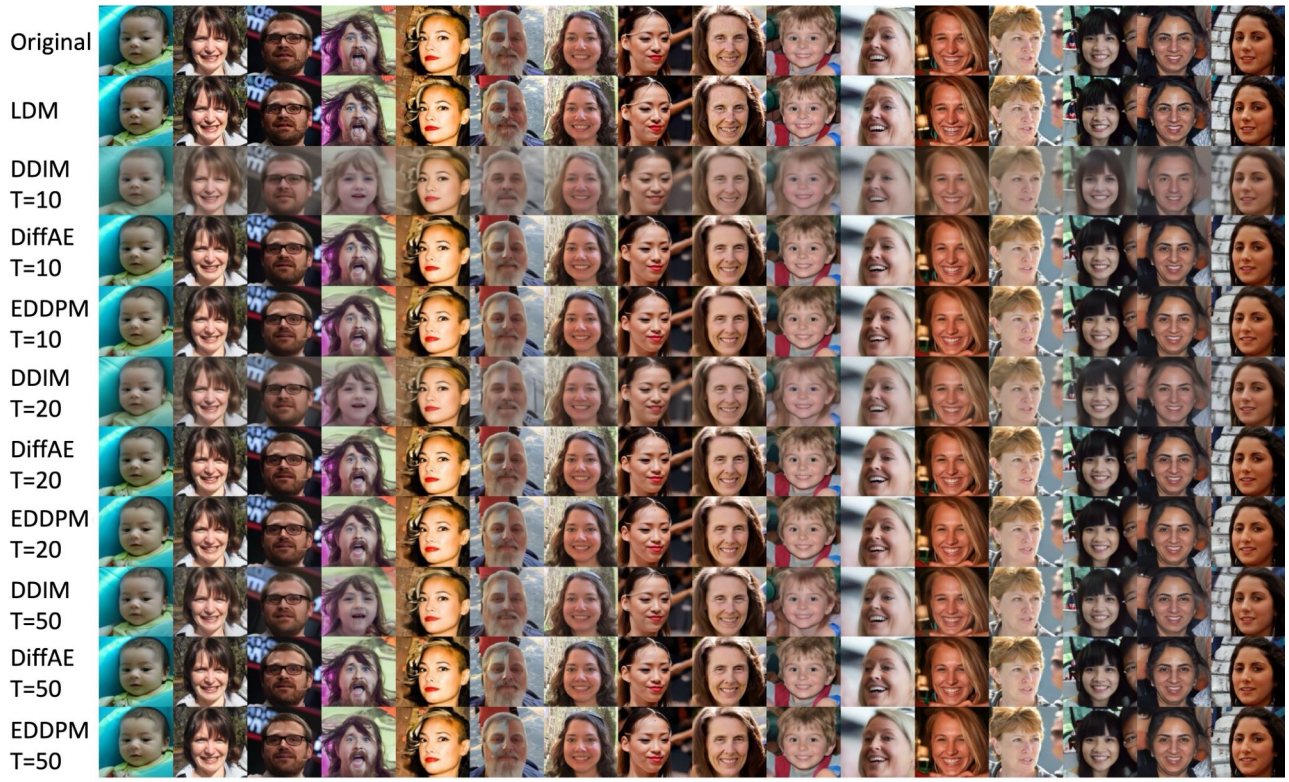

Figure 15. Image reconstructions for FFHQ128 with different models. Best viewed with zooming in.

C.3.2. Protein Optimization

We optimize a protein sequence by optimizing its corresponding embedding in the latent space. Given a protein sequence $\mathbf{x}_0$, we first obtain its latent embedding with $\mathbf{x}_1 = \mathcal{E}_\lambda(\mathbf{x}_0)$. We adopt the sampling algorithm introduced in LatentOps that solves an ODE involving the regressor Liu et al. (2023a). This approach requires a target fitness value to guide the optimization. We set this value to 1, 1.5, 2, and 2.5 for the Gifford dataset and 3, 4 for the GFP dataset which are all reasonably high fitness values in their corresponding datasets. For evaluation, we optimize 60 random protein sequence and evaluate their fitness value using the regressor. We assess the results on Diversity, quantified by the average Levenshtein distance of each sequence relative to the other 59 optimized sequences; on Novelty, determined by the median of the minimum Levenshtein distance between each optimized sequence and the training set; and on Quality, measured by the negative log likelihood given by ProtGPT2, a large protein language model trained on the UniRef50 dataset. The results are presented in Table 10, Table 11 and Table 12.

C.3.3. Protein Reconstruction

Reconstruction is evaluated with Cross Entropy to compare input and reconstructed protein sequences in the test set. The results are presented in the last column of Table 9. From Table 9, EDDPM achieves comparable reconstruction ability compared to the baselines. Note that since NOS is a transformer-based model with word embedding that could not perform reconstruction in the same settings as other models.

C.3.4. Conditional Protein Generation

For conditional generation with VAE and EDDPM, latent representations are sampled and optimized with LatentOps algorithm with target fitness of 1 and 3 for Gifford and GFP dataset respectively. For NOS, the latents are updated through out each step of the diffusion process with Langevin Dynamics using gradients of the regressor. The quantitative evaluation are presentsed in 13. Although NOS models achieved better NLL, the generated proteins are have lower fitness values than

Figure 16. Image reconstructions for CelebA64 with different models. Best viewed with zooming in.

other methods.

C.3.5. Influence of weight w

In the objective (Eq.33), the weight w aims to balance the generation and reconstruction. As mentioned in §3.1, we adopt $w = 8$ for text and $w = 1$ for both image and protein. The choice of w is primarily influenced by the different loss functions used for the reconstruction term L_{Rec} (Eq.4) across various data types. For example, cross-entropy is used for text, while mean squared error (MSE) is used for images. To ensure that the different loss terms have similar scales, we selected w values that roughly balanced their magnitudes. The w values are then fixed during training. As a comparison, beta-VAE on text typically requires careful scheduling/annealing of β values during training (Higgins et al., 2017b; Li et al., 2020), making the training more difficult and unstable.

C.3.6. Posterior Collapse

EDDPM can alleviate the *posterior collapse* issue on text sequences thanks to the improved formulation and training:

- As discussed in §3.3, EDDPMs can be viewed as a VAE with a jointly learned diffusion model prior (instead of a standard Gaussian prior in vanilla VAEs). More formally, the EDDPMs' objective in Eq.7 includes the L_{align} term that explicitly encourages alignment between the encoder's posterior distribution $q_\lambda(\mathbf{x}_1|\mathbf{x}_0)$ and the diffusion model's prior $p_\theta(\mathbf{x}_1|\mathbf{x}_2)$. This learned prior is more flexible and helps avoid the unreasonable regularization posed by standard Gaussian prior that causes posterior collapse.
- EDDPMs derive the training objective from the well-established DDPM framework, enabling stable and effective joint training of the encoder, decoder, and diffusion components. For example, as mentioned above, EDDPMs are robust to the balancing weight w. This is in contrast to the instability and sensitivity to hyperparameters in text-VAE training. To alleviate posterior collapse, training text-VAEs often requires tricks like beta-annealing (Bowman et al., 2016), free bits (Kingma et al., 2017), cyclic annealing schedule (Fu et al., 2019), and so on. This can be observed through the training loss curves in Figure 3.

Therefore, the *posterior collapse* issue is not obvious in EDDPMs.

Figure 17. Image interpolations for FFHQ128 with different models. T is fixed to 50 in these experiments. Best viewed with zooming in.

Figure 18. Image interpolations for CelebA64 with different models. T is fixed to 50 in these experiments. Best viewed with zooming in.

Class	# Positives	DiffAE	EDDPM
5_o_Clock_Shadow	1314	0.9469	0.9466
Arched_Eyebrows	3171	0.8822	0.8811
Attractive	5009	0.8849	0.8792
Bags_Under_Eyes	2528	0.8787	0.8821
Bald	174	0.9886	0.9843
Bangs	1593	0.9779	0.9770
Big_Lips	3187	0.7031	0.7108
Big_Nose	2796	0.8757	0.8683
Black_Hair	1895	0.9483	0.9427
Blond_Hair	1469	0.9775	0.9760
Blurry	27	0.8434	0.8762
Brown_Hair	2074	0.8476	0.8334
Bushy_Eyebrows	1658	0.9174	0.9117
Chubby	579	0.9439	0.9339
Double_Chin	481	0.9503	0.9486
Eyeglasses	404	0.9916	0.9877
Goatee	650	0.9724	0.9690
Gray_Hair	346	0.9865	0.9856
Heavy_Makeup	3963	0.9714	0.9695
High_Cheekbones	4043	0.9490	0.9474
Male	3184	0.9977	0.9969
Mouth_Slightly_Open	4125	0.9784	0.9777
Mustache	495	0.9573	0.9547
Narrow_Eyes	1061	0.8569	0.8600
No_Beard	7047	0.9784	0.9754
Oval_Face	1772	0.7494	0.7469
Pale_Skin	458	0.9635	0.9618
Pointy_Nose	2738	0.7263	0.7235
Receding_Hairline	718	0.9373	0.9340
Rosy_Cheeks	975	0.9482	0.9416
Sideburns	684	0.9768	0.9726
Smiling	4050	0.9827	0.9803
Straight_Hair	1866	0.8030	0.8027
Wavy_Hair	3099	0.8804	0.8770
Wearing_Earrings	2328	0.8933	0.8851
Wearing_Hat	311	0.9875	0.9862
Wearing_Lipstick	4911	0.9803	0.9789
Wearing_Necklace	1496	0.7788	0.7789
Wearing_Necktie	600	0.9583	0.9560
Young	6871	0.9230	0.9130
Weighted Avg AUC	-	**0.9174**	0.9154
Weighted Avg ACC	-	0.7954	**0.8929**

Table 8. Classification performance comparison.

Dataset	Model	MSE	L1	Pearson	Spearman	Reconstruction CE
Gifford	VAE (Kingma & Welling, 2014)	0.255	0.361	0.829	0.454	1.464
	ReLSO (Castro et al., 2022)	0.293	0.401	0.826	0.477	0.940
	NOS-Discrete (Gruver et al., 2023)	0.563	0.648	0.633	0.407	-
	NOS-Gaussian (Gruver et al., 2023)	0.753	0.785	0.350	0.292	-
	EDDPM (w_{protein}=0.5)	0.231	0.346	0.840	0.477	1.016
	EDDPM (w_{protein}=5)	0.211	0.331	0.844	0.472	1.143
GFP	VAE (Kingma & Welling, 2014)	0.681	0.615	0.873	0.764	0.093
	ReLSO (Castro et al., 2022)	0.516	0.592	0.793	0.701	0.099
	NOS-Discrete (Gruver et al., 2023)	7.984	2.620	0.378	0.350	-
	NOS-Gaussian (Gruver et al., 2023)	8.040	2.630	-0.124	-0.129	-
	EDDPM (w_{protein}=0.5)	0.496	0.464	0.854	0.753	0.095
	EDDPM (w_{protein}=0.5)	0.488	0.440	0.848	0.734	0.094

Table 9. Protein Representation and Reconstruction

Dataset	Model	Max Fitness	Mean Fitness	Diversity	Novelty	NLL
Gifford	VAE (Kingma & Welling, 2014)	2.005	1.969	14.789	7	29.301
	ReLSO (Castro et al., 2022)	0.738	0.737	2.14	7	28.736
	EDDPM (w_{protein}=0.5)	2.003	2.000	12.615	6	29.219
	EDDPM (w_{protein}=5)	2.000	2.000	12.256	4	29.452
GFP	VAE (Kingma & Welling, 2014)	4.007	4.000	2.743	1	23.476
	ReLSO (Castro et al., 2022)	2.875	2.524	1.70	1	23.468
	EDDPM (w_{protein}=0.5)	4.001	4.000	12.836	1	23.488
	EDDPM (w_{protein}=5)	4.001	4.000	1.076	1	23.480

Table 10. Comparison of Protein Optimization.

Target Fitness	Model	Max Fitness	Mean Fitness	Diversity	Novelty	NLL
1	ReLSO	0.738	0.737	2.70	6	28.645
	EDDPM (w_{protein}=0.5)	1.000	1.000	13.35	6	29.352
1.5	ReLSO	0.738	0.737	2.06	7	28.713
	EDDPM (w_{protein}=0.5)	1.501	1.500	13.07	6	29.294
2	ReLSO	0.738	0.737	2.14	7	28.736
	EDDPM (w_{protein}=0.5)	2.003	2.000	12.61	6	29.219
2.5	ReLSO	0.738	0.737	2.26	7	28.790
	EDDPM (w_{protein}=0.5)	2.504	2.500	12.30	6	28.736

Table 11. Comparison of Protein Optimization with different Target Fitness Value on Gifford.

Target Fitness	Model	Max Fitness	Mean Fitness	Diversity	Novelty	NLL
3	ReLSO	2.873	2.872	1.70	1	23.468
	EDDPM (w_{protein}=0.5)	3.001	3.000	3.41	1	23.478
4	ReLSO	2.873	2.872	1.08	1	23.469
	EDDPM (w_{protein}=0.5)	4.000	4.000	12.83	1	23.488

Table 12. Comparison of Protein Optimization with different Target Fitness Value on GFP.

Dataset	Model	Max Fitness	Mean Fitness	Diversity	Novelty	NLL
Gifford	VAE (Kingma & Welling, 2014)	1.010	0.999	15.771	9	29.389
	NOS-Discrete (Gruver et al., 2023)	1.650	0.702	14.549	10	29.01
	NOS-Gaussian (Gruver et al., 2023)	0.493	-0.112	8.160	3	29.05
	EDDPM (w_{protein}=0.5)	1.003	1.000	13.067	6	29.200
	EDDPM (w_{protein}=5)	1.004	1.000	12.486	5	29.490
GFP	VAE (Kingma & Welling, 2014)	3.036	3.000	114.2	105.5	23.702
	NOS-Discrete (Gruver et al., 2023)	0.056	0.023	22.450	220	10.218
	NOS-Gaussian (Gruver et al., 2023)	-0.005	-0.007	3.626	220	10.066
	EDDPM (w_{protein}=0.5)	3.007	3.001	4.173	1	23.481
	EDDPM (w_{protein}=5)	3.003	3.002	0.715	1	23.475

Table 13. Comparison of Protein Conditional Generation.

D. Contrasting with Similar Works

In §3.3, we provide a high-level comparison of EDDPMs with many related models, highlighting the connections and differences between our approach and various existing methods. In §5, we present additional discussion of related works, focusing on recent efforts to combine VAEs, GANs, and diffusion models. We further mention the limitations of these approaches and how EDDPMs aim to overcome them. In this section, we would like to offer an in-depth discussion of EDDPMs with two closely related methods: Latent Diffusion Models and Latent Score-based Generative Models.

D.1. Latent Diffusion Models

Latent Diffusion Models (LDMs), often termed Stable Diffusion (Rombach et al., 2021), utilize an architectural combination of an autoencoder and a diffusion model in latent space. In this section, we delineate the primary distinctions between LDMs and our proposed approach in detail:

Autoencoder's Functionality: *LDMs:* The overarching objective of their autoencoder is twofold: to compress images into a compact latent representation and to ensure robust reconstruction capabilities from these latent vectors. To bias the autoencoder towards stronger reconstruction, they introduce a KL term but assign it a minute KL weight ($\sim 10^{-6}$). Their employment of a purely convolution-based encoder-decoder emphasizes spatial preservation in the latent space, which, while bolstering reconstruction, poses challenges to distilling semantically rich latent features.

Our Approach: Our autoencoder extends beyond mere dimensionality reduction. It's intricately tailored to synchronize effectively with the diffusion process, thereby fostering a more semantically-coherent latent space. Instead of relying on the conventional KL regularization against a standard Gaussian, we harness a learnable prior, rendering our latent space more adaptive and insightful.

Training Paradigm: *LDMs:* Their training strategy bifurcates into two discrete phases: initial autoencoder training followed by subsequent training of the diffusion model in latent space. Consequently, the latent space's architecture predominantly adheres to the objectives set forth by the autoencoder.

Our Approach: Our methodology pivots on end-to-end training, ensuring the latent space's structure is sculpted by the holistic objectives of the entire model. This integrated approach instills the latent space with nuanced semantics and more discernible significance, enhancing both interpretability and utility.

D.2. Latent Score-based Generative Model

The Latent Score-based Generative Model (LSGM) enhances generative capabilities by learning Score-based Generative Models (SGM) within the latent space of a Variational Autoencoder (VAE). We will delineate the detailed differences between LSGM and EDDPMs in the following sections.

Training Objective: *LSGM:* Their method can be regarded as a Variational Autoencoder (VAE) with a Score-based Generative Model (SGM) as the prior. Consequently, their objective is derived from Eq. (37). However, they uniquely decompose the KL term into two components, and each component is approximated based on certain assumptions. This approach results in a fundamentally different final training objective to that of EDDPMs.

Our Approach: Our method begins with a foundational Diffusion model framework. In this context, we interpret the encoder and decoder as an extended step in the diffusion process. We then formulate a novel, generalized diffusion process that incorporates a learnable noise addition step. The derivation of our training objective strictly adheres to the conventional principles of standard diffusion models and does not rely on further approximations.

Training Paradigm: *LSGM:* In their approach, there are two distinct training objectives: one for the VAE and another for the SGM. Although both losses are computed and used to update the model parameters, they are theoretically independent. For instance, the SGM objective does not update the weights of the VAE. This implies that the latent space of the VAE is not actively regularized by the SGM; rather, the SGM learns the latent distribution as defined by the VAE. Furthermore, the training process for their model incorporates some complex tricks, resulting in instability. This complexity makes it challenging to train their model effectively, a point acknowledged by the authors themselves in their repository `https://github.com/NVlabs/LSGM?tab=readme-ov-file#common-issues`.

Our Approach: Contrarily, our method employs a singular, unified training objective for the entire model, which is trained in an end-to-end manner. The latent diffusion model is learned in conjunction with the encoder/decoder, which actively regularizes the latent space. This results in an improved latent structure, as detailed in §3.