Aligning Latent Spaces with Flow Priors

Yizhuo Li[1,2], Yuying Ge[2,✉], Yixiao Ge[2], Ying Shan[2], Ping Luo[1,✉]
[1]The University of Hong Kong, [2]ARC Lab, Tencent PCG

Project Page: `https://liyizhuo.com/align/`

Abstract

This paper presents a novel framework for aligning learnable latent spaces to arbitrary target distributions by leveraging flow-based generative models as priors. Our method first pretrains a flow model on the target features to capture the underlying distribution. This fixed flow model subsequently regularizes the latent space via an alignment loss, which reformulates the flow matching objective to treat the latents as optimization targets. We formally prove that minimizing this alignment loss establishes a computationally tractable surrogate objective for maximizing a variational lower bound on the log-likelihood of latents under the target distribution. Notably, the proposed method eliminates computationally expensive likelihood evaluations and avoids ODE solving during optimization. As a proof of concept, we demonstrate in a controlled setting that the alignment loss landscape closely approximates the negative log-likelihood of the target distribution. We further validate the effectiveness of our approach through large-scale image generation experiments on ImageNet with diverse target distributions, accompanied by detailed discussions and ablation studies. With both theoretical and empirical validation, our framework paves a new way for latent space alignment.

1 Introduction

Latent models like autoencoders (AEs) are a cornerstone of modern machine learning [24, 3, 36, 8, 45]. These models typically map high-dimensional observations to a lower-dimensional latent space, aiming to capture salient features and dependencies [40, 44]. A highly desirable property of latent models is that the latent space should have structural properties, such as being close to a predefined target distribution [53, 33, 63, 5]. Such structure can incorporate domain-specific prior knowledge [28, 51], enhance the interpretability of the latent space[23, 9, 30], and facilitate latent space generation [52, 37, 35, 62, 64]. While significant progress has been made, ensuring that the learned latent representations possess such desired structure remains a crucial challenge.

Traditional approaches to enforcing distributional conformity often involve minimizing divergences like the Kullback-Leibler (KL) divergence [33, 52]. However, KL can be restrictive, particularly when the target prior is only implicitly defined (e.g., by samples). In latent generative modeling, the latent space is usually regularized with known prior distributions, such as the Gaussian distribution for Variational Autoencoders (VAE) [33, 18], and the categorical distribution for Vector Quantized VAE (VQ-VAE) [61]. Recent works [50, 38, 39, 4, 63, 5] have proposed to use pre-trained feature extractors as target distribution and directly optimize the latent distances, which are shown to be effective but computationally expensive and require per-sample features.

Recent advances in flow-based generative models [41, 42] offer a promising avenue to capture complex target distributions. In this work, we address the question: *Can we efficiently align a learnable latent space to an arbitrary target distribution using a pre-trained flow model as a prior?* We answer this question affirmatively by proposing a novel framework that leverages a pre-trained flow model to define a computationally tractable alignment loss, which effectively guides the latents towards the target distribution.

Preprint. Under review.

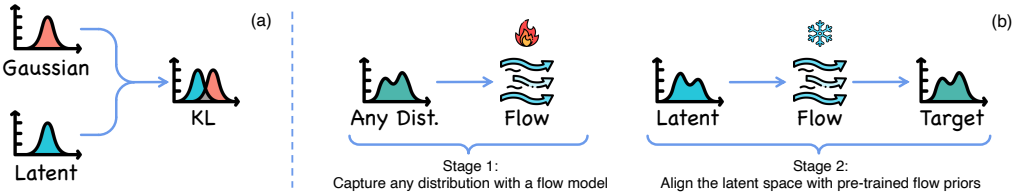


Figure 1: (a) Conventional alignment works with only known priors (e.g., Gaussian or categorical) using KL or cross-entropy losses. (b) Our proposed method can align the latent distribution to **arbitrary** target distribution captured by a pre-trained flow model.

Our proposed approach unfolds in a two-stage process as illustrated in Fig.1. The first stage involves pre-training a flow-based generative model on the desired target features, allowing it to learn the mapping from a base distribution (e.g., Gaussian) to the target distribution. Once this flow model accurately captures the target distribution, its parameters are fixed. In the second stage, this flow model serves as a prior to regularize a learnable latent space, for instance, the output of the encoder in an AE. This regularization is achieved by minimizing an alignment loss, which ingeniously adapts the standard flow matching objective by treating the learnable latents as the target. This pipeline provides an efficient mechanism to guide the latent space towards the desired target structure without requiring direct comparison to target samples or expensive likelihood evaluations of the flow model.

We theoretically justify our method by connecting the alignment loss to the maximum likelihood estimation of the latents under the target distribution. While directly maximizing this likelihood under a flow model is often computationally prohibitive due to the need to evaluate the trace of Jacobian determinants and solve an ordinary differential equation (ODE) for each step, our alignment loss offers a more tractable alternative. We formally demonstrate that minimizing this loss serves as a computationally efficient proxy for maximizing a variational lower bound on the log-likelihood of the latents under the flow-defined target distribution.

Our framework offers three key advantages. First, our approach enables alignment to **arbitrary target distributions**, even those implicitly defined by samples, overcoming the limitations of conventional methods that require explicit parametric priors. Second, the alignment loss acts as a **direct surrogate** for the log-likelihood of latents under the target distribution, providing a theoretically grounded objective that avoids heuristic metrics like cosine similarity used in per-sample feature matching [4, 63, 5]. Third, our framework is **computationally lightweight**, requiring only a single forward pass through the pre-trained flow model during training, thereby bypassing the need for expensive adversarial optimization [19], likelihood evaluations, or per-sample feature extraction [50, 38, 39].

We empirically validate the efficacy of our proposed alignment strategy through a series of experiments. We start with illustrative experiments in a controlled toy setting using a mixture of Gaussians to confirm that our alignment loss landscape indeed serves as a proxy for the log-likelihood of the latents under the target distribution. Then we demonstrate the scalability of our approach by conducting large-scale image generation experiments on ImageNet [11] with diverse target distributions. Detailed discussions and ablation studies are provided to underscore the robustness and effectiveness.

We believe this method offers a powerful and flexible tool for incorporating rich distributional priors into latent models. Our work paves the way for more flexible and powerful structured representation learning, and we anticipate its application and extension in various domains requiring distributional structure control over latent spaces.

2 Related Work

2.1 Flow-based Models

Flow-based generative models have emerged as a powerful class of generative models [17, 34, 7, 31, 66, 68, 55]. They were first introduced as CNFs [6, 20] that learn an invertible mapping between a simple base distribution (e.g., Gaussian) and a complex data distribution. Early works like NICE [13] and Real NVP [14] introduced additive and affine coupling layers to construct invertible neural networks. A notable recent development is Flow Matching (FM) [41, 1, 42, 46, 21, 59], which

simplifies the training by regressing a vector field against a target field derived from pairs of samples, avoiding the need for simulating ODEs during training. In ICTM [67], flow priors of generative models have been employed for MAP estimation to solve linear inverse problems. Our work leverages flow-based models to learn complex distributions as a prior for latent space alignment.

2.2 Latent Space Alignment

The alignment of latent spaces with predefined distributions is a crucial aspect of representation learning. In VAE [33], the latent space is typically regularized to follow a standard Gaussian distribution. Several approaches have been proposed to use more flexible priors, such as hierarchical VAEs [56, 60] or VAEs with inverse autoregressive flow (IAF) priors [32]. Another line of work focuses on aligning latent spaces with features extracted from pre-trained models [50, 38, 39, 4, 63, 5, 29, 65]. Our method differs by utilizing a pre-trained flow model to define an expressive target and a novel alignment loss, avoiding expensive likelihoods, adversarial training, or direct per-sample feature comparison.

3 Preliminaries

3.1 Flow-Based Models

We consider an ordinary differential equation (ODE) ideally defined by a time-dependent velocity field $\boldsymbol{u}(\boldsymbol{x}_t, t)$. The dynamics are given by:

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}_t, t), \quad \boldsymbol{x}_0 \sim p_{\text{init}}, \quad \boldsymbol{x}_1 \sim p_{\text{data}} \tag{1}$$

Here, p_{init} is a simple prior distribution (e.g., a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$), and p_{data} is the target data distribution. We denote $\boldsymbol{x}_t \in \mathbb{R}^d$ as the state at time t, with $\boldsymbol{x}_0$ being the initial state and $\boldsymbol{x}_1$ the state at $t = 1$. The velocity field $\boldsymbol{u} : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is assumed to be Lipschitz continuous in $\boldsymbol{x}$ and continuous in t to ensure the existence and uniqueness of ODE solutions.

In practice, the ideal velocity field $\boldsymbol{u}$ is unknown. We approximate it with a parametric model, typically a neural network $\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$, parameterized by θ. This defines a learned generative process:

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t), \quad \boldsymbol{x}_0 \sim p_{\text{init}} \tag{2}$$

For a given initial condition $\boldsymbol{x}_0$, the solution to this ODE, denoted by $\boldsymbol{x}_t = \boldsymbol{\Phi}_t^\theta(\boldsymbol{x}_0)$, is a trajectory (or flow) evolving from $\boldsymbol{x}_0$. The aim is to train $\boldsymbol{v}_\theta$ such that the $\boldsymbol{x}_1 = \boldsymbol{\Phi}_1^\theta(\boldsymbol{x}_0)$ matches p_{data}.

Flow matching techniques [41, 42] train $\boldsymbol{v}_\theta$ by minimizing its difference from a target velocity field. This target field is often defined by constructing a probability path $p_t(\boldsymbol{x})$ that interpolates between p_{init} and p_{data}. A common choice is a conditional path $\boldsymbol{x}_t(\boldsymbol{x}_0, \boldsymbol{x}_1)$ defined for pairs $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ sampled from $p_{\text{init}} \times p_{\text{data}}$. For instance, Rectified Flow uses a linear interpolation: $\boldsymbol{x}_t(\boldsymbol{x}_0, \boldsymbol{x}_1) = (1 - t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$. The target velocity field corresponding to this path is $\boldsymbol{u}_t(\boldsymbol{x}_t|\boldsymbol{x}_0, \boldsymbol{x}_1) = \boldsymbol{x}_1 - \boldsymbol{x}_0$. The neural network $\boldsymbol{v}_\theta$ is then trained to predict this target field by minimizing the flow matching loss:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathop{\mathbb{E}}_{t\sim\mathcal{U}[0,1], \boldsymbol{x}_0\sim p_{\text{init}}, \boldsymbol{x}_1\sim p_{\text{data}}} \left[\|\boldsymbol{v}_\theta((1 - t)\boldsymbol{x}_0 + t\boldsymbol{x}_1, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2\right] \tag{3}$$

In this paper, we consider $\boldsymbol{v}_\theta$ to be pre-trained, fixed, and optimal. That is, $\boldsymbol{v}_\theta$ is assumed to have perfectly minimized Eq. (3), such that $\boldsymbol{v}_\theta((1-t)\boldsymbol{x}_0+t\boldsymbol{x}_1, t) = \boldsymbol{x}_1-\boldsymbol{x}_0$ for all $\boldsymbol{x}_0 \sim p_{\text{init}}$, $\boldsymbol{x}_1 \sim p_{\text{data}}$, and $t \in [0, 1]$. Such a $\boldsymbol{v}_\theta$ can serve as a regularizer to align latents to the target distribution.

3.2 Likelihood Estimation with Flow Priors

Let $p_1^{\boldsymbol{v}_\theta}(\boldsymbol{x}_1)$ denote the probability density at $t = 1$ induced by the flow model $\boldsymbol{v}_\theta$ evolving from p_{init}. Using the instantaneous change of variables formula, the log-likelihood of a sample $\boldsymbol{x}_1$ under this model can be computed by [6, 20]:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{x}_1) = \log p_{\text{init}}(\boldsymbol{x}_0) - \int_0^1 \mathrm{Tr}(\nabla_{\boldsymbol{x}} \boldsymbol{v}_\theta(\boldsymbol{x}_s, s))\mathrm{d}s \tag{4}$$

Here, $\boldsymbol{x}_s = \boldsymbol{\Phi}_s^\theta(\boldsymbol{x}_0)$ is the trajectory generated by $\boldsymbol{v}_\theta$ starting from $\boldsymbol{x}_0$ and ending at $\boldsymbol{x}_1 = \boldsymbol{\Phi}_1^\theta(\boldsymbol{x}_0)$. Thus, $\boldsymbol{x}_0 = (\boldsymbol{\Phi}_1^\theta)^{-1}(\boldsymbol{x}_1)$ is obtained by flowing $\boldsymbol{x}_1$ backward in time to $t = 0$. Given a pre-trained flow model $\boldsymbol{v}_\theta$ that maps p_{init} (e.g., Gaussian noise) to a target distribution (e.g., target features), one can align new input samples with these target features by maximizing their log-likelihood under $p_1^{\boldsymbol{v}_\theta}$. However, computing Eq. (4) is often computationally expensive, primarily due to the trace of the Jacobian term ($\text{Tr}(\nabla_{\boldsymbol{x}} \boldsymbol{v}_\theta)$) and the need for an ODE solver. In this paper, we demonstrate that a similar alignment objective can be achieved by minimizing the flow matching loss Eq. (3) with respect to $\boldsymbol{x}_1$, treating $\boldsymbol{x}_1$ as a variable to be optimized rather than a fixed sample from p_{data}.

4 Method

In this paper, we aim to align a learnable latent space, whose latents are denoted by $\boldsymbol{y}$, to a target distribution p_{data}. We first describe the overall pipeline in Sec. 4.1. Our method leverages a pre-trained flow model to implicitly capture p_{data} and subsequently align the latents $\boldsymbol{y}$. Then, we provide an intuitive explanation in Sec. 4.2 and a formal proof of the proposed method in Sec. 4.3.

4.1 Pipeline

Let $\boldsymbol{y} \in \mathbb{R}^{d_1}$ denote a sample from a learnable latent space. These latents $\boldsymbol{y}$ are typically produced by a parametric model G_ϕ (e.g., the encoder of an AE), whose parameters ϕ we aim to train. Let $\boldsymbol{x} \in \mathbb{R}^{d_2}$ be a sample from the target feature space, characterized by an underlying distribution $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$. Our objective is to train G_ϕ such that the distribution of its outputs, $p_\phi(\boldsymbol{y})$, aligns to $p_{\text{data}}(\boldsymbol{x})$. This alignment can be formulated as maximizing the likelihood of $\boldsymbol{y}$ under p_{data}. For instance, in an AE setting where we wish the latent space (from which $\boldsymbol{y}$ is sampled) to conform to the distribution of features from a pre-trained feature extractor (from which $\boldsymbol{x}$ is sampled), our method facilitates this alignment.

Addressing the Dimension Mismatch A challenge arises if the latent space dimension d_1 differs from the target feature space dimension d_2. To address this, we employ fixed (non-learnable) linear projections to map target features $\boldsymbol{x}$ from $\mathbb{R}^{d_2}$ to $\mathbb{R}^{d_1}$. We still keep the notation for the projected features and their distribution as $\boldsymbol{x}$ and p_{data} respectively for simplicity. We consider three alternative projection operators: *Random Projection*, *Average Pooling*, and *PCA*. We ablate these methods in Sec. 5.3 and select random projection as the default due to its simplicity and empirical effectiveness.

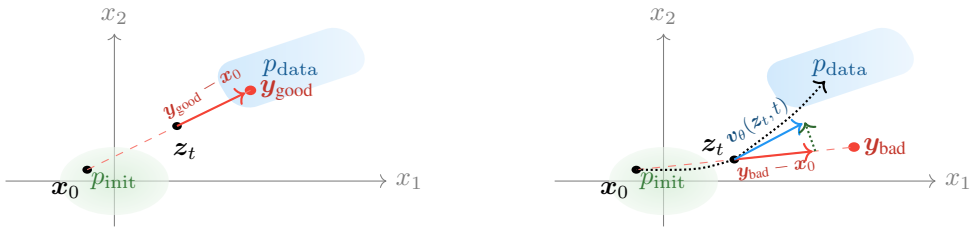
The use of linear projection is theoretically supported by the Johnson-Lindenstrauss (JL) lemma [26]. The JL lemma states that for a set of N points in $\mathbb{R}^{d_2}$, a random linear mapping can preserve all pairwise Euclidean distances within a multiplicative distortion factor. The linear projection is defined by a matrix $\boldsymbol{W} \in \mathbb{R}^{d_1 \times d_2}$. Assuming the target features $\boldsymbol{x}$ are appropriately normalized, we initialize the projection matrix $\boldsymbol{W}$ by sampling its entries from $\mathcal{N}(0, 1/d_2)$. This scaling helps ensure that the components of $\boldsymbol{x}$ also have approximately unit variance if the components of $\boldsymbol{x}$ are uncorrelated, thereby preserving key statistical properties.

Flow Prior Estimation With the projected target features $\boldsymbol{x} \sim p_{\text{data}}$, we first train a flow model $\boldsymbol{v}_\theta : \mathbb{R}^{d_1} \times [0, 1] \to \mathbb{R}^{d_1}$, parameterized by θ. This model is trained using the flow matching objective Eq. (3), where the dimension d is set to d_1, $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\boldsymbol{x}_1$ is replaced by samples $\boldsymbol{x}$ from p_{data}. After training, the parameters θ of the flow model $\boldsymbol{v}_\theta$ are frozen. This fixed $\boldsymbol{v}_\theta$ implicitly defines a generative process capable of transforming samples from p_{init} (now in $\mathbb{R}^{d_1}$) into samples that approximate p_{data}. It captures the underlying target distribution and serves as a distributional prior for aligning the latent space.

Latent Space Regularization Once $\boldsymbol{v}_\theta$ is trained and its parameters fixed, we use it to regularize the learnable latents $\boldsymbol{y}$. The goal is to encourage the distribution $p_\phi(\boldsymbol{y})$ to conform to p_{data} as captured by $\boldsymbol{v}_\theta$. For each $\boldsymbol{y}$ produced by G_ϕ, we incorporate the flow matching objective described in Eq. (3) into the training objective of G_ϕ:

$$\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta) = \mathop{\mathbb{E}}_{t \sim \mathcal{U}[0,1], \boldsymbol{x}_0 \sim p_{\text{init}}(\boldsymbol{x}_0)} \left[\| \boldsymbol{v}_\theta((1-t)\boldsymbol{x}_0 + t\boldsymbol{y}, t) - (\boldsymbol{y} - \boldsymbol{x}_0) \|^2 \right] \tag{5}$$

Here, p_{init} is the same d_1-dimensional base distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ used for training $\boldsymbol{v}_\theta$. In Sec. 4.3, we formally prove that minimizing Eq. (5) with respect to $\boldsymbol{y}$ serves as a proxy to maximizing a lower



(a) "Good" case (b) "Bad" case

Figure 2: Intuitive illustration of latent space alignment via flow matching, best viewed in color. (a) A "good" $\boldsymbol{y}_{\text{good}}$ in p_{data} (green) aligns the straight path velocity (red solid arrow) with the pre-trained flow model's velocity $\boldsymbol{v}_\theta(\boldsymbol{z}_t, t)$ (overlapped and omitted), yielding low loss. (b) A "bad" $\boldsymbol{y}_{\text{bad}}$ outside p_{data} causes a mismatch between the path velocity and $\boldsymbol{v}_\theta(\boldsymbol{z}_t, t)$ (green solid arrow), resulting in high loss. Minimizing this loss steers y_{bad} to p_{data} (blud dotted arrow).

bound on the log-likelihood $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$. This establishes that minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ effectively trains G_ϕ such that its outputs $\boldsymbol{y}$ align with the distribution of the target features $\boldsymbol{x}$.

The key insight is that the pre-trained velocity field $\boldsymbol{v}_\theta$ encapsulates the dynamics that transport probability mass from the base distribution p_{init} to the target distribution p_{data} along linear paths. By minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$, we penalize latents $\boldsymbol{y}$ that do not conform to these learned dynamics—that is, $\boldsymbol{y}$ values for which the path $(1-t)\boldsymbol{x}_0 + t\boldsymbol{y}$ is not "natural" under $\boldsymbol{v}_\theta$. This procedure shapes $p_\phi(\boldsymbol{y})$ to match p_{data} without requiring explicit computation of potentially intractable likelihoods, relying instead on the computationally efficient flow matching objective.

4.2 Intuitive Explanation

Our alignment method leverages the pre-trained flow model, $\boldsymbol{v}_\theta$, as an expert on the target feature distribution p_{data}. Having been well trained, $\boldsymbol{v}_\theta$ precisely captures the dynamics required to transform initial noise samples $\boldsymbol{x}_0$ into target features $\boldsymbol{x}$ along straight interpolation paths. Specifically, it has learned to predict the exact velocity $\boldsymbol{x} - \boldsymbol{x}_0$ at any point $(1-t)\boldsymbol{x}_0 + t\boldsymbol{x}$ along such a path. This effectively means $\boldsymbol{v}_\theta$ can validate whether a given trajectory from noise is characteristic of those leading to the true target distribution.

We utilize this knowledge to shape the distribution of our learnable latents $\boldsymbol{y}$. The alignment loss, $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$, challenges $\boldsymbol{v}_\theta$: for a given $\boldsymbol{y}$ and a random $\boldsymbol{x}_0$, it asks whether the velocity field predicted by $\boldsymbol{v}_\theta$ along the straight path $(1-t)\boldsymbol{x}_0 + t\boldsymbol{y}$ matches the path's inherent velocity, $\boldsymbol{y} - \boldsymbol{x}_0$. If $\boldsymbol{y}$ is statistically similar to samples from p_{data}, this match will be close, resulting in a low loss. Conversely, a significant mismatch indicates that $\boldsymbol{y}$ is not a plausible target according to the learned dynamics, yielding a high loss. By minimizing this loss (by optimizing the generator G_ϕ that produces $\boldsymbol{y}$), we iteratively guide $\boldsymbol{y}$ towards regions where its connecting path from noise is endorsed by $\boldsymbol{v}_\theta$. As depicted in Fig. 2, this process progressively aligns the distribution of $\boldsymbol{y}$ (blue) with the target distribution p_{data} (orange), achieving distributional conformity.

4.3 Relating the Alignment Loss to an ELBO on Log-Likelihood

In this section, we demonstrate that minimizing the alignment loss $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ (Eq. (5)) with respect to a given $\boldsymbol{y} \in \mathbb{R}^{d_1}$ corresponds to maximizing a variational lower bound (ELBO) on the log-likelihood $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$. Here, $p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$ denotes the probability density at $t = 1$ induced by the ODE dynamics $\frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} = \boldsymbol{v}_\theta(\boldsymbol{z}_t, t)$, with $\boldsymbol{z}_0 \sim p_{\text{init}}$.

Proposition 1. *Let $\boldsymbol{v}_\theta : \mathbb{R}^{d_1} \times [0,1] \to \mathbb{R}^{d_1}$ be a given velocity field, and p_{init} be a base distribution. For $\boldsymbol{y} \in \mathbb{R}^{d_1}$, the log-likelihood $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$ is lower-bounded as:*

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq C(\boldsymbol{y}) - \lambda \mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta), \tag{6}$$

where $\lambda > 0$ is a constant, $\mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta)$ is defined in Eq. (5), and $C(\boldsymbol{y})$ is dependent on $\boldsymbol{y}$ and $\boldsymbol{v}_\theta$.

Proof. We establish this result by constructing a specific variational lower bound on $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$. Variational lower bounds for log-likelihoods in continuous-time generative models can be constructed

by introducing a proposal distribution for the latents that could generate $\boldsymbol{y}$. Consider a family of "proposal" paths [41], which are straight lines interpolating from an initial point $\boldsymbol{x}_0 \sim p_{\text{init}}$ to the given point $\boldsymbol{y}$:

$$\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}) = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, \quad s \in [0,1] \tag{7}$$

The velocity of such a path is constant: $\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{x}_0$. We adopt a variational distribution over the initial states $\boldsymbol{x}_0$, conditioned on $\boldsymbol{y}$, as $q(\boldsymbol{x}_0|\boldsymbol{y}) = p_{\text{init}}(\boldsymbol{x}_0)$. That is, we consider initial states drawn from the prior, irrespective of $\boldsymbol{y}$ for the functional form of q.

A known variational lower bound on $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$s [6, 20, 42] can be written a:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq \mathbb{E}_{\boldsymbol{x}_0 \sim q(\cdot|\boldsymbol{y})} \Bigg[\log p_{\text{init}}(\boldsymbol{x}_0) - \int_0^1 \left(\lambda_s \|\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y}) - \boldsymbol{v}_\theta(\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}), s)\|^2 \right) \mathrm{d}s \\ - \log q(\boldsymbol{x}_0|\boldsymbol{y}) - \int_0^1 \left(\mathrm{Tr}(\nabla_{\boldsymbol{z}_s} \boldsymbol{v}_\theta(\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}), s)) \right) \mathrm{d}s \Bigg] \tag{8}$$

Here, $\lambda_s > 0$ is a time-dependent weighting factor. For simplicity and consistency with the definition of $\mathcal{L}_{\text{align}}$ (Eq. (5)), we set $\lambda_s = \lambda = 1$ for all $s \in [0,1]$. With $q(\boldsymbol{x}_0|\boldsymbol{y}) = p_{\text{init}}(\boldsymbol{x}_0)$, the term $\log p_{\text{init}}(\boldsymbol{x}_0) - \log q(\boldsymbol{x}_0|\boldsymbol{y})$ vanishes. Substituting the expressions for $\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y})$ from Eq. (7) and its velocity $\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{x}_0$:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq - \mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{init}}} \left[\int_0^1 \mathrm{Tr}(\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)) \mathrm{d}s \right] \\ - \mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{init}}} \left[\int_0^1 \|(\boldsymbol{y} - \boldsymbol{x}_0) - \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)\|^2 \mathrm{d}s \right] \tag{9}$$

The second term in this inequality matches the definition of $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ (Eq. (5)). Let us define the first term of the ELBO's right-hand side. To maintain consistency with the expectation over time in $\mathcal{L}_{\text{align}}$, we can write:

$$C(\boldsymbol{y}) = -\mathbb{E}_{s \sim \mathcal{U}[0,1], \boldsymbol{x}_0 \sim p_{\text{init}}} \left[\mathrm{Tr}(\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)) \right] \tag{10}$$

So, the ELBO (Eq. (9)) can be expressed as:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq C(\boldsymbol{y}) - \mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta) \tag{11}$$

This concludes the derivation of the lower bound as stated in the proposition (with $\lambda = 1$). □

Interpretation and Significance The inequality (11) demonstrates that maximizing the derived lower bound with respect to $\boldsymbol{y}$ involves two parts: maximizing $C(\boldsymbol{y})$ and minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$. The term $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ directly measures how well the velocity field $\boldsymbol{v}_\theta$ predicts the velocity of that straight path, i.e., $\boldsymbol{y} - \boldsymbol{x}_0$. Minimizing this term forces $\boldsymbol{y}$ into regions where it behaves like a point reachable from p_{init} via a straight path whose dynamics are consistent with the learned $\boldsymbol{v}_\theta$. This is precisely the behavior expected if $\boldsymbol{y}$ were a sample from the distribution p_{data}.

The term $C(\boldsymbol{y})$ represents the expected negative trace of the Jacobian of $\boldsymbol{v}_\theta$, averaged over the chosen straight variational paths. By minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$, we are not strictly maximizing the ELBO in Eq. (11) with respect to $\boldsymbol{y}$. Instead, we are optimizing a crucial component of it that directly enforces consistency with the learned flow dynamics. We analyze the behavior of $C(\boldsymbol{y})$ in Appendix A to show that if $\boldsymbol{y}$ aligns with a more concentrated target distribution (making $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ small), $C(\boldsymbol{y})$ tends to be positive and larger, contributing favorably to the ELBO.

Assumption 1 (Optimality of $\boldsymbol{v}_\theta$)**.** *The velocity field $\boldsymbol{v}_\theta : \mathbb{R}^{d_1} \times [0,1] \to \mathbb{R}^{d_1}$ is (pre-trained) and optimal, satisfying $\boldsymbol{v}_\theta((1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1, t) = \boldsymbol{x}_1 - \boldsymbol{x}_0 \quad \forall \boldsymbol{x}_0 \sim p_{\text{init}}, \boldsymbol{x}_1 \sim p_{\text{data}}, t \in [0,1]$.*

To further interpret the method, we consider the Assumption 1 that $\boldsymbol{v}_\theta$ is optimally trained such that $\boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1, s) = \boldsymbol{x}_1 - \boldsymbol{x}_0$ for $\boldsymbol{x}_1 \sim p_{\text{data}}$. If $\boldsymbol{y}$ is itself a sample from p_{data}, then $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ would be (close to) zero. However, when optimizing an arbitrary $\boldsymbol{y}$, especially if it is far from p_{data}, the $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ term can be substantial. Its minimization drives $\boldsymbol{y}$ towards regions of higher plausibility under the learned flow.

In practice, directly maximizing $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$ via Eq. (4) is computationally demanding, requiring ODE solves and computation of Jacobian traces along these true ODE paths. Maximizing the full ELBO

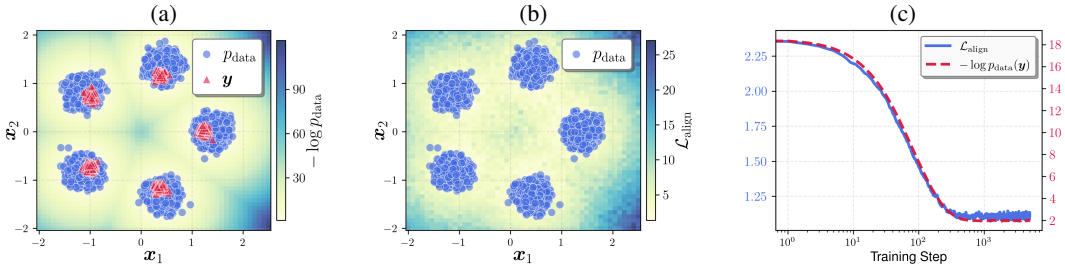


Figure 3: Illustration with a Mixture of Gaussians distribution. (a) Aligned latent variables $\boldsymbol{y}$ (red triangles) concentrate in low negative log-likelihood (NLL) regions of p_{data} (blue dots; heatmap shows $-\log p_{\text{data}}$). (b) Alignment loss $\mathcal{L}_{\text{align}}$ heatmap mirrors the NLL landscape of p_{data}, with p_{data} samples in low-$\mathcal{L}_{\text{align}}$ areas. (c) $\mathcal{L}_{\text{align}}$ (blue solid) and $-\log p_{\text{data}}(\boldsymbol{y})$ (red dashed) decline simultaneously in training, showing $\mathcal{L}_{\text{align}}$ serves as a proxy for maximizing the log-likelihood of $\boldsymbol{y}$ under p_{data}.

(Eq. (6)) would still require computing $C(\boldsymbol{y})$, which involves trace computations. By focusing on minimizing only $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$, we adopt a computationally tractable proxy. This objective encourages $\boldsymbol{y}$ to have a high likelihood under $p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$ by ensuring consistency with the learned flow dynamics, thereby aligning the distribution of $\boldsymbol{y}$ with the target distribution p_{data} implicitly modeled by $\boldsymbol{v}_\theta$. A more complete proof can be found in Appendix A.

5 Experiments

This section presents an empirical validation of the proposed alignment method with flow priors. The investigation starts with an illustrative experiment in Sec. 5.1. Subsequently, large-scale experiments are conducted on image generation tasks using the ImageNet dataset, as detailed in Sec. 5.2. In Sec. 5.3, we conduct ablation studies of the proposed method.

5.1 Toy Examples

We present a toy example as an illustrative experiment in a 2D setting. The target distribution, denoted p_{data}, is configured as a mixture of five isotropic Gaussians. Following the methodology outlined in Sec. 4.1, we first train a flow model $\boldsymbol{v}_\theta$ to map a standard 2D Normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to p_{data}. This flow model $\boldsymbol{v}_\theta$ is implemented by a multi-layer perceptron (MLP) incorporating adaptive layer normalization for time modulation [49]. Upon completion of training, the parameters θ of this flow model are frozen. Subsequently, instead of a parameterized model G_ϕ, we directly initialize a set of learnable 2D variables as $\boldsymbol{y}$ and optimize them by minimizing the alignment loss $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$.

The results are presented in Fig. 3. Fig. 3 (a) compares the target distribution p_{data} (blue point samples) with the optimized variables $\boldsymbol{y}$ (red triangles). The background visualizes the negative log-likelihood (NLL) of p_{data}, which is computed analytically. It is evident that $\boldsymbol{y}$ successfully converges to the high log-likelihood regions of p_{data}. Fig. 3 (b) displays the landscape of the alignment loss $\mathcal{L}_{\text{align}}$, which is estimated numerically with $\boldsymbol{v}_\theta$. The landscape mirrors the NLL surface of p_{data} depicted in (a). Samples drawn from p_{data} (blue dots) are concentrated in regions where $\mathcal{L}_{\text{align}}$ is low, suggesting that $\mathcal{L}_{\text{align}}$ effectively captures the underlying structure of the target distribution. Fig. 3 (c) illustrates $\mathcal{L}_{\text{align}}$ (blue solid line) and the true NLL $-\log p_{\text{data}}(\boldsymbol{y})$ (red dashed line) during the training of $\boldsymbol{y}$. The alignment loss and the NLL exhibit a strong positive correlation, decreasing concomitantly throughout the training process. More detailed toy examples can be found in Appendix B.

5.2 Image Generation

Prior work has demonstrated that aligning the latent space of AEs with semantic encoders can enhance generative model performance [4, 63, 5, 50]. To validate this observation and further showcase the efficacy of our proposed method, we conduct large-scale image generation experiments on the ImageNet-1K [11] dataset at 256×256 resolution.

Implementation Details Our AE architecture employs two Vision Transformer (ViT)-Large [15] models, each with 391M parameters, serving as the latent encoder and decoder, respectively. The

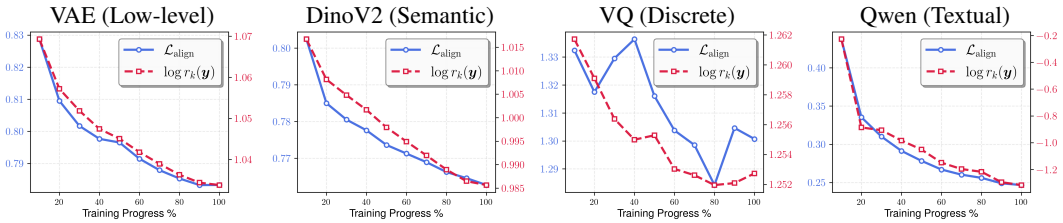


Figure 4: Aligning autoencoders on ImageNet-1K with different target distributions. The alignment loss $\mathcal{L}_{\text{align}}$ (blue solid) and the k-NN distance $\log r_k(\boldsymbol{y})$ (red dashed) are proportional throughout the training. Confirming that $\mathcal{L}_{\text{align}}$ serves as a good proxy for the NLL of the latents under p_{data}.

encoder maps input images to a latent space of 64 tokens, each with dimension 32, striking a balance between reconstruction quality and computational efficiency. We impose *token-level* alignment on the latents. The alignment loss on the latents is set to $\lambda = 0.01$ by default. We also incorporate conventional reconstruction loss, perceptual loss, and adversarial loss on the pixel outputs [18, 52].

For the target distribution p_{data}, we investigate four distinct variants: *low-level* visual features from a VAE [33, 18], continuous *semantic* visual features from DinoV2 [48], *discrete* visual codebook embeddings from LlamaGen VQ [58, 61], and *textual* embeddings from Qwen [2]. Their feature dimensions are 32, 768, 8, 896, respectively. The flow-based prior is modeled by a 6-layer MLP with 1024 hidden units, trained for 1 million steps using the AdamW [43] optimizer to match Assumption 1. More details can be found in Appendix C.

Alignment Results Analogous to the toy example, we aim to correlate the alignment loss $\mathcal{L}_{\text{align}}$ with the NLL of latents under the target distribution p_{data}. Since the NLL is intractable for implicitly defined distributions, we estimate the density using k-nearest neighbors. The probability density $p(\boldsymbol{y})$ at a point $\boldsymbol{y}$ is inversely proportional to the volume of the hypersphere enclosing its k^{th} nearest neighbor among the target samples. Consequently, the NLL can be estimated as $-\log p_{\text{data}}(\boldsymbol{y}) \propto \log r_k(\boldsymbol{y})$ where $r_k(\boldsymbol{y})$ is the Euclidean distance to the k^{th} neighbor, and D is the dimension. We use $\log r_k(\boldsymbol{y})$ as our proxy measure for the NLL. We first index the set of target distribution samples using Faiss [16]. During the training, we periodically sample 10k points from the latent space and measure the alignment quality by averaging the $\log r_k(\boldsymbol{y})$.

The results are presented in Fig. 4. A strong correlation is observed between the alignment loss $\mathcal{L}_{\text{align}}$ and the k-NN distance proxy $\log r_k(\boldsymbol{y})$. The only unstable case is the VQ variant, for which the GAN loss collapses during training due to its low dimension (8-dim), yet the general trend is still consistent. This finding corroborates our conclusion that $\mathcal{L}_{\text{align}}$ serves as an effective proxy for the NLL of the latents under p_{data}. Crucially, our method captures the underlying structure across diverse target distributions, spanning different forms (continuous, discrete) and modalities (visual, textual), even when applied to a large-scale dataset like ImageNet and a high-capacity model such as ViT-Large.

Generation Results After demonstrating effective latent space alignment, we investigated its impact on generative model performance. We evaluated both reconstruction and generation capabilities on ImageNet using the MAR-B [37] architecture. For MAR-B, we incorporated qk-norm [10] and replaced the diffusion head with a flow head to ensure stable training. We choose flow-based MAR-B as it does not favor continuous Gaussian-like latent structure like Diffusion models [57, 12, 27, 47, 52] do. To ensure an "apple-to-apple" comparison, configurations and hardware remained identical across all experiments, with the only difference being the specific AE used for each alignment variant.

The results are presented in Tab. 1. Reconstruction performance was measured by rFID [22] and PSNR on the ImageNet validation set. Generation performance was assessed using FID, IS [54], Precision, and Recall on 50k generated samples and the validation set, both with and without classifier-free guidance (CFG) [25]. Our key findings are:

1) Alignment vs. Reconstruction Trade-off: Latent space alignment typically degrades reconstruction quality (rFID, PSNR) compared to vanilla AEs, as constraints reduce capacity. SoftVQ[5] excels among aligned methods due to its sample-level alignment. *2) Alignment Enhances Generation:* Structured latent spaces improve generative metrics (FID, IS), but complexity is not decisive. Simple features (text embeddings like Qwen) may match the performance of richer visual features (DinoV2).

Table 1: ImageNet 256×256 conditional generation using MAR-B. All models are trained and evaluated using identical settings. The CFG scale is tuned for KL and kept the same for others.

Autoencoder	rFID↓	PSNR↑	w/o CFG				w/ CFG			
			FID↓	IS↑	Pre.↑	Rec.↑	FID↓	IS↑	Pre.↑	Rec.↑
AE	1.13	20.20	15.08	86.37	**0.60**	**0.59**	5.26	237.60	0.56	0.65
KL	1.65	22.59	12.94	91.86	0.60	0.58	5.29	200.85	0.57	0.65
SoftVQ	**0.61**	23.00	13.30	93.40	0.60	0.57	6.09	198.53	**0.58**	0.61
Low-level (VAE)	1.22	22.31	12.04	98.66	0.56	0.57	5.02	240.03	0.56	0.62
Semantic (Dino)	1.26	23.07	**11.47**	101.74	0.59	0.59	**4.87**	**250.38**	0.54	0.67
Discrete (VQ)	2.99	22.32	24.63	48.17	0.55	0.53	10.04	119.64	0.47	0.65
Textual (Qwen)	0.85	**23.12**	11.89	**102.23**	0.55	0.57	6.56	262.89	0.49	**0.69**

Table 2: Ablation studies on ImageNet 256×256 for different configurations using autencoders regularized by textual features (Qwen). We use a shorter training schedule when ablating weight.

(a) Downsampling Methods

Method	rFID↓	PSNR↑	FID↓	IS↑
Random Proj.	**0.85**	23.12	**11.89**	**102.23**
Avg. Pooling	0.94	22.98	16.06	60.37
PCA	0.87	**23.14**	14.95	83.59

(b) Alignment Loss Weight

Weight λ	rFID↓	PSNR↑	FID↓	IS↑
0.001	**0.89**	22.78	17.57	75.20
0.005	1.02	22.98	16.93	78.01
0.01	1.31	**23.12**	13.67	82.13
0.05	1.81	21.82	**12.30**	**92.48**

3) Optimal prior selection is open: No consensus exists on optimal priors. Low-dimensional discrete features (LlamaGen VQ) underperform, while cross-modal alignment (Qwen text embeddings) demonstrates transferable structural benefits. More discussions can be found in Appendix D.

5.3 Ablation Study

Downsampling Operators We ablate the downsampling operators in Tab. 2 (a). We adopt the same settings as in Tab. 1 using the model with the textual embeddings (Qwen) as the target distribution. Despite all being linear downsampling operators, PCA and Avg. Pooling perform worse than Random Projection. We hypothesize that this is because unlike Random Projection which preserves the structure of the data, both PCA and Avg. Pooling are likely to destroy the structure. Avg. Pooling performs especially poorly since it merges close features that are independent from the location.

Alignment Loss Weight We apply different strengths of regularization by altering the alignment loss weight λ in Tab. 2 (b). As expected, larger weight implies heavier regularization, worse reconstruction, and easier generation. However, heavier regularization limits the generation performance and may even cause the GAN loss to collapse. A trade-off exists between reconstruction and generation when the capacity of the model is limited.

6 Conclusion

This paper introduced a novel method for aligning learnable latent spaces with arbitrary target distributions by leveraging pre-trained flow-based generative models as expressive priors. Our approach utilizes a computationally tractable alignment loss, adapted from the flow matching objective, to guide latent variables towards the target distribution. We theoretically established that minimizing this alignment loss serves as a proxy for maximizing a variational lower bound on the log-likelihood of the latents under the flow-defined target. The effectiveness of our method is validated through empirical results, including controlled toy settings and large-scale ImageNet experiments. Ultimately, this work provides a flexible and powerful framework for incorporating rich distributional priors, paving the way for more structured and interpretable representation learning. A *limitation*, and also a promising future direction, is that the selection of the optimal prior remains a challenge. While semantic priors are effective for image generation, we posit that no single "silver bullet" prior exists for all tasks; rather, the optimal choice is likely task-specific and need be further explored.

References

[1] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *ArXiv*, abs/2209.15571, 2022.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[4] Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. *arXiv preprint arXiv:2502.03444*, 2025.

[5] Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. Softvq-vae: Efficient 1-dimensional continuous tokenizer. *arXiv preprint arXiv:2412.10958*, 2024.

[6] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[7] Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, et al. Goku: Flow based video generative foundation models. *arXiv preprint arXiv:2502.04896*, 2025.

[8] Shuangshuang Chen and Wei Guo. Auto-encoders in deep learning—a review with new perspectives. *Mathematics*, 11(8):1777, 2023.

[9] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems*, 2016.

[10] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[13] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv: Learning*, 2014.

[14] Laurent Dinh, Jascha Narain Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2016.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

[16] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[17] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[18] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2020.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[20] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[21] Eric Heitz, Laurent Belcour, and Thomas Chambon. Iterative α-(de) blending: A minimalist deterministic diffusion model. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–8, 2023.

[22] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems*, 2017.

[23] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[24] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[25] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[26] William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[27] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *ArXiv*, abs/2206.00364, 2022.

[28] Ilyes Khemakhem, Diederik P. Kingma, and Aapo Hyvärinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[29] Dongwon Kim, Ju He, Qihang Yu, Chenglin Yang, Xiaohui Shen, Suha Kwak, and Liang-Chieh Chen. Democratizing text-to-image masked generative models with compact text-aware one-dimensional tokens. *arXiv preprint arXiv:2501.07730*, 2025.

[30] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, 2018.

[31] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *ArXiv*, abs/1807.03039, 2018.

[32] Diederik P. Kingma, Tim Salimans, and Max Welling. Improved variational inference with inverse autoregressive flow. *ArXiv*, abs/1606.04934, 2016.

[33] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[34] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[35] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025.

[36] Pengzhi Li, Yan Pei, and Jianqiang Li. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, 138:110176, 2023.

[37] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[38] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Bhiksha Raj, and Zhe Lin. Imagefolder: Autoregressive image generation with folded tokens. *arXiv preprint arXiv:2410.01756*, 2024.

[39] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Jindong Wang, Zhe Lin, and Bhiksha Raj. Xq-gan: An open-source image tokenization framework for autoregressive generation. *arXiv preprint arXiv:2412.01762*, 2024.

[40] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.

[41] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[42] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[44] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J Kennedy. Relational autoencoder for feature extraction. In *2017 International joint conference on neural networks (IJCNN)*, pages 364–371. IEEE, 2017.

[45] Ibomoiye Domor Mienye and Theo G Swart. Deep autoencoder neural networks: A comprehensive review and new perspectives. *Archives of computational methods in engineering*, pages 1–20, 2025.

[46] Kirill Neklyudov, Rob Brekelmans, Daniel Severo, and Alireza Makhzani. Action matching: Learning stochastic dynamics from samples. In *International conference on machine learning*, pages 25858–25889. PMLR, 2023.

[47] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *ArXiv*, abs/2102.09672, 2021.

[48] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[49] William S. Peebles and Saining Xie. Scalable diffusion models with transformers. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182, 2022.

[50] Kai Qiu, Xiang Li, Jason Kuen, Hao Chen, Xiaohao Xu, Jiuxiang Gu, Yinyi Luo, Bhiksha Raj, Zhe Lin, and Marios Savvides. Robust latent matters: Boosting image generation with sampling error synthesis. *arXiv preprint arXiv:2503.08354*, 2025.

[51] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

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

[53] Rifai Salah, P Vincent, X Muller, X Gloro, and Y Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. of the 28th International Conference on Machine Learning*, pages 833–840, 2011.

[54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[55] Inkyu Shin, Chenglin Yang, and Liang-Chieh Chen. Deeply supervised flow-based generative models. *arXiv preprint arXiv:2503.14494*, 2025.

[56] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Neural Information Processing Systems*, 2016.

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

[58] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[59] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

[60] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *ArXiv*, abs/2007.03898, 2020.

[61] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[62] Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. " principal components" enable a new language of images. *arXiv preprint arXiv:2503.08685*, 2025.

[63] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. *arXiv preprint arXiv:2501.01423*, 2025.

[64] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

[65] Kaiwen Zha, Lijun Yu, Alireza Fathi, David A Ross, Cordelia Schmid, Dina Katabi, and Xiuye Gu. Language-guided image tokenization for generation. *arXiv preprint arXiv:2412.05796*, 2024.

[66] Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Angel Bautista, Navdeep Jaitly, and Joshua M. Susskind. Normalizing flows are capable generative models. *ArXiv*, abs/2412.06329, 2024.

[67] Yasi Zhang, Peiyu Yu, Yaxuan Zhu, Yingshan Chang, Feng Gao, Yingnian Wu, and Oscar Leong. Flow priors for linear inverse problems via iterative corrupted trajectory matching. *ArXiv*, abs/2405.18816, 2024.

[68] Wenliang Zhao, Minglei Shi, Xumin Yu, Jie Zhou, and Jiwen Lu. Flowturbo: Towards real-time flow-based image generation with velocity refiner. *ArXiv*, abs/2409.18128, 2024.

A Complete Proof

A.1 Complete Proof for Proposition 1

We restate Proposition 1 for clarity and self-containedness.

Proposition 1. *Let $\boldsymbol{v}_\theta : \mathbb{R}^{d_1} \times [0,1] \to \mathbb{R}^{d_1}$ be a given velocity field, and p_{init} be a base distribution. For any $\boldsymbol{y} \in \mathbb{R}^{d_1}$, the log-likelihood $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$ of $\boldsymbol{y}$ under the distribution induced by flowing p_{init} with $\boldsymbol{v}_\theta$ from $t = 0$ to $t = 1$, is lower-bounded as:*

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq C(\boldsymbol{y}) - \lambda \mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta), \tag{12}$$

where $\lambda > 0$ is a constant, $\mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta)$ is defined as

$$\mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta) = \mathop{\mathbb{E}}_{s \sim \mathcal{U}[0,1], \boldsymbol{x}_0 \sim p_{\text{init}}(\boldsymbol{x}_0)} \left[\|(\boldsymbol{y} - \boldsymbol{x}_0) - \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)\|^2 \right], \tag{13}$$

and $C(\boldsymbol{y})$ is a term dependent on $\boldsymbol{y}$ and $\boldsymbol{v}_\theta$, given by

$$C(\boldsymbol{y}) = -\mathbb{E}_{s \sim \mathcal{U}[0,1], \boldsymbol{x}_0 \sim p_{\text{init}}} \left[\text{Tr}(\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)) \right]. \tag{14}$$

To prove this, we make the following assumptions:

Assumption 2 (Properties of the Velocity Field)**.** *The velocity field $\boldsymbol{v}_\theta : \mathbb{R}^{d_1} \times [0,1] \to \mathbb{R}^{d_1}$ is continuous in t and Lipschitz continuous in its spatial argument with continuously differentiable components, ensuring that $\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta(\boldsymbol{z}, s)$ exists and is bounded on compact sets.*

Assumption 3 (Variational Path Choice)**.** *The variational paths used to construct the ELBO are straight lines $\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}) = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}$ for $s \in [0,1]$, originating from $\boldsymbol{x}_0 \sim p_{\text{init}}$ and terminating at $\boldsymbol{y}$. The velocity of such a path is $\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{x}_0$.*

Assumption 4 (Variational Distribution Choice)**.** *The variational distribution over the initial states $\boldsymbol{x}_0$ conditioned on $\boldsymbol{y}$ is chosen as $q(\boldsymbol{x}_0|\boldsymbol{y}) = p_{\text{init}}(\boldsymbol{x}_0)$.*

Assumption 5 (Weighting Factor in ELBO)**.** *The time-dependent weighting factor λ_s in the general ELBO formulation (Eq. 15 below) is chosen as a positive constant $\lambda_s = \lambda > 0$ for all $s \in [0,1]$.*

Remark 1 (Optimality of $\boldsymbol{v}_\theta$)**.** *Sec. 4.3 introduces Assumption 1, which states that $\boldsymbol{v}_\theta$ is optimally pre-trained such that $\boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1, s) = \boldsymbol{x}_1 - \boldsymbol{x}_0$ for $\boldsymbol{x}_1 \sim p_{\text{data}}$. This assumption is not required for the mathematical validity of the ELBO in Proposition 1 itself, which holds for any $\boldsymbol{v}_\theta$ satisfying Assumption 2. However, Assumption 1 is crucial for interpreting why minimizing $\mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta)$ drives $\boldsymbol{y}$ towards p_{data}, as it implies $\mathcal{L}_{\textit{align}}(\boldsymbol{y}; \theta) \approx 0$ if $\boldsymbol{y} \sim p_{\text{data}}$.*

Proof. The proof is based on a variational approach to lower-bound the log-likelihood in continuous-time generative models. This technique has been established in the literature for Neural ODEs and continuous normalizing flows [6, 20, 42].

For a target point $\boldsymbol{y}$, we consider a family of paths $\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y})$ parameterized by initial states $\boldsymbol{x}_0$ drawn from a proposal distribution $q(\boldsymbol{x}_0|\boldsymbol{y})$, where each path starts at $\boldsymbol{x}_0$ and ends at $\boldsymbol{y}$ (i.e., $\boldsymbol{z}_0(\boldsymbol{x}_0, \boldsymbol{y}) = \boldsymbol{x}_0$ and $\boldsymbol{z}_1(\boldsymbol{x}_0, \boldsymbol{y}) = \boldsymbol{y}$). The variational lower bound is derived by considering the path integral formulation of the likelihood. For any such family of paths with velocities $\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y})$, the bound takes the form:

$$\begin{aligned} \log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq \mathbb{E}_{\boldsymbol{x}_0 \sim q(\cdot|\boldsymbol{y})} \Bigg[& \log p_{\text{init}}(\boldsymbol{x}_0) - \log q(\boldsymbol{x}_0|\boldsymbol{y}) \\ & - \int_0^1 \lambda_s \|\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y}) - \boldsymbol{v}_\theta(\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}), s)\|^2 \mathrm{d}s \\ & - \int_0^1 \text{Tr}(\nabla_{\boldsymbol{z}_s} \boldsymbol{v}_\theta(\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}), s)) \mathrm{d}s \Bigg]. \end{aligned} \tag{15}$$

We now apply our specific assumptions:

- By Assumption 3, the paths are $\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}) = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}$, and their velocities are $\dot{\boldsymbol{z}}_s(\boldsymbol{x}_0, \boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{x}_0$.

- By Assumption 4, $q(\boldsymbol{x}_0|\boldsymbol{y}) = p_{\text{init}}(\boldsymbol{x}_0)$. This causes the term $\log p_{\text{init}}(\boldsymbol{x}_0) - \log q(\boldsymbol{x}_0|\boldsymbol{y})$ to vanish.
- By Assumption 5, we set $\lambda_s = \lambda$, a positive constant for all $s \in [0, 1]$.

Substituting these into Eq. 15:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq \mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{init}}} \Bigg[- \int_0^1 \lambda \|(\boldsymbol{y} - \boldsymbol{x}_0) - \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)\|^2 \mathrm{d}s \\ - \int_0^1 \mathrm{Tr}(\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)) \mathrm{d}s \Bigg]. \tag{16}$$

Using the equivalence $\int_0^1 f(s)\mathrm{d}s = \mathbb{E}_{s\sim\mathcal{U}[0,1]}[f(s)]$ for integrable functions f, we can rewrite each term. The first term becomes:

$$\mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{init}}} \left[-\lambda \int_0^1 \|(\boldsymbol{y} - \boldsymbol{x}_0) - \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)\|^2 \mathrm{d}s \right] \\ = -\lambda \mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta),$$

using the definition of $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ from Eq. 13. The second term becomes:

$$\mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{init}}} \left[- \int_0^1 \mathrm{Tr}(\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)) \mathrm{d}s \right] = C(\boldsymbol{y}),$$

using the definition of $C(\boldsymbol{y})$ from Eq. 14. Combining these, the ELBO becomes:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) \geq C(\boldsymbol{y}) - \lambda \mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta). \tag{17}$$

This completes the proof of Proposition 1. Our paper uses $\lambda = 1$ for simplicity, yielding the bound $C(\boldsymbol{y}) - \mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$. □

A.2 Rigorous Analysis of $C(\boldsymbol{y})$

We now provide a rigorous analysis of the term $C(\boldsymbol{y})$ in the ELBO and establish conditions under which minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ leads to favorable behavior of $C(\boldsymbol{y})$.

A.2.1 Geometric Interpretation of $C(\boldsymbol{y})$

The term $C(\boldsymbol{y})$ represents the negative expected divergence of the velocity field $\boldsymbol{v}_\theta$ along straight-line variational paths from $\boldsymbol{x}_0 \sim p_{\text{init}}$ to $\boldsymbol{y}$:

$$C(\boldsymbol{y}) = -\mathbb{E}_{s\sim\mathcal{U}[0,1], \boldsymbol{x}_0 \sim p_{\text{init}}} \left[\mathrm{Tr}(\nabla_{\boldsymbol{z}} \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)) \right]. \tag{18}$$

To understand its role, recall that in the exact likelihood computation for a flow model, we have:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}) = \log p_{\text{init}}(\boldsymbol{x}_0^*(\boldsymbol{y})) - \int_0^1 \mathrm{Tr}(\nabla_{\boldsymbol{x}} \boldsymbol{v}_\theta(\boldsymbol{x}_s^*(\boldsymbol{y}), s)) \mathrm{d}s, \tag{19}$$

where $\boldsymbol{x}_s^*(\boldsymbol{y})$ is the unique ODE trajectory satisfying $\dot{\boldsymbol{x}}_s^* = \boldsymbol{v}_\theta(\boldsymbol{x}_s^*, s)$ with $\boldsymbol{x}_1^*(\boldsymbol{y}) = \boldsymbol{y}$. The divergence integral measures the logarithmic volume change induced by the flow.

Our variational bound approximates this exact computation by averaging over straight-line paths rather than the true ODE trajectory. The quality of this approximation depends on how well the straight paths approximate the true flow geometry.

A.2.2 Relationship Between $C(\boldsymbol{y})$ and Distributional Alignment

We establish the key relationship between $C(\boldsymbol{y})$ and the alignment quality measured by $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$.

Lemma 1 (Consistency of Variational Paths)**.** *Under Assumption 1 (optimal $\boldsymbol{v}_\theta$), for $\boldsymbol{y} \sim p_{\text{data}}$, the straight-line variational paths $\boldsymbol{z}_s = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}$ satisfy:*

$$\mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{init}}} \left[\|(\boldsymbol{y} - \boldsymbol{x}_0) - \boldsymbol{v}_\theta(\boldsymbol{z}_s, s)\|^2 \right] = 0 \quad \forall s \in [0, 1]. \tag{20}$$

Consequently, $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta) = 0$ when $\boldsymbol{y} \sim p_{\text{data}}$.

Proof. By Assumption 1, for $\boldsymbol{y} \sim p_{\text{data}}$ and $\boldsymbol{x}_0 \sim p_{\text{init}}$, we have:

$$\boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s) = \boldsymbol{y} - \boldsymbol{x}_0.$$

Therefore, $\|(\boldsymbol{y} - \boldsymbol{x}_0) - \boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}, s)\|^2 = 0$ for all $\boldsymbol{x}_0$ and s, which implies the result. □

Theorem 1 (Monotonic Behavior of the ELBO)**.** *Consider two points $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^{d_1}$ such that $\mathcal{L}_{align}(\boldsymbol{y}_1; \theta) > \mathcal{L}_{align}(\boldsymbol{y}_2; \theta)$. If the velocity field $\boldsymbol{v}_\theta$ is L-Lipschitz in its spatial argument and satisfies Assumption 1, then:*

$$|C(\boldsymbol{y}_1) - C(\boldsymbol{y}_2)| \leq L \cdot d_1 \cdot \frac{1}{2}\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|, \tag{21}$$

where the factor $\frac{1}{2}$ comes from $\mathbb{E}_{s\sim\mathcal{U}[0,1]}[s] = \frac{1}{2}$. Moreover, if $\mathcal{L}_{align}(\boldsymbol{y}_1; \theta) - \mathcal{L}_{align}(\boldsymbol{y}_2; \theta) > \frac{L\cdot d_1}{2} \cdot \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|$, then:

$$\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}_2) - \log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}_1) > 0. \tag{22}$$

Proof. From the definition of $C(\boldsymbol{y})$ in Eq. 14:

$$\begin{aligned} C(\boldsymbol{y}_1) - C(\boldsymbol{y}_2) = &-\mathbb{E}_{s,\boldsymbol{x}_0}\left[\text{Tr}(\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}_1, s))\right] \\ &+ \mathbb{E}_{s,\boldsymbol{x}_0}\left[\text{Tr}(\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta((1-s)\boldsymbol{x}_0 + s\boldsymbol{y}_2, s))\right]. \end{aligned} \tag{23}$$

By the Lipschitz continuity of $\boldsymbol{v}_\theta$, its Jacobian $\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}, s)$ has bounded operator norm $\|\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}, s)\|_{op} \leq L$. Therefore:

$$\begin{aligned} |\text{Tr}(\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}_1, s)) - \text{Tr}(\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}_2, s))| &\leq d_1 \cdot \|\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}_1, s) - \nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}_2, s)\|_{op} \\ &\leq d_1 \cdot L \cdot \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|. \end{aligned} \tag{24}$$

Setting $\boldsymbol{z}_1 = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}_1$ and $\boldsymbol{z}_2 = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}_2$, we get $\|\boldsymbol{z}_1 - \boldsymbol{z}_2\| = s\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|$. Taking expectations yields Eq. 21.

For the second part, using the ELBO bound from Proposition 1:

$$\begin{aligned} \log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}_2) - \log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y}_1) &\geq [C(\boldsymbol{y}_2) - \mathcal{L}_{\text{align}}(\boldsymbol{y}_2; \theta)] - [C(\boldsymbol{y}_1) - \mathcal{L}_{\text{align}}(\boldsymbol{y}_1; \theta)] \\ &= [C(\boldsymbol{y}_2) - C(\boldsymbol{y}_1)] + [\mathcal{L}_{\text{align}}(\boldsymbol{y}_1; \theta) - \mathcal{L}_{\text{align}}(\boldsymbol{y}_2; \theta)]. \end{aligned} \tag{25}$$

Using the bound on $|C(\boldsymbol{y}_1) - C(\boldsymbol{y}_2)|$ and the condition on $\mathcal{L}_{\text{align}}(\boldsymbol{y}_1; \theta) - \mathcal{L}_{\text{align}}(\boldsymbol{y}_2; \theta)$, the result follows. □

A.2.3 Analysis of the Idealized Case

We address the mathematical singularity that arises in the idealized rectified flow case where $\boldsymbol{v}_\theta$ has the exact form $\boldsymbol{v}_\theta(\boldsymbol{z}, s) = \boldsymbol{z}/s$ for $s > 0$.

Proposition 3 (Regularization by Neural Network Parameterization)**.** *Let $\boldsymbol{v}_\theta$ be parameterized by a neural network with bounded weights. Then there exists a constant $M > 0$ such that:*

$$|\text{Tr}(\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}, s))| \leq M \quad \forall \boldsymbol{z} \in \textit{compact sets},\ s \in [\epsilon, 1] \tag{26}$$

for any $\epsilon > 0$. Consequently, $C(\boldsymbol{y})$ is well-defined and finite.

Proof. Neural networks with bounded parameters have Lipschitz continuous components. The Jacobian $\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta(\boldsymbol{z}, s)$ inherits this boundedness on compact sets, preventing the $1/s$ singularity from occurring exactly. The trace is therefore bounded, ensuring $C(\boldsymbol{y})$ remains finite. □

A.2.4 Practical Implications and Optimization Strategy

Our analysis establishes that:

1. Consistency Principle: When $\boldsymbol{y} \sim p_{\text{data}}$, both $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta) = 0$ and $C(\boldsymbol{y})$ takes on the value appropriate for samples from the target distribution.

2. Monotonicity Property: Theorem 1 shows that sufficiently large reductions in $\mathcal{L}_{\text{align}}(\boldsymbol{y}; \theta)$ guarantee improvements in the ELBO lower bound, even accounting for changes in $C(\boldsymbol{y})$.

3. Computational Tractability: While computing $C(\boldsymbol{y})$ requires evaluating Jacobian traces, minimizing only $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$ provides a computationally efficient proxy that, by Theorem 1, leads to ELBO improvements under reasonable conditions.

4. Robustness: Proposition 3 ensures that practical neural network implementations avoid the theoretical singularities, making the method stable in practice.

This analysis demonstrates that minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$ is not merely heuristic but has solid theoretical foundation as a strategy for maximizing the variational lower bound on $\log p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$.

A.3 The Significance of Assumptions

The derivation of Proposition 1 and its interpretation rely on several assumptions, as listed in Sec. A.1. In this section, we discuss the significance of each assumption.

Assumption 2 (Properties of the Velocity Field): Lipschitz continuity of $\boldsymbol{v}_\theta$ in its spatial argument ensures that the ODE $\dot{\boldsymbol{z}}_t = \boldsymbol{v}_\theta(\boldsymbol{z}_t, t)$ has unique solutions, fundamental for defining $p_1^{\boldsymbol{v}_\theta}(\boldsymbol{y})$. Differentiability is required for the Jacobian $\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta$ to exist, and thus for the divergence term $\mathrm{Tr}(\nabla_{\boldsymbol{z}}\boldsymbol{v}_\theta)$ in the ELBO to be well-defined. These are standard regularity conditions for flow-based models. Without them, the ELBO formulation would be ill-defined.

Assumption 3 (Variational Path Choice): The choice of straight-line paths $\boldsymbol{z}_s(\boldsymbol{x}_0, \boldsymbol{y}) = (1-s)\boldsymbol{x}_0 + s\boldsymbol{y}$ is a specific variational decision. This leads to the path velocity $\dot{\boldsymbol{z}}_s = \boldsymbol{y} - \boldsymbol{x}_0$, which is key to the definition of $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$. This assumption is thus crucial for the specific form of $\mathcal{L}_{\text{align}}$ used.

Assumption 4 (Variational Distribution Choice): Setting $q(\boldsymbol{x}_0|\boldsymbol{y}) = p_{\text{init}}(\boldsymbol{x}_0)$ greatly simplifies the ELBO by causing the term $\log p_{\text{init}}(\boldsymbol{x}_0) - \log q(\boldsymbol{x}_0|\boldsymbol{y})$ to vanish. This common choice implies the ELBO considers paths originating from the prior, without inferring a specific $\boldsymbol{x}_0$ for each $\boldsymbol{y}$. While simplifying, this choice affects the ELBO's tightness.

Assumption 5 (Weighting Factor in ELBO): Choosing $\lambda_s = \lambda$ makes the loss term in the ELBO directly correspond to $\mathcal{L}_{\text{align}}$. A time-dependent $\lambda_s > 0$ is also valid and could yield a tighter bound or differentially weight errors across time s. The constant λ ensures a direct link to the standard L2 norm in $\mathcal{L}_{\text{align}}$. This choice affects the ELBO's value but not its validity as a lower bound.

Assumption 1 (Optimality of $\boldsymbol{v}_\theta$): As detailed in Remark 1 of Sec. A.1, this assumption is not necessary for the mathematical derivation of Proposition 1 itself; the ELBO inequality holds for any $\boldsymbol{v}_\theta$ satisfying Assumption 2. However, Assumption 1 is paramount for the *interpretation* and *effectiveness* of minimizing $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$ as a strategy to align $\boldsymbol{y}$ with p_{data}. If $\boldsymbol{v}_\theta$ is optimal as defined, then $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$ will be minimized ideally to zero when $\boldsymbol{y}$ is drawn from p_{data}. Consequently, minimizing this loss for $\boldsymbol{y}$ encourages $\boldsymbol{y}$ to conform to p_{data}.

In essence, Assumptions 2 through 5 are primarily structural, defining the specific ELBO being analyzed. They ensure the bound is well-defined and takes the presented form. Assumption 1 concerning the optimality of $\boldsymbol{v}_\theta$ is interpretative, providing the rationale for why minimizing a component of this ELBO ($\mathcal{L}_{\text{align}}$) is a meaningful objective for achieving distributional alignment. The overall conclusion that minimizing $\mathcal{L}_{\text{align}}$ serves as a proxy for maximizing a log-likelihood lower bound relies on these assumptions.

B Additional Toy Examples

To further demonstrate the effectiveness of our proposed method, we present additional toy examples with diverse target distributions p_{data}: a Grid of Gaussians, Two Moons, Concentric Rings, a Spiral, and a Swiss Roll. For each of these distributions, following the visualization style of Fig. 3, we illustrate: (a) The optimized variables $\boldsymbol{y}$ (red triangles) and samples from p_{data} (blue dots), overlaid on the negative log-likelihood (NLL) landscape of p_{data} (background heatmap showing $-\log p_{\text{data}}(\cdot)$). (b) The landscape of the alignment loss $\mathcal{L}_{\text{align}}$ (background heatmap), with samples from p_{data} (blue dots). (c) The evolution of $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$ (blue solid line) and the true NLL $-\log p_{\text{data}}(\boldsymbol{y})$ (red dashed line) during the optimization of $\boldsymbol{y}$.

For the Grid of Gaussians, which is also a mixture of Gaussians, the NLL $-\log p_{\text{data}}(\boldsymbol{y})$, is computed analytically. For the other distributions (Two Moons, Concentric Rings, Spiral, and Swiss Roll), where an analytical form for p_{data} is not readily available, we estimate the NLL using Kernel Density

Estimation (KDE). This estimation is based on $N = 100,000$ samples drawn from the respective p_{data} and employs a Gaussian kernel with a bandwidth of $h = 0.1$. The probability density $\hat{p}_{\text{data}}(\mathbf{x})$ at a point $\mathbf{x}$ is estimated as:

$$\hat{p}_{\text{data}}(\mathbf{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \tag{27}$$

where $\mathbf{x}_i$ are the N samples drawn from p_{data}, d is the dimensionality (here, $d = 2$), and $K(\cdot)$ is the Gaussian kernel function. The NLL for an optimized variable $\boldsymbol{y}$ is then approximated by $-\log(\hat{p}_{\text{data}}(\boldsymbol{y}))$. This provides an empirical measure of how well $\boldsymbol{y}$ aligns with the target distribution as estimated by KDE.

The results for these additional toy examples are comprehensively presented in Fig. 5. Each row in this figure corresponds to one of the five target distributions. The left column (a,d,g,j,m) shows that the optimized variables $\boldsymbol{y}$ (red triangles) successfully converge to the high-density (low-NLL) regions of p_{data}. The middle column (b,e,h,k,n) demonstrates that the landscape of our alignment loss $\mathcal{L}_{\text{align}}$ closely mirrors the NLL surface of p_{data}, with true data samples (blue dots) residing in low-loss areas. The right column (c,f,i,l,o) confirms the strong positive correlation between $\mathcal{L}_{\text{align}}$ and the NLL of $\boldsymbol{y}$, as both decrease concomitantly during optimization. Furthermore, Fig. 6 visualizes the optimization trajectory of $\boldsymbol{y}$ for the initial mixture of Gaussians (from Sec. 5.1) alongside the five additional toy distributions. These sequential snapshots illustrate how minimizing $\mathcal{L}_{\text{align}}$ effectively steers the variables $\boldsymbol{y}$ from their initialization towards the intricate structures of the target distributions, reinforcing the robustness and efficacy of our alignment loss.

C Implementation Details

C.1 Implementation Details of the Toy Example

The primary toy example, illustrated in Figure 3, utilizes a 2D Mixture of Gaussians (MoG) as the target data distribution $p_{\text{data}}(\boldsymbol{x})$. This MoG distribution consists of 5 components, each with an isotropic standard deviation of 0.3. The means of these Gaussian components are distributed evenly on a circle of radius 3.0. Prior to model training, samples drawn from this MoG distribution are normalized by dividing by their standard deviation, which is empirically computed from a large batch of 10 million samples. In addition to the MoG, our toy experiments also encompassed other 2D synthetic distributions, including Spiral, Moons, Concentric Rings, Swiss Roll, and Grid of Gaussians, to demonstrate the versatility of our approach. The general setup for the flow model and learnable latents applies across these various distributions.

The conditional flow model, denoted $v_\phi(\boldsymbol{x}, t)$, is implemented using a MLP with AdaLN. This network has 2 input channels, 2 output channels, a hidden dimensionality of 512, and incorporates 4 residual blocks. The flow model is trained for 100,000 steps using the Adam optimizer (beta values of (0.9, 0.999) and no weight decay) with a constant learning rate of 1×10^{-4}, and a batch size of 256.

A set of 1,000 learnable latent variables $\{\boldsymbol{y}_i\}$ are initialized by sampling from a standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. These latents are then optimized to align with the target distribution p_{data} by minimizing the alignment loss $\mathcal{L}_{\text{align}}$. This alignment training phase also employs the Adam optimizer (betas=(0.9, 0.999), no weight decay), with a learning rate of 1×10^{-2}, and runs for 5,000 steps.

C.2 Implementation Details of the Flow Model

The flow model $\boldsymbol{v}_\theta(\boldsymbol{z}, t) : \mathbb{R}^{d_1} \times [0, 1] \to \mathbb{R}^{d_1}$ is implemented as a multi-layer perceptron (MLP) with 6 layers and 1024 hidden units per layer. The network employs GELU activation functions and incorporates time modulation through adaptive layer normalization (AdaLN) to handle the temporal dimension t. When dimension mismatch occurs between the latent space dimension d_1 and target feature space dimension d_2, fixed linear projection layers are applied to map target features to the appropriate dimension. These projection matrices are initialized with Gaussian weights scaled by $1/\sqrt{d_2}$ and remain frozen during training.

The flow model is trained using the flow matching objective on the target distribution p_{data} for 1 million steps. During training, the model learns to predict velocity fields that transport samples from a standard Gaussian base distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to the target distribution along straight-line interpolation

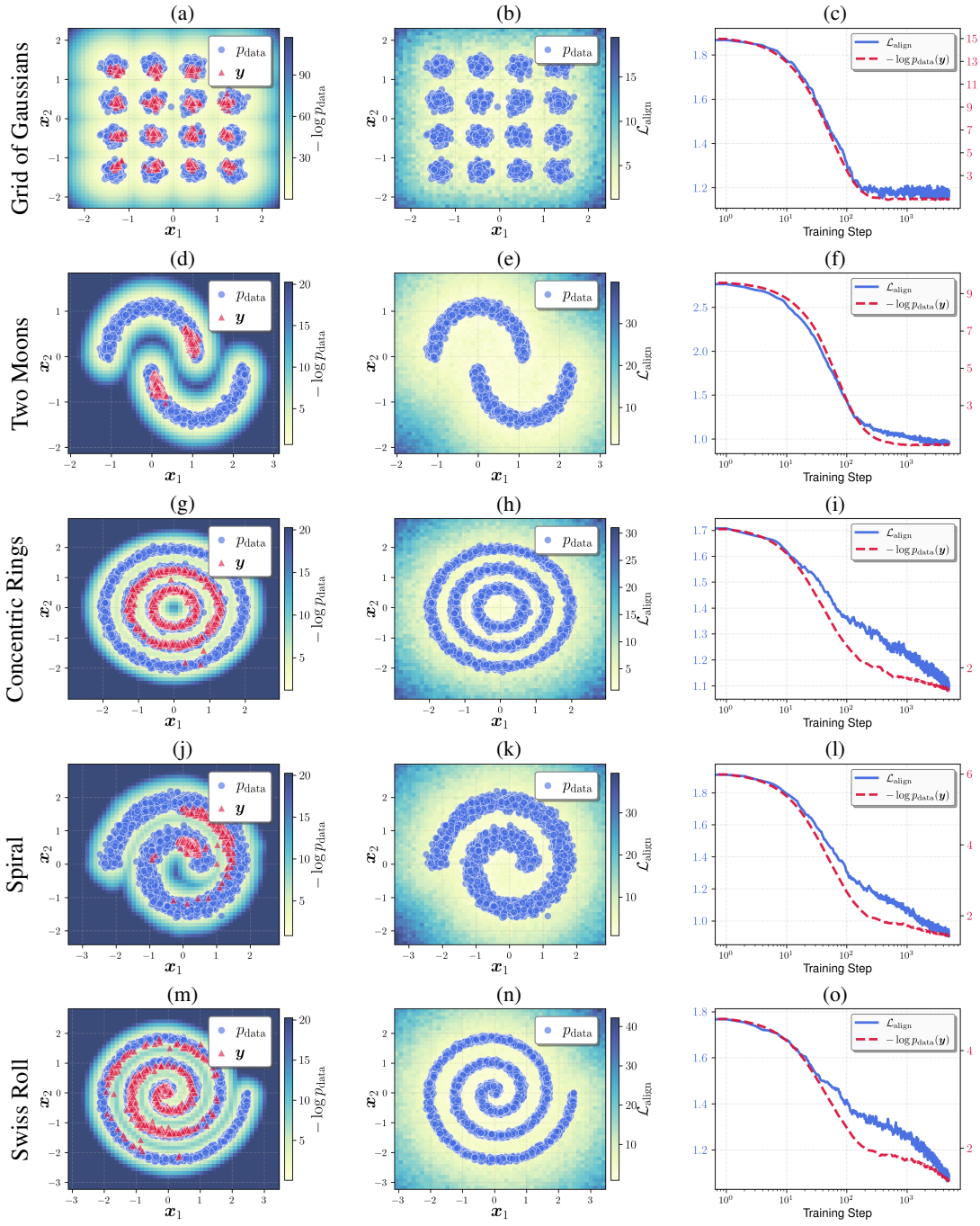


Figure 5: Further illustrations of our method's performance on various 2D toy examples. Each row corresponds to a different target distribution p_{data} (Grid of Gaussians, Two Moons, Concentric Rings, Spiral, and Swiss Roll). **Left column (a,d,g,j,m):** Optimized variables $\boldsymbol{y}$ (red triangles) and samples from p_{data} (blue dots). The background heatmap visualizes the negative log-likelihood (NLL) $-\log p_{\text{data}}(\cdot)$, with $\boldsymbol{y}$ converging to low-NLL (high-density) regions. **Middle column (b,e,h,k,n):** The landscape of the alignment loss $\mathcal{L}_{\text{align}}$ (heatmap) with p_{data} samples (blue dots). This landscape mirrors the NLL surface, and p_{data} samples are concentrated in areas of low $\mathcal{L}_{\text{align}}$. **Right column (c,f,i,l,o):** Training curves for $\mathcal{L}_{\text{align}}(\boldsymbol{y};\theta)$ (blue solid line) and NLL $-\log p_{\text{data}}(\boldsymbol{y})$ (red dashed line). Their strong positive correlation and concurrent decrease during optimization demonstrate that $\mathcal{L}_{\text{align}}$ effectively serves as a proxy for maximizing the log-likelihood of $\boldsymbol{y}$ under p_{data}.

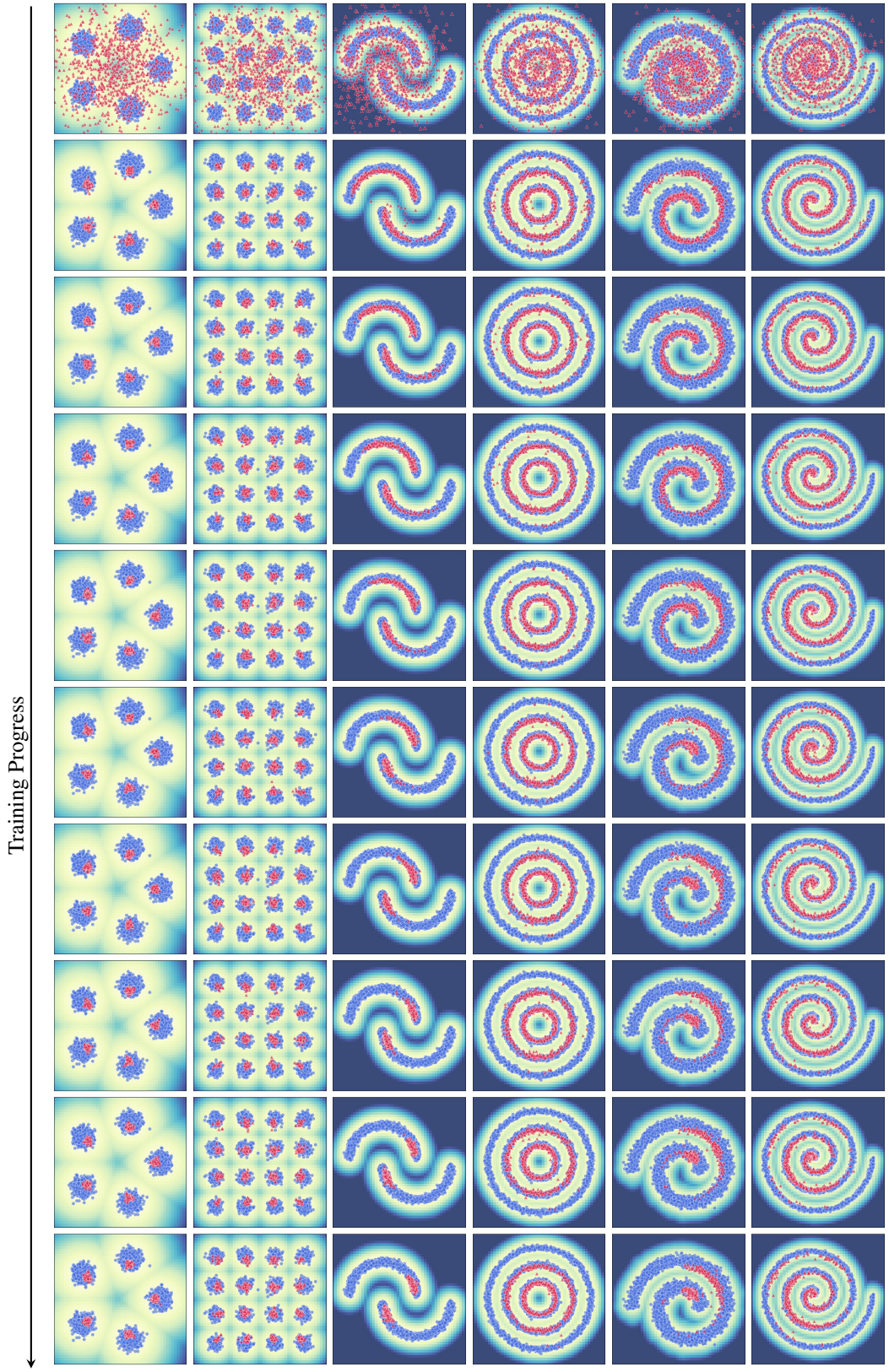


Figure 6: Evolution of the optimized variables $\boldsymbol{y}$ (red triangles) during training across various toy examples. Each column represents a target distribution p_{data}. The training progress demonstrates how minimizing $\mathcal{L}_{\text{align}}$ guides $\boldsymbol{y}$ to converge towards low-NLL (high-density) regions of p_{data}.

Table 3: Training Hyperparameters

Hyperparameter	Flow	Autoencoder	MAR
Global Batch Size		256	
Steps	$1000k$	$50k$	$250k$
Optimizer		AdamW	
Base Learning Rate		1.0×10^{-4}	
LR Scheduler	Cosine	Cosine	Constant
Warmup Steps	2.5k	2.5k	62.5k
Adam β_1		0.9	
Adam β_2	0.95	0.95	0.999
Weight Decay	1.0×10^{-4}	1.0×10^{-4}	0.02
Max Grad Norm		1.0	
Mixed Precision		BF16	
EMA Rate		0.9999	

paths. The training employs mixed precision (BF16) with gradient clipping and exponential moving averages (EMA). Upon completion of training, the flow model parameters θ are frozen and used for subsequent latent space alignment. Detailed hyperparameters are provided in Table 3.

C.3 Implementation Details of Autoencoders

Our autoencoder architecture follows the SoftVQ design, which employs Vision Transformer (ViT) based encoder and decoder networks. The encoder utilizes a ViT-Large model with patch size 14 from DINOv2 [48], initialized with pre-trained weights and fine-tuned with full parameter updates during training. The decoder employs the same ViT-Large architecture but is initialized randomly without pre-trained weights.

The training process utilizes adversarial loss with a DINOv2-based discriminator, incorporating patch-based adversarial training with hinge loss formulation. Perceptual loss is applied using VGG features with a warmup period of $10k$ steps. The model is trained for $50k$ steps with cosine learning rate scheduling and exponential moving averages for stable training dynamics. Unlike SoftVQ, we do not employ the sample-level alignment loss (i.e., REPA loss), making our method more general and efficient. Detailed hyperparameters are provided in Table 3.

We followed the SoftVQ implementation as closely as possible. While we can reproduce almost identical reconstruction results, our tokenizer doesn't quite match the generation performance of the released pre-trained model, even after significant effort to optimize it. We believe this gap comes from differences in the cleaned-up code and the specific hardware we used for training. To keep things fair and validate the effectiveness of our method, we conduct all experiments on *the same hardware with identical training settings*.

C.4 Implementation Details of MAR

We follow the original MAR-B implementation with several key modifications. We incorporate qk-norm in the attention mechanism and replace the diffusion head with a flow-based head trained using per-token flow matching loss. The original SD-KL-16 autoencoder is replaced with our trained autoencoders, applying input normalization with scaling factor 1.7052 estimated from sample batches.

Our model uses MAR-B architecture with 256×256 input images. The flow-based MLP head features adaptive layer normalization with 6 layers and 1024 hidden units per layer, identical to the original diffusion implementation. The model processes sequences of length 64 corresponding to our 64-token latent representation. More training details are provided in Table 3.

For inference, we employ an Euler sampler with 100 steps for the flow-based generation. The autoregressive sampling is limited to 64 steps. Generation uses batch size 256 and produces 50,000 images for evaluation. All evaluations use the standard toolkit from guided diffusion with FID and IS metrics computed at regular intervals during training.

D Additional Discussions for the Experiments

Here we provide additional discussions and analysis for the experiments presented in Section 5.2.

Does Johnson-Lindenstrauss Lemma Really Hold? While the Johnson-Lindenstrauss (JL) lemma theoretically guarantees that random projections preserve distances with high probability, our experimental setup violates its conditions due to the large sample size relative to the target dimension. However, our results demonstrate that random projections can still preserve distributional structure to a sufficient extent for effective alignment. In our ablation study with Tab. 2a, random projection achieves the best performance with FID of 11.89 and IS of 102.23, significantly outperforming PCA (FID: 14.95, IS: 83.59) and average pooling (FID: 16.06, IS: 60.37). This suggests that the structure-preserving properties of random projections, even when the JL lemma doesn't strictly hold, are more beneficial than the variance-maximizing properties of PCA or the spatial averaging of pooling operations.

Continuous or Discrete? Our method demonstrates robustness across both continuous and discrete target distributions. Continuous semantic features from DinoV2 achieve the best generation performance among all variants in Tab. 1 and the discrete textual features from Qwen also achieve effective performance. In contrast, discrete VQ features perform poorly, likely due to structural limitations imposed by low dimensionality (8-dim). The collapse observed in discrete VQ experiments during training can be attributed to the insufficient capacity of the low-dimensional latent space to capture the complexity of ImageNet data while simultaneously satisfying the alignment constraint.

Why Textual Features Work? The surprising effectiveness of textual embeddings (Qwen) for visual generation warrants deeper analysis. Despite being trained on text data, Qwen embeddings achieve competitive generation performance (FID: 11.89 without CFG) and the best PSNR (23.12) among aligned methods. This suggests that high-quality textual representations capture abstract semantic structures that are transferable across modalities. The 896-dimensional Qwen embeddings provide a rich semantic space that can effectively constrain the visual latent space without being overly restrictive. This cross-modal transferability indicates that the structural benefits of alignment are not limited to within-modality features.

Is Generation Loss a Good Indicator? The training loss in generation of our aligned autoencoders is significantly lower than other models. However, we observe that lower training losses do not necessarily translate to better generation results, even for flow-based models where loss is proven to be a direct indicator for generation performance. This paradox can be attributed to the simplification of the latent space under strong alignment constraints. While simplified latent spaces are easier for generative models to sample from (hence lower training losses), they may sacrifice the diversity and fine-grained details necessary for high-quality generation. This suggests that generation quality depends not only on the ease of modeling the latent distribution but also on the expressiveness and diversity preserved in the aligned space.

How to Select the Prior? The optimal choice of target distribution remains an open research question. Our experiments suggest several guidelines: (1) Higher dimensionality generally enables better performance, as evidenced by the poor performance of 8-dimensional VQ features compared to higher-dimensional alternatives. (2) Semantic richness matters, but not necessarily complexity—simple textual features can match sophisticated visual features. (3) The structural properties of the target distribution (e.g., smoothness, cluster separation) may be more important than its semantic content for generation quality.