# Tunable Latent Generative Priors for Compressed Sensing and Inverse Problems

**Sean Gunn**
Northeastern University
gunn.s@northeastern.edu

**Jorio Cocola**
Harvard University
jcocola@seas.harvard.edu

**Oliver De Candido**
Technical University of Munich
oliver.de-candido@tum.de

**Vaggos Chatziafratis**
UC Santa Cruz
vaggos@ucsc.edu

**Paul Hand**
Northeastern University
p.hand@northeastern.edu

## Abstract

Latent generative models have emerged as powerful priors for solving inverse problems. These models typically represent a class of natural signals at a single, fixed complexity, governed by the latent dimensionality. This can be limiting: depending on the problem, a latent dimensionality that is too small may result in high representation error, while one that is too large may overfit to noise. We develop tunable latent priors for diffusion models, normalizing flows, and variational autoencoders, leveraging nested dropout. Across tasks including compressed sensing, inpainting, denoising, and phase retrieval, we show empirically that tunable priors consistently achieve lower reconstruction errors than fixed-complexity baselines. In the linear denoising setting, we derive the optimal complexity in closed form, showing how it depends on the noise level and the signal spectrum. This work demonstrates the potential of tunable latent generative priors and motivates both the development of supporting theory and their application across a wide range of inverse problems.

## 1 Introduction

Inverse problems aim to reconstruct an unknown signal, potentially corrupted by noise, from a set of measurements given by a forward model, which may or may not be known in advance. Such a formulation applies to various image-processing applications, including compressed sensing, denoising, and super-resolution. In practice, inverse problems are ill-posed and thus they require prior information about the signal to yield a successful recovery [1].

Deep generative models have been demonstrated to be powerful signal priors when used to solve inverse problems [2, 3, 4, 5, 6, 7]. The process of using a deep generative prior typically has two phases: training and inversion. In the training phase, a generative neural network is trained on a dataset representative of the natural signal class intended for inversion. In the second phase, the model parameters are fixed from training, and an algorithm is deployed to estimate the signal of interest for a given forward operator. This approach has the benefit that the prior can be learned

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

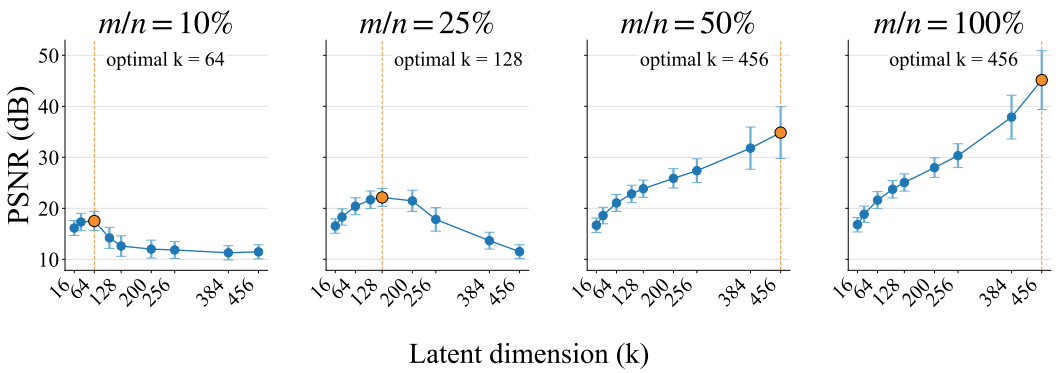



Figure 1: **Intermediate latent dimensionalities yield the best reconstruction at low measurement ratios.** We train a separate injective flow model (no parameter sharing) on MNIST ($n = 32 \times 32 = 1024$) for each latent dimensionality $k$ from 16 to 456. Each panel corresponds to a different measurement ratio $m/n$: for $m = n$ the forward operator is the identity matrix, and for $m < n$ it is a random Gaussian matrix. At small $m/n$, intermediate values of $k$ ($64 \leq k \leq 200$) give the highest PSNR, and the optimal $k$ grows with $m$. Error bars show $\pm 1$ standard deviation.

in isolation from the inverse problems being solved. Thus, the approach can apply to a variety of inverse problems, which is in contrast to other neural network-based approaches such as end-to-end training.

In recent years, there has been a dominant framework for using generative models as priors for inverse problems. Within this framework, generative priors have a fixed complexity that is set during training. For example, existing priors include generative adversarial networks (GANs), which typically have a latent space of fixed low dimensionality [2, 3, 8, 9, 10]; normalizing flow models, which have a latent space of fixed high dimensionality equal to that of the images [4, 11, 12, 13]; and diffusion models, which maintain information about the probability density over the fixed high-dimensional space of all images [7, 14, 15, 16, 17].

We step outside the dominant framework and demonstrate the benefits of solving inverse problems using generative priors whose complexity can be selected by the user at inference time, after training. Such generative priors simultaneously maintain representations of varying complexities of the natural signal class, and we refer to them as **tunable generative priors**. *In this paper, we demonstrate that latent tunable generative priors can significantly improve reconstruction when the latent dimension $k$ is matched to a specific inverse problem. By training the model to preserve meaningful representations across values of $k$, we enable practitioners to select an appropriate $k$ at inference time, without retraining.*

To motivate our approach, we first consider injective flows [18]. We train a family of models, each with a distinct latent dimensionality $k$ and no parameter sharing, and apply them to compressed sensing with random measurements. As shown in Figure 1, reconstruction quality (PSNR) follows an inverted U-shaped curve: with fewer measurements, intermediate values of $k$ outperform both larger and smaller ones. Moreover, the optimal complexity depends on the number of available measurements. While this experiment requires training separate models for each $k$, a naïve and computationally expensive strategy, it illustrates the importance of tunability. This motivates the more efficient algorithms we develop next, which scale to practical settings.

As an initial theoretical exploration into the effect of model tunability, we study denoising in the case of an invertible linear generative model and the best lower-dimensional linear models that approximate it. In this setting, we derive an explicit maximum-likelihood criterion for selecting the optimal model complexity. Our analysis shows that the optimal complexity depends on the generator's singular values relative to the noise level: directions with singular values above the noise level improve reconstruction, while lower-signal directions can overfit noise.

Building on this motivation, we will show how tunability can be achieved efficiently in three major classes of generative models: variational autoencoders (VAEs) [19], normalizing flows (NFs) [20], and latent diffusion models (LDMs) [21]. In each case, we design a single family of models with parameter sharing and provide an efficient training procedure. For LDMs, we introduce a new algorithm based on nested dropout [22], described in Section 3.1. For NFs, we adopt an existing

ordering method [23] (Section C). For VAEs, we extend the adversarial objective of [24] with a nested dropout regularization term. Across all three settings, we empirically demonstrate that tunable generative priors consistently achieve lower reconstruction errors than fixed-complexity baselines over a range of inverse problems, undersampling ratios, and noise levels (Figure 3).

The results in this paper demonstrate the benefits of using generative priors with a tunable complexity for inverse problems both empirically and theoretically. Empirically, we show that tunable complexity consistently improves reconstruction quality across multiple generative model architectures, inverse problem types, and inversion algorithms.

**Our main contributions are as follows:**

- We observe a potentially surprising phenomenon in the use of latent generative priors for solving inverse problems. We train a single generative model to simultaneously represent the natural signal class across multiple latent dimensionalities $k$. Across a variety of architectures, inverse problems, and inversion algorithms, we empirically find that some intermediate latent dimensionality yields an improved reconstruction error.
- We propose a new training algorithm for latent diffusion that utilizes nested dropout and a convex combination of the original and truncated latent objectives. This yields a tunable latent diffusion model that learns hierarchical representations across latent dimensionalities, enabling a single model to be tuned as a prior for downstream inverse problems.

## 2 Background and Related Work

The background will focus on diffusion models due to their relevance in recent research trends. For additional details on normalizing flows and variational autoencoders, we refer the reader to the appropriate references.

### 2.1 Diffusion Models

The standard diffusion framework gradually corrupts data $\boldsymbol{x}_0 \sim p_{\text{data}}$ $\boldsymbol{x} \in \mathbb{R}^n$ into noise $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I}_n)$ via a forward noising process, and learns a parameterization of the reverse process to recover the data distribution [25, 26, 27]. We adopt latent diffusion models (LDMs) [21], which improve computational efficiency by performing diffusion in a learned latent space: a *variational autoencoder (VAE)* [19] with encoder $\mathcal{E}\colon \mathbb{R}^n \to \mathbb{R}^d$ and decoder $\mathcal{D}\colon \mathbb{R}^d \to \mathbb{R}^n$ (typically $d \ll n$) yields $\boldsymbol{z}_0 = \mathcal{E}(\boldsymbol{x}_0)$ and $\boldsymbol{x}_0 \approx \mathcal{D}(\boldsymbol{z}_0)$ for all $\boldsymbol{x}_0 \sim p_{\text{data}}$. The encoder maps $p_{\text{data}}$ to the latent distribution $p_Z \coloneqq \mathcal{E}_{\#} p_{\text{data}}$, and the diffusion process then operates on the latent variable $\boldsymbol{z}_t$.

With schedule $\{\beta_t\}_{t=1}^T$, $\alpha_t \coloneqq 1 - \beta_t$, and $\bar{\alpha}_t \coloneqq \prod_{j=1}^t \alpha_j$ (with $\bar{\alpha}_0 = 1$), the forward marginal in latent space is

$$\boldsymbol{z}_t = \sqrt{\bar{\alpha}_t}\, \boldsymbol{z}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_d), \tag{1}$$

which allows sampling $\boldsymbol{z}_t$ at any timestep $t$ directly from $\boldsymbol{z}_0$. The reverse process is parameterized by a neural network:

$$\boldsymbol{z}_{t-1} = \tfrac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{z}_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\, \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)\Big) + \sigma_t^{\text{DDPM}}\, \boldsymbol{\epsilon}'_t, \qquad \boldsymbol{\epsilon}'_t \sim \mathcal{N}(0, \boldsymbol{I}_d), \tag{2}$$

where $\sigma_t^{\text{DDPM}} = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,(1-\alpha_t)}$ is the posterior standard deviation. Write $\hat{\boldsymbol{\epsilon}}_t = \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$ for the predicted noise and define the clean-latent estimate

$$\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t) = \frac{\boldsymbol{z}_t - \sqrt{1-\bar{\alpha}_t}\, \hat{\boldsymbol{\epsilon}}_t}{\sqrt{\bar{\alpha}_t}}.$$

DDIM [28] uses a distinct reverse update:

$$\boldsymbol{z}'_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, \hat{\boldsymbol{z}}_0(\boldsymbol{z}_t) + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\, \hat{\boldsymbol{\epsilon}}_t + \sigma_t \boldsymbol{\epsilon}'_t, \qquad \sigma_t = \eta \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}(1-\alpha_t)}.$$

Here $\eta \in [0, 1]$ controls DDIM stochasticity, and $\eta = 0$ gives deterministic DDIM.

A denoising network $\boldsymbol{\epsilon}_\theta$ (typically a U-Net) is trained to predict the noise in the latent space:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{t\sim\mathcal{U}(\{1,\ldots,T\}),\ \boldsymbol{z}_0,\ \boldsymbol{\epsilon}\sim\mathcal{N}(0,\boldsymbol{I})} \big\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\sqrt{\bar{\alpha}_t}\,\boldsymbol{z}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon},\ t\big)\big\|_2^2, \tag{3}$$

where $t \sim \mathcal{U}(\{1,\ldots,T\})$ denotes that the timestep is sampled uniformly from all diffusion steps. To enable latent diffusion, the autoencoder is trained using reconstruction, latent regularization, and adversarial losses. These terms respectively encourage accurate reconstructions, alignment with a reference latent prior, and perceptually realistic outputs:

$$\mathcal{L}_{\text{Autoencoder}} = \mathcal{L}_{\text{rec}}\big(\boldsymbol{x},\ \mathcal{D}(\mathcal{E}(\boldsymbol{x}))\big) + \lambda_{\text{reg}}\,\mathcal{L}_{\text{reg}}\big(\mathcal{E}(\boldsymbol{x})\big) + \lambda_{\text{adv}}\,\mathcal{L}_{\text{adv}}\big(\mathcal{C},\ \boldsymbol{x},\ \mathcal{E}(\boldsymbol{x})\big). \tag{4}$$

### 2.2 Inverse Problems with Generative Priors

We consider the general inverse problem of recovering an unknown signal $\boldsymbol{x} \in \mathbb{R}^n$ from noisy measurements $\boldsymbol{y} \in \mathbb{R}^m$,

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_m), \tag{5}$$

where $\mathcal{A}\colon \mathbb{R}^n \to \mathbb{R}^m$ [1] is the forward operator (*e. g.*, a linear projection, convolutional blur, or other transformation), and $\boldsymbol{\eta}$ is additive noise drawn from a Gaussian distribution. Generative models can be employed as priors for inverse problems in two main ways: supervised, where the forward operator is known during training and models are trained on paired data $\{(x_i, y_i)\}$, or unsupervised, where the forward operator is unknown [1, 29].

For inverse problems, we modify the reverse dynamics in Eq. (2) to account for measurements $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\eta}$. Since we operate entirely in latent space ($\boldsymbol{x} = \mathcal{D}(\boldsymbol{z})$), posterior sampling is guided by the conditional score, which decomposes as

$$\nabla_{\boldsymbol{z}_t} \log p_t(\boldsymbol{z}_t \mid \boldsymbol{y}) = \nabla_{\boldsymbol{z}_t} \log p_t(\boldsymbol{z}_t) + \nabla_{\boldsymbol{z}_t} \log p_t(\boldsymbol{y} \mid \boldsymbol{z}_t). \tag{6}$$

The first term is provided by the pretrained diffusion prior through the denoising network $\boldsymbol{\epsilon}_\theta$, while the second term enforces data-consistency with the forward operator $\mathcal{A}$ under the measurement model.

Following the taxonomy of interleaving methods highlighted by [30], these solvers alternate unconditional reverse diffusion steps (*e. g.*, DDIM) with data-consistency corrections that either explicitly approximate the measurement likelihood or enforce feasibility. One line of methods explicitly approximates the measurement-likelihood term via a projection or gradient update [15, 31, 17, 32, 33]. A canonical formulation is latent diffusion posterior sampling [34]:

$$\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y} \mid \boldsymbol{z}_t) \ \approx\ -\frac{1}{2\sigma^2} \nabla_{\boldsymbol{z}_t} \left\| \boldsymbol{y} - \mathcal{A}(\mathcal{D}(\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t))) \right\|_2^2. \tag{7}$$

This is a Gaussian-likelihood approximation for measurement noise $\sigma > 0$, evaluated at the noise-prediction estimate $\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t)$ defined in Section 2.1. This estimate approximates the posterior mean via Tweedie's formula [35]. Differentiation is through the full map $\boldsymbol{z}_t \mapsto \hat{\boldsymbol{z}}_0(\boldsymbol{z}_t) \mapsto \mathcal{D}(\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t)) \mapsto \mathcal{A}(\mathcal{D}(\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t)))$, including the denoising network.

Another class of methods does not explicitly compute the measurement likelihood, but instead approximates the feasible set of solutions $\{\boldsymbol{x} \mid \boldsymbol{y} = \mathcal{A}(\boldsymbol{x})\}$ [31, 36, 37]. In particular, ReSample [37] employs a hard data-consistency term that enforces $\mathcal{D}(\boldsymbol{z}) \in \{\, \boldsymbol{x} \mid \boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) \,\}$, which is approximated in practice by gradient descent.

## 3 Methods

This section describes the training of a tunable latent diffusion model (Section 3.1) and its application as a prior for inverse problems (Section 3.2). We begin by training a variational autoencoder (VAE) as the backbone of the latent diffusion model. The VAE consists of an encoder $\mathcal{E}\colon \mathbb{R}^n \to \mathbb{R}^d$ and a decoder $\mathcal{D}\colon \mathbb{R}^d \to \mathbb{R}^n$, such that for samples $\boldsymbol{x}_0 \sim p_{\text{data}}(\boldsymbol{x}_0)$ we have $\boldsymbol{x}_0 \approx \mathcal{D}(\mathcal{E}(\boldsymbol{x}_0))$. Once the autoencoder is trained, a denoising network $\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$ is trained in latent space to predict the Gaussian noise at each diffusion step $t$. To generate a sample, we run the reverse diffusion process in latent space with $\boldsymbol{\epsilon}_\theta$ and decode the final latent variable with $\mathcal{D}$.

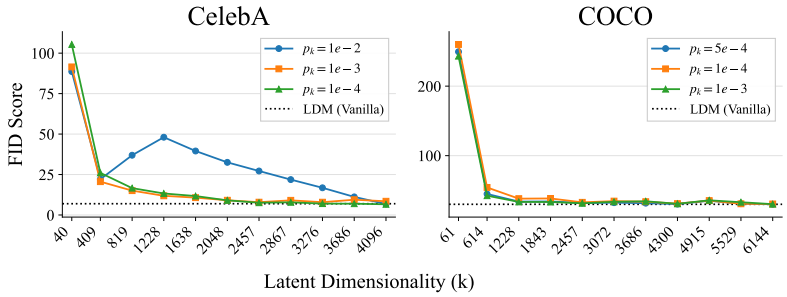



Figure 2: **Nested dropout training produces tunable latent diffusion models that maintain generation quality across latent dimensionalities.** FID score is plotted as a function of latent dimensionality $k$ for models trained with different values of the dropout distribution parameter $p$. The vanilla LDM baseline (dotted line) operates only at full dimensionality. As $k$ increases, the tunable models approach baseline performance while retaining the ability to generate from lower-dimensional representations. Results are evaluated on 50k training images from each dataset. FID scores computed with [38].

### 3.1 Training a Tunable Latent Diffusion Model

We aim to train a latent generative model that can represent the natural signal class across multiple latent dimensionalities. To achieve this, we leverage nested dropout [22], which imposes an ordered structure on the latent variables by always preserving a prefix of coordinates. Formally, let $k \sim p_k$ be drawn from $\{1, \ldots, d\}$. In our experiments, we use a truncated geometric distribution with success parameter $p$, but other distributions over $\{1, \ldots, d\}$ can be adopted depending on the application. The truncation operator is defined as $\boldsymbol{z}_{\downarrow k} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k, 0, \ldots, 0]$ with $k \leq d$. Given $\boldsymbol{z} = \mathcal{E}(\boldsymbol{x})$, reconstruction is performed from $\boldsymbol{z}_{\downarrow k}$, which encourages earlier coordinates to carry more information about the signal class.

Latent diffusion is trained in two stages. First, the autoencoder is trained to reconstruct $\boldsymbol{x} \in \mathbb{R}^n$ robustly across a range of latent dimensions $k$. Building on Section 2.1, the VAE backbone is trained with reconstruction, regularization, and adversarial terms, which we extend here with a nested dropout objective. This follows the standard VAE training objective with an added nested dropout term:

$$\mathcal{L}_{\text{VAE}} = \min_{\mathcal{E},\mathcal{D}} \max_{\mathcal{C}} \Big[ \mathcal{L}_{\text{rec}}\big(\boldsymbol{x}, \mathcal{D}(\mathcal{E}(\boldsymbol{x}))\big) + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}\big(\mathcal{E}(\boldsymbol{x})\big) + \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}\big(\mathcal{C}, \boldsymbol{x}, \mathcal{E}(\boldsymbol{x})\big) + \lambda_{\text{drop}} \mathcal{L}_{\text{drop}}\big(\boldsymbol{x}, \mathcal{D}(\mathcal{E}(\boldsymbol{x})_{\downarrow k})\big) \Big], \tag{8}$$

where $\mathcal{C} : \mathbb{R}^n \to (0, 1)$ is a discriminator. We adopt a perceptual loss [39] for $\mathcal{L}_{\text{drop}}$.

In the second stage, the diffusion model is trained in latent space with a loss that interpolates between the standard diffusion objective and its truncated-latent variant. For $\lambda \in [0, 1]$, we define

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathcal{E}(\boldsymbol{x}), \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_d), t \sim \mathcal{U}(\{1, \ldots, T\})} \Big[ (1 - \lambda) \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t) \right\|_2^2 + \lambda \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta((\boldsymbol{z}_t)_{\downarrow k}, t) \right\|_2^2 \Big]. \tag{9}$$

The first term is the standard latent diffusion objective, while the second applies it to the truncated latent $(\boldsymbol{z}_t)_{\downarrow k}$, encouraging effective denoising even from a reduced representation. Here, $\boldsymbol{z}_t$ is sampled from the forward DDPM process, Eq. (1), and $k$ is drawn from the same distribution $p_k$ used in the VAE objective, ensuring consistency. The result is a hierarchically organized latent space: the first few coordinates capture the most essential signal structure, with each additional coordinate contributing finer detail as the model's representational capacity grows with $k$. This effect is illustrated in Figure 2, which shows how FID varies with latent dimensionality under different hyperparameters. Lastly, for details about training, please see Appendix B.2.

### 3.2 Inverse Problems with Tunable Priors

A broad family of methods applies a data-consistency step via projection, gradient, or small optimization after each reverse update to move the prior iterate toward the feasible set $\{\boldsymbol{x} \mid \mathcal{A}(\boldsymbol{x}) = \boldsymbol{y}\}$. In

**Algorithm 1** General Template for Tunable Diffusion Priors

**Input:** $\boldsymbol{y}$, $\mathcal{A}$, $\mathcal{D}$, $\boldsymbol{\epsilon}_\theta$, steps $T$, schedule $\{\bar{\alpha}_t\}_{t=0}^{T}$, parameter $k$
Reverse sampler (DDIM or DDPM), data-consistency correction
**Output:** $\mathcal{D}(\boldsymbol{z}_0)$
$\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I}_d)$; $\boldsymbol{z}_T \leftarrow (\boldsymbol{z}_T)_{\downarrow k}$
**for** $t = T, \ldots, 1$ **do**
    $\hat{\boldsymbol{\epsilon}}_t \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$
    $\hat{\boldsymbol{z}}_0 \leftarrow (\boldsymbol{z}_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\boldsymbol{\epsilon}}_t)/\sqrt{\bar{\alpha}_t}$
    $\boldsymbol{z}'_{t-1} \leftarrow$ chosen DDIM or DDPM reverse update using $\boldsymbol{z}_t, \hat{\boldsymbol{z}}_0, \hat{\boldsymbol{\epsilon}}_t$
    $\boldsymbol{z}_{t-1} \leftarrow$ data-consistency correction of $\boldsymbol{z}'_{t-1}$ using $\boldsymbol{y}, \mathcal{A}, \mathcal{D}$
    $\boldsymbol{z}_{t-1} \leftarrow (\boldsymbol{z}_{t-1})_{\downarrow k}$
**end for**
**return** $\mathcal{D}(\boldsymbol{z}_0)$

latent space, representative examples include Posterior Sampling with Latent Diffusion (PSLD) [34], ReSample [37], DAPS [40], and our formulation in Algorithm 2. Algorithm 2 is a simplified mathematical template for a broadly applicable instantiation across a variety of inverse problems, including nonlinear forward operators. The experiments in Figure 3 employ an implementation of this template, while Algorithm 1 describes how the same tunability principle can be incorporated into other latent-space inversion methods.

Both algorithms use $t = T, \ldots, 1$ with $\bar{\alpha}_0 = 1$ and full latent dimension $d$. For a subsampled schedule, $t-1$ denotes the previous selected timestep (the terminal clean state is indexed by 0), and the reverse formulas use the effective $\alpha_t = \bar{\alpha}_t/\bar{\alpha}_{t-1}$ and $\beta_t = 1 - \alpha_t$.

Starting from Gaussian noise truncated to $k$ coordinates in the latent space, Algorithm 1 alternates a learned generative update, a data-consistency correction, and a truncation step. The generative update follows the diffusion prior, the data-consistency step enforces agreement with the measurements $\boldsymbol{y} = \mathcal{A}(\mathcal{D}(\boldsymbol{z}))$, and the truncation operator $(\cdot)_{\downarrow k}$ controls the effective complexity of the representation. Although Algorithm 1 is written for diffusion priors, the same tunability principle can be applied to flow-based generative models [41], where the generative update is defined by a learned velocity field rather than a denoising or score model.

## 4 Theory for Denoising with Linear Generative Model

The goal of this section is to provide preliminary theoretical conclusions that show tunability can lead to improved reconstruction errors relative to corresponding nontunable models. Further, this

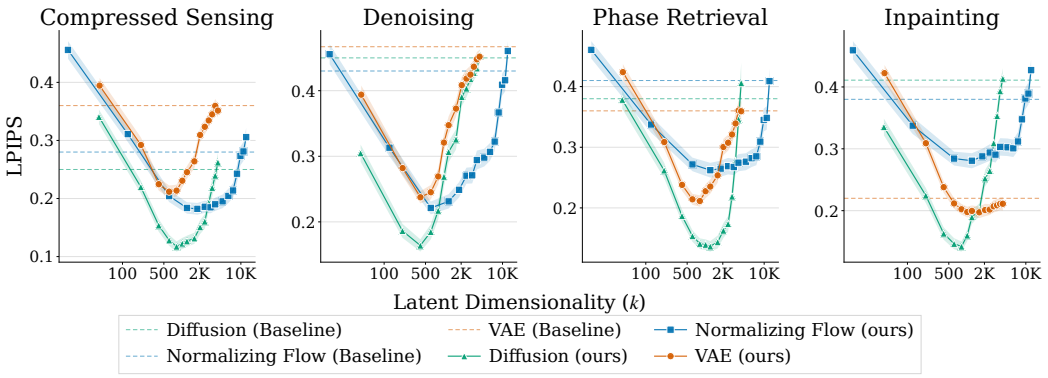



Figure 3: **Tunable priors outperform fixed-complexity baselines across multiple different methods and inverse problems.** Reconstruction performance (LPIPS, lower is better) is shown as a function of latent dimensionality $k$ for compressed sensing, denoising, phase retrieval, and inpainting. The tunable LDM prior is trained on CelebA and evaluated on FFHQ. It is compared against the baseline operating at full dimensionality.

section aims to provide justification for how to select the tuning parameter for a generative model with tunable complexity. We illustrate this in the setting of denoising under additive Gaussian noise, where the benefits of tunability can already be shown for linear generative models.

Consider a family of linear generative models given as follows. Fix an invertible $\boldsymbol{G} \in \mathbb{R}^{n\times n}$, and let $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition with $\boldsymbol{\Sigma} = \operatorname{diag}(s_1, \dots, s_n) \in \mathbb{R}^{n\times n}$. For $k \in [n]$, let $\boldsymbol{U}_k$ contain the first $k$ columns of $\boldsymbol{U}$, let $\boldsymbol{\Sigma_k} = \operatorname{diag}(s_1, \dots, s_k) \in \mathbb{R}^{k\times k}$, and choose $\boldsymbol{V}_k \in \mathbb{R}^{k\times n}$ with orthonormal rows and zero last $n-k$ columns. Set $\boldsymbol{V}_n = \boldsymbol{V}^T$ and $\boldsymbol{G_k} = \boldsymbol{U}_k\boldsymbol{\Sigma_k}\boldsymbol{V}_k \in \mathbb{R}^{n\times n}$, so $\boldsymbol{G}_n = \boldsymbol{G}$. Each $\boldsymbol{G}_k$ induces a probability distribution over $\mathbb{R}^n$ by

$$\boldsymbol{x} = \boldsymbol{G}_k\boldsymbol{z}, \quad \boldsymbol{z} = (\boldsymbol{z}^k, 0) \in \mathbb{R}^n, \quad \boldsymbol{z}^k \sim \mathcal{N}(0, \boldsymbol{I}_k).$$

We denote this distribution by $p_{\boldsymbol{G}_k} = \mathcal{N}(0, \boldsymbol{G}_k\boldsymbol{G}_k^T)$. The parameter $k$ governs the complexity of the modeled signal class.

We consider the following denoising problem. Let $\boldsymbol{y} = \boldsymbol{x}_0 + \boldsymbol{\eta}$, where $\boldsymbol{x}_0 \sim p_{\boldsymbol{G_n}}$ and $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I}_n)$ are independent. Our goal is to recover $\boldsymbol{x}_0$ given $\boldsymbol{y}$ and $\boldsymbol{G}_n$. For a given $k$, we consider a maximum likelihood estimate (MLE) of $\boldsymbol{x}_0$ under the signal prior $p_{\boldsymbol{G}_k}$. This results in the following optimization problem over the latent space:

$$\hat{\boldsymbol{z}}(k) \in \arg\min_{\boldsymbol{z}\in\mathbb{R}^n} \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{G_k}\boldsymbol{z}\right\|^2, \tag{10}$$

where the estimated signal is $\hat{\boldsymbol{x}}(k) = \boldsymbol{G}_k\hat{\boldsymbol{z}}(k) = \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{y}$. The following theorem provides an exact expression for the mean square error of the estimate above.

**Theorem 4.1.** *Suppose we have a family $\{\boldsymbol{G_k}\}_{k=1,\dots,n}$ of generative models as given above, and let $p_{\boldsymbol{G_k}} = \mathcal{N}(0, \boldsymbol{G_k}\boldsymbol{G_k}^T)$, and let $\boldsymbol{G_n} \in \mathbb{R}^{n\times n}$ have singular values $s_1 \geq s_2 \geq \cdots \geq s_n > 0$. Let $\boldsymbol{x}_0 \sim p_{\boldsymbol{G}_n}$ and $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I}_n)$ be independent. Then the estimator given by Eq. (10) yields*

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\left[\,\|\hat{\boldsymbol{x}}(k) - \boldsymbol{x}_0\|^2\,\right] = k\sigma^2 + \sum_{j=k+1}^{n} s_j^2.$$

The exact expression for reconstruction error in Theorem 4.1 makes it possible to analytically find the optimal value of the tunable complexity parameter $k$, as established in the following corollary.

**Corollary 4.2.** *Under the assumptions of Theorem 4.1, suppose additionally that $s_1 \geq \sigma$. Then the reconstruction error is minimized by*

$$k^\star = \max\left\{k \mid s_k \geq \sigma\right\} \in \arg\min_{k\in[n]} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\hat{\boldsymbol{x}}(k) - \boldsymbol{x}_0\|^2,$$

The corollary shows that the optimal model retains only singular directions above the noise level, so higher noise leads to a smaller optimal $k$. Proofs are provided in Appendix A, along with the maximum *a posteriori* formulation.

## 5 Experiments

Our experiments underscore several key findings. First, we demonstrate that a single generative model can be trained to operate across a wide range of latent dimensionalities, enabling the same model to be meaningfully tuned at inference time rather than retrained for each complexity level (Figure 2). Second, in the denoising setting, we empirically validate the theoretical findings of Section 4, which identify an intermediate optimal latent complexity in the presence of additive noise (Figures 11 and 13). Third, we show that the observed non-monotonic dependence on latent dimensionality generalizes across multiple inverse problems, datasets, and generative architectures, including latent diffusion models, variational autoencoders, and normalizing flows (Figure 3). Finally, we demonstrate that incorporating tunability can complement recent advances in inversion algorithms, with especially large gains for methods that explicitly optimize for measurement consistency, such as ReSample [37], flow-based MAP estimation [42], and more mixed effects for gradient-guided posterior-sampling methods [43, 40, 34] (Tables 1 to 3). We include both positive and negative paired comparisons in these broader solver experiments to show where tunability is most useful, rather than to suggest that it should improve every inversion method uniformly. Training, evaluation, inverse problem implementations, configurations, and environment instructions are available in our public GitHub repository.

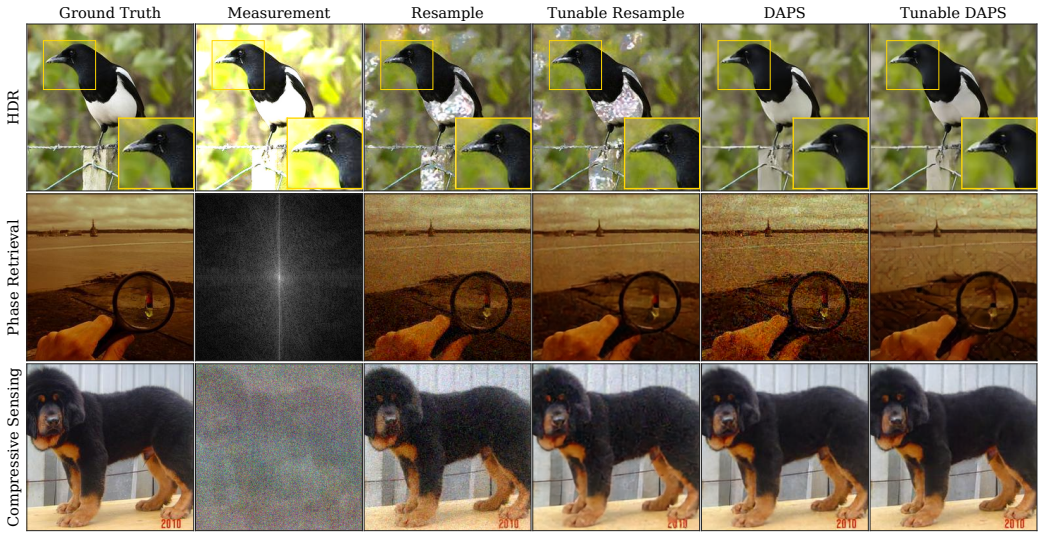



Figure 4: **Qualitative results on ImageNet across three inverse problems.** Rows: HDR (clipping with scale factor 2), phase retrieval ($m/n = 15\%$), and compressed sensing ($m/n = 10\%$). Columns: ground truth, measurement ($\mathbf{A}^\top \mathbf{y}$ for CS and phase retrieval; clipped input for HDR), and reconstructions from ReSample [37] and LatentDAPS [40] alongside their tunable counterparts. Insets show enlarged views of the regions highlighted in yellow.

**Datasets and Metrics.** We evaluate on five datasets spanning different resolutions: CelebA [44], CelebA-HQ [45], and MS COCO [46] at $64 \times 64 \times 3$, and ImageNet [47] and FFHQ [48] at $256 \times 256 \times 3$. For generative quality, we report the Fréchet Inception Distance (FID $\downarrow$) computed on 50k samples. For inverse problems, we measure reconstruction fidelity using Peak Signal-to-Noise Ratio (PSNR $\uparrow$) and perceptual quality using LPIPS ($\downarrow$). Unless otherwise specified, reconstruction results are reported on a test set of 100 images. For the experiments reported in Tables 1 and 2, the latent dimensionality $k$ was selected using a held-out validation set of 12 images; Figure 5 shows a ground-truth-free alternative using synthetic images.

**Baselines.** We compare against representatives of three families of generative priors for inverse problems. Flow matching: Flow CHEF [43] and Flow ICTM [42]. Latent diffusion: PSLD [34], LatentDAPS [40], and ReSample [37]. Normalizing flows: NF [4], which performs MAP estimation in the flow's latent space. For each baseline, we additionally construct a *tunable variant* that replaces the fixed latent prior with a tunable one while leaving the inversion procedure unchanged, isolating the effect of prior flexibility.

**Implementation Details.** We build upon the Diffusers package [49] for both the denoising U-Net and VAE architectures. All VAE models employ continuous latent spaces without quantized codebooks. For each dataset, the VAE was trained on the same data used for subsequent diffusion model training, with a held-out validation set used to evaluate LPIPS for model selection. For the LDM, we employ a U-Net in half-precision, following the architectural specifications of [21], and optimize using AdamW [50]. During training, we maintained a held-out set to evaluate FID, selecting the model with the lowest FID score for inverse problem applications. All models were trained on a single H200 GPU, and all inverse problems were solved on a single A100 (40 GB). Additional implementation details are provided in Section B.2.

**Inverse Problem Details** For Figure 3, we evaluated four forward operators. All experiments were done on $64 \times 64 \times 3$ images (FFHQ downsampled to $64 \times 64$). Compressed sensing used a random Gaussian operator $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with i.i.d. $\mathcal{N}(0, 1/m)$ entries, where $m = 1228$ (approximately $10\%$ of $n = 12288$). Denoising was modeled as additive Gaussian noise $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n)$ with $\sigma = 0.2$. Inpainting employed a binary mask $\boldsymbol{M}$ with approximately $80\%$ missing pixels. Phase retrieval used phaseless Gaussian measurements $|\boldsymbol{A}\boldsymbol{x}|$, using a Gaussian $\boldsymbol{A}$ with measurement ratio $m/n = 0.25$ ($25\%$). All inverse problems were solved using Algorithm 2 for latent diffusion priors.

Table 1: **Quantitative comparison on FFHQ across five inverse problems.** Mean PSNR ↑ and LPIPS ↓ are reported over 100 test images. Tunable priors are compared against their fixed-complexity counterparts using the same inversion method; bold indicates the better result within each matched pair, and dashes denote settings not evaluated.

| Method | Compressed Sensing | | Inpainting | | Super Resolution | | Phase Retrieval | | HDR | |
|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ |
| Flow CHEF [43] | **26.31** | **0.36** | 24.53 | 0.42 | 24.07 | 0.52 | — | | — | |
| Tunable Flow CHEF (Ours) | 26.24 | **0.36** | **24.76** | **0.39** | **24.41** | **0.49** | — | | — | |
| Flow ICTM [42] | 22.98 | 0.57 | 21.62 | 0.58 | 20.91 | 0.59 | — | | — | |
| Tunable Flow ICTM (Ours) | **26.13** | **0.41** | **22.60** | **0.51** | **21.38** | **0.56** | — | | — | |
| LatentDAPS [40] | **28.5** | **0.27** | 24.21 | 0.35 | **26.68** | **0.27** | **29.32** | **0.23** | **19.79** | **0.33** |
| Tunable LatentDAPS (Ours) | 28.31 | 0.29 | **24.50** | **0.34** | 26.61 | **0.27** | **29.32** | **0.23** | 19.36 | 0.34 |
| PSLD [34] | 24.92 | 0.37 | 17.12 | 0.59 | 21.95 | 0.46 | — | | — | |
| Tunable PSLD (Ours) | **25.41** | **0.34** | **17.55** | **0.57** | **22.36** | **0.43** | — | | — | |
| ReSample [37] | 24.49 | 0.51 | 22.26 | 0.48 | 22.43 | 0.44 | 22.68 | 0.58 | 20.53 | 0.38 |
| Tunable ReSample (Ours) | **28.43** | **0.28** | **24.84** | **0.39** | **24.53** | **0.39** | **28.51** | **0.29** | **20.70** | **0.36** |

Table 2: **Quantitative comparison on ImageNet across five inverse problems.** Mean PSNR ↑ and LPIPS ↓ are reported over 100 test images. Tunable variants are evaluated against the corresponding fixed-complexity baselines, isolating the effect of latent-space tunability; bold indicates the better result within each matched pair, and dashes denote settings not evaluated.

| Method | Compressed Sensing | | Inpainting | | Super Resolution | | Phase Retrieval | | HDR | |
|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ |
| LatentDAPS [40] | 28.01 | **0.24** | 22.65 | 0.48 | 21.15 | 0.54 | 23.67 | 0.43 | 18.84 | 0.32 |
| Tunable LatentDAPS (Ours) | **28.12** | **0.24** | **22.99** | **0.42** | **21.17** | **0.50** | **23.96** | **0.38** | **18.96** | **0.31** |
| PSLD [34] | 18.61 | 0.61 | 18.82 | 0.63 | 18.56 | 0.64 | — | | — | |
| Tunable PSLD (Ours) | **19.39** | **0.60** | **19.25** | **0.61** | **19.34** | **0.61** | — | | — | |
| ReSample [37] | 23.74 | 0.47 | 19.55 | 0.49 | 18.82 | 0.55 | 23.00 | 0.49 | 20.62 | 0.30 |
| Tunable ReSample (Ours) | **25.86** | **0.36** | **22.19** | **0.43** | **21.62** | **0.47** | **26.49** | **0.33** | **20.71** | **0.29** |

In Tables 1 and 2, we compare our approach to a contemporary state-of-the-art (SOTA) diffusion prior and show that our framework can also enhance other SOTA priors on certain inverse problems. We deliberately include settings in which our method does *not* improve reconstruction error, to make clear that it is not intended as a universal one-shot solver.

We consider the following measurement operators: compressed sensing with measurement ratio $m/n = 10\%$; inpainting with $85\%$ of pixels missing uniformly at random; Gaussian deblurring with a $9 \times 9$ kernel of standard deviation 3; bicubic $4\times$ super-resolution; and HDR, implemented as clipping with scale factor 2. For phase retrieval, the forward map applies a Rademacher sign flip followed by circular convolution with a randomly drawn Gaussian filter, after which the signal is subsampled to $m$ measurements; we use $m/n = 15\%$ on ImageNet and $m/n = 12.5\%$ on FFHQ. Across all settings in Tables 1 and 2, we add Gaussian measurement noise with standard deviation $\sigma = 0.05$.

**Selecting the latent dimensionality $k$.** Tunability addresses a practical question: how should a practitioner select $k$ when the ground truth is unknown? In the obvious setting, when validation data is available one could use that. For both FFHQ and ImageNet, we generate 25 images at full latent dimensionality and apply known forward operators for compressed sensing, phase retrieval and inpainting with random missing pixels without measurement noise. We reconstruct these images across a range of $k$ values using ReSample [37] and compare reconstruction performance with 25 real images, a held-out set separate from both the 12 validation images and the 100 test images (Figure 5). Across both datasets and all three inverse problems, the $k$ with the lowest mean LPIPS on synthetic images matches the best $k$ for real images in four of the six settings, and in the other two it is within $0.0013$ LPIPS of the best real-image result. These results suggest that synthetic images can be used

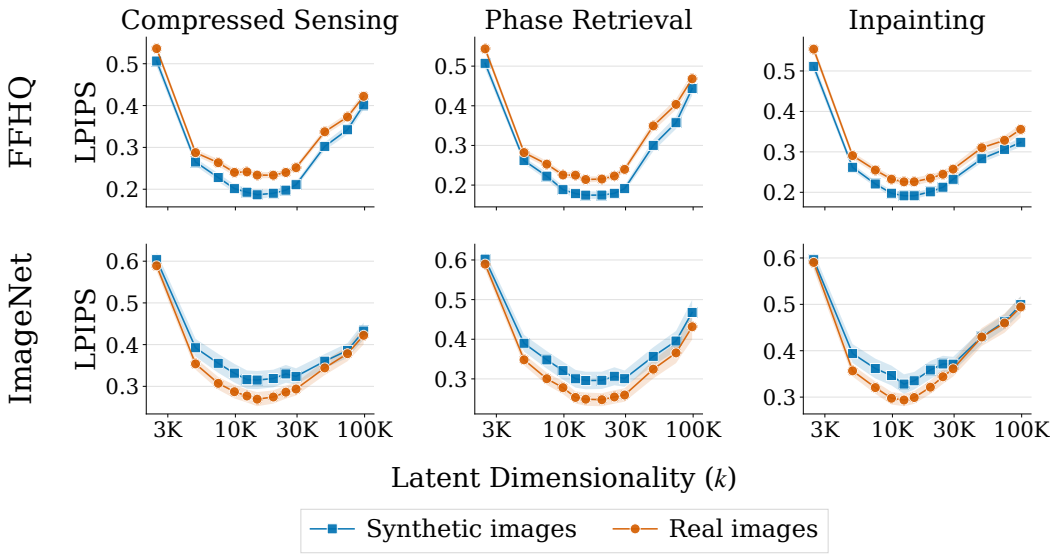



Figure 5: **Synthetic images provide useful calibration for selecting $k$.** Mean LPIPS over 25 synthetic and 25 real images on FFHQ (top) and ImageNet (bottom), using ReSample. Measurement ratios $m/n$: compressed sensing $12.5\%$ (FFHQ) and $15\%$ (ImageNet); phase retrieval $15\%$ (FFHQ) and $20\%$ (ImageNet). Inpainting removes $80\%$ of pixels at random. No measurement noise is added in any setting. Shading shows $\pm 1$ standard error.

to select $k$, and that even when the minima of the synthetic and real curves do not align, the loss in reconstruction quality is negligible.

## 6 Conclusion and Limitations

Prior work on inverse problems with generative priors has primarily focused on fixed-complexity models, with most effort directed at inversion algorithms. In this work, we study latent tunable priors as a complementary axis of improvement: rather than changing the inversion procedure, we vary the effective latent dimensionality of the prior, giving practitioners an additional degree of freedom with minimal training overhead and no increase in the number of model parameters. Using nested dropout as an existing mechanism for learning ordered latent representations, we demonstrate that tunability can improve reconstruction error across several model classes and inverse problems, with intermediate latent dimensions often outperforming both low- and high-complexity extremes. Our goal is not to introduce nested dropout as a new method, but to use it to test the hypothesis that controlling latent dimensionality can meaningfully improve inverse-problem performance. The gains are largest for inversion methods that solve a measurement-consistency minimization problem at each step: ReSample [37], flow-based MAP [42], and Algorithm 2. LatentDAPS [40] shows less consistent gains.

Several limitations point to important directions for future work. Our theory currently covers only linear generators under additive Gaussian noise; its value is mainly in suggesting that analogous criteria for selecting the optimal latent dimension $k$ may be possible for a richer model class. We focus on latent priors, where complexity is naturally controlled by the number of retained coordinates, while extending tunability to pixel-space architectures would require a different mechanism. We also treat the latent dimensionality $k$ as a solver hyperparameter, selecting a single value for each dataset, inverse problem, and noise level, either from a small held-out validation set with ground truth or, when real data is unavailable, from synthetic images generated by the prior itself. Neither uses ground truth at test time, but both assume that the validation or synthetic images are representative of the test data. Developing measurement-only or instance-adaptive rules for choosing $k$ remains an open problem.

## Acknowledgments

SG and PH acknowledge support from NSF Awards DMS-1848087 and DMS-2022205. For parts of this project, VC was generously supported by an NSF Foundations of Data Science Institute fellowship at Northeastern University and MIT, and a Hellman's Fellowship at UC Santa Cruz.

The authors declare no competing interests.

## References

[1] Jonathan Scarlett, Reinhard Heckel, Miguel R. D. Rodrigues, Paul Hand, and Yonina C. Eldar. Theoretical perspectives on deep learning methods in inverse problems. *IEEE Journal on Selected Areas in Information Theory*, 3(3):433–453, 2022. doi: 10.1109/JSAIT.2023.3241123.

[2] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR, 2017.

[3] Paul Hand, Oscar Leong, and Vlad Voroninski. Phase retrieval under a generative prior. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/1bc2029a8851ad344a8d503930dfd7f7-Paper.pdf.

[4] Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pages 399–409. PMLR, 2020.

[5] Giannis Daras, Joseph Dean, Ajil Jalal, and Alex Dimakis. Intermediate layer optimization for inverse problems using deep generative models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2421–2432. PMLR, 2021. URL http://proceedings.mlr.press/v139/daras21a.html.

[6] Giannis Daras, Yuval Dagan, Alex Dimakis, and Constantinos Daskalakis. Score-guided intermediate level optimization: Fast Langevin mixing for inverse problems. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4722–4753. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/daras22a.html.

[7] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vaRCHVj0uGI.

[8] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. PULSE: self-supervised photo upsampling via latent space exploration of generative models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2434–2442. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00251. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Menon_PULSE_Self-Supervised_Photo_Upsampling_via_Latent_Space_Exploration_of_Generative_CVPR_2020_paper.html.

[9] Jorio Cocola, Paul Hand, and Vlad Voroninski. Nonasymptotic guarantees for spiked matrix recovery with generative priors. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15185–15197. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ad62cfd33e3870262d6bf5331c1f13b0-Paper.pdf.

[10] Paul Hand and Babhru Joshi. Global guarantees for blind demodulation with generative priors. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates,

Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/598a90004bace6540f0e2230bdc47c09-Paper.pdf.

[11] Jay Whang, Qi Lei, and Alex Dimakis. Solving inverse problems with a flow-based noise model. In *International Conference on Machine Learning*, pages 11146–11157. PMLR, 2021.

[12] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJed6j0cKX.

[13] Tianci Liu, Tong Yang, Quan Zhang, and Qi Lei. Optimization for amortized inverse problems. In *International Conference on Machine Learning*, pages 22289–22319. PMLR, 2023.

[14] Xiangming Meng and Yoshiyuki Kabashima. Quantized compressed sensing with score-based generative models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OOWLRfAI_V_.

[15] Ajil Jalal, Sushrut Karmalkar, Alex Dimakis, and Eric Price. Instance-optimal compressed sensing via posterior sampling. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4709–4720. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jalal21a.html.

[16] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tplXNcHZs1.

[17] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

[18] Brendan Leigh Ross and Jesse C Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=DqU-rIHy4Eh.

[19] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[20] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

[21] 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 (CVPR)*, pages 10684–10695, June 2022.

[22] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014.

[23] Artur Bekasov and Iain Murray. Ordering dimensions with nested dropout normalizing flows, 2020.

[24] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, June 2021.

[25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[26] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

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

[28] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

[29] Xin Li, Yulin Ren, Xin Jin, Cuiling Lan, Xingrui Wang, Wenjun Zeng, Xinchao Wang, and Zhibo Chen. Diffusion models for image restoration and enhancement: A comprehensive survey, 2025. URL https://arxiv.org/abs/2308.09388.

[30] Hengkang Wang, Xu Zhang, Taihui Li, Yuxiang Wan, Tiancong Chen, and Ju Sun. DMPlug: A plug-in method for solving inverse problems with diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=81IFFsfQUj.

[31] Bahjat Kawar, Gregory Vaksman, and Michael Elad. SNIPS: Solving noisy inverse problems stochastically. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=9xPJ7cZ4ntc.

[32] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=mRieQgMtNTQ.

[33] Litu Rout, Yujia Chen, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Beyond first-order tweedie: Solving inverse problems using latent diffusion. In *CVPR*, pages 9472–9481, 2024. URL https://doi.org/10.1109/CVPR52733.2024.00905.

[34] Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=XKBFdYwfRo.

[35] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011. doi: 10.1198/jasa.2011.tm11181. URL https://doi.org/10.1198/jasa.2011.tm11181. PMID: 22505788.

[36] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

[37] Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=j8hdRqOUhN.

[38] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11410–11420, June 2022.

[39] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

[40] Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving diffusion inverse problem solving with decoupled noise annealing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20895–20905, 2025.

[41] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[42] Yasi Zhang, Peiyu Yu, Yaxuan Zhu, Yingshan Chang, Feng Gao, Ying Nian Wu, and Oscar Leong. Flow priors for linear inverse problems via iterative corrupted trajectory matching. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 57389–57417. Curran Associates, Inc., 2024. doi: 10.52202/079017-1829. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/698570ae5ec88e07e9f4547f831b4593-Paper-Conference.pdf.

[43] Maitreya Patel, Song Wen, Dimitris N. Metaxas, and Yezhou Yang. Steering rectified flow models in the vector field for controlled image generation. *arXiv preprint arXiv:2412.00100*, 2024.

[44] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[45] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[46] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

[47] 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.

[48] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[49] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[50] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[51] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[52] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[53] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

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

[55] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[56] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, Spain, 2011.

# A Proofs

In Section 4, we present the maximum-likelihood setting in the main text because it is conceptually simpler. Here, we give the corresponding maximum *a posteriori* (MAP) formulation, of which maximum likelihood is a special case.

We use the family $\boldsymbol{G_k} = \boldsymbol{U}_k \boldsymbol{\Sigma_k} \boldsymbol{V}_k$ defined in Section 4, with $\boldsymbol{\Sigma_k} = \mathrm{diag}(s_1, \dots, s_k) \in \mathbb{R}^{k \times k}$ and $\boldsymbol{G}_n = \boldsymbol{G}$. Each $\boldsymbol{G}_k$ induces a probability distribution over $\mathbb{R}^n$ by

$$\boldsymbol{x} = \boldsymbol{G}_k \boldsymbol{z}, \quad \boldsymbol{z} = (\boldsymbol{z}^k, 0) \in \mathbb{R}^n, \quad \boldsymbol{z}^k \sim \mathcal{N}(0, \boldsymbol{I}_k).$$

We denote this distribution by $p_{\boldsymbol{G}_k}$ and observe that $p_{\boldsymbol{G}_k} = \mathcal{N}(0, \boldsymbol{G}_k \boldsymbol{G}_k^T)$. The parameter $k$ controls the complexity of the modeled signal class.

We consider the following denoising problem. Let $\boldsymbol{y} = \boldsymbol{x}_0 + \boldsymbol{\eta}$, where $\boldsymbol{x}_0 \sim p_{\boldsymbol{G_n}}$ and $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n)$ are independent. Our goal is to recover $\boldsymbol{x}_0$ given $\boldsymbol{y}$ and $\boldsymbol{G}_n$. For a given $k$, we consider a maximum *a posteriori* estimate of $\boldsymbol{x}_0$ under the signal prior $p_{\boldsymbol{G}_k}$. This leads to the following optimization problem over the latent space:

$$\hat{\boldsymbol{z}}_\gamma(k) \in \arg\min_{\boldsymbol{z} \in \mathbb{R}^n} \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{G_k} \boldsymbol{z} \right\|^2 + \frac{\gamma}{2} \|\boldsymbol{z}\|^2. \tag{11}$$

Here, the estimated signal is $\hat{\boldsymbol{x}}_\gamma(k) = \boldsymbol{G}_k \hat{\boldsymbol{z}}_\gamma(k)$, and $\gamma \geq 0$ controls the strength of the latent prior. The case $\gamma = 0$ corresponds to the maximum-likelihood estimate (MLE), while $\gamma = \sigma^2$ corresponds to the standard MAP estimator under the Gaussian latent prior. Since $\gamma$ is often treated as a hyperparameter in practice, we study the behavior of Eq. (11) for all $\gamma \geq 0$.

The following theorem gives an exact expression for the mean-squared reconstruction error of this estimator.

**Theorem A.1.** *Suppose we have a family $\{\boldsymbol{G_k}\}_{k=1}^n$ of generative models as defined above, with $p_{\boldsymbol{G_k}} = \mathcal{N}(0, \boldsymbol{G_k} \boldsymbol{G_k}^T)$. Let $\boldsymbol{G}_n \in \mathbb{R}^{n \times n}$ have singular values $s_1 \geq s_2 \geq \cdots \geq s_n > 0$, and suppose $\boldsymbol{x}_0 \sim p_{\boldsymbol{G}_n}$ and $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n)$ are independent. Then the estimator in Eq. (11) satisfies*

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\eta}} \left[ \|\hat{\boldsymbol{x}}_\gamma(k) - \boldsymbol{x}_0\|^2 \right] = \sum_{i=1}^{k} \frac{s_i^2 \left( s_i^2 \sigma^2 + \gamma^2 \right)}{(s_i^2 + \gamma)^2} + \sum_{j=k+1}^{n} s_j^2. \tag{12}$$

The exact expression in Theorem A.1 makes it possible to read off the optimal value of the tunable complexity parameter $k$, as established in the following corollary.

**Corollary A.2.** *Under the assumptions of Theorem A.1, suppose $0 \leq \gamma \leq \sigma^2/2$ and $s_1 \geq \sigma$. Then*

$$k^\star = \max \left\{ k \in [n] \mid s_k \geq \sqrt{\sigma^2 - 2\gamma} \right\}$$

*is the largest minimizer of the expected squared reconstruction error over $k \in [n]$.*

This corollary shows that, for sufficiently small $\gamma$ and sufficiently large noise, the optimal complexity parameter can be strictly smaller than the full signal dimensionality. In the MLE case $\gamma = 0$, an intermediate signal complexity is optimal whenever the noise level exceeds the smallest singular value of the linear generator $\boldsymbol{G}$. More generally, as the noise level increases, the optimal value of the tunable complexity parameter decreases. Setting $\gamma = 0$ in Theorems A.1 and A.2 recovers Theorem 4.1 and Theorem 4.2.

*Proof of Theorem A.1.*
Without loss of generality, we take $\boldsymbol{G}_k$ to be a diagonal matrix. This follows because of invariance of Euclidean norms with respect to orthogonal transformations and because of the rotational invariance of the random variable $\boldsymbol{z_0}$. In these singular-vector coordinates, when solving Eq. (11), we may take

$$\boldsymbol{z} = (\boldsymbol{z}^k, 0) \in \mathbb{R}^n, \qquad \boldsymbol{z}^k \in \mathbb{R}^k,$$

since setting the remaining coordinates to zero leaves $\boldsymbol{G_k z}$ unchanged and cannot increase the regularization penalty. Let $\mathcal{P}_k$ and $\mathcal{P}_k^\perp$ be the orthogonal projectors onto $\mathrm{span}(\{e_1, \dots, e_k\})$ and its orthogonal complement, respectively. Throughout this proof, we identify each projected vector with

its coordinates in the corresponding standard basis, so that $\mathcal{P}_k(\boldsymbol{x}) \in \mathbb{R}^k$ and $\mathcal{P}_k^{\perp}(\boldsymbol{x}) \in \mathbb{R}^{n-k}$; this identification preserves Euclidean norms. For a diagonal matrix $\Sigma = \text{diag}(s_1, s_2, \cdots, s_n) \in \mathbb{R}^{n\times n}$, let $\boldsymbol{S}_k \in \mathbb{R}^{k\times k}$ and $\boldsymbol{S}_k^{\perp} \in \mathbb{R}^{n-k\times n-k}$ be such that

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{S}_k & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_k^{\perp} \end{pmatrix}.$$

We want to find the expectation

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\hat{\boldsymbol{x}}_\gamma(k) - \boldsymbol{x}_0\|^2 = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\mathcal{P}_k(\hat{\boldsymbol{x}}_\gamma(k) - \boldsymbol{x}_0)\|^2 + \mathbb{E}_{\boldsymbol{x}_0}\|\mathcal{P}_k^{\perp}(\boldsymbol{x}_0)\|^2. \tag{13}$$

For the equation above we will compute the first term on the right-hand side first, then do the same for the second term on the right-hand side. The first term on the right-hand side breaks into a sum of two terms which can readily be computed, while the second term on the right-hand side follows from a lemma. Notice $\mathcal{P}_k(\hat{\boldsymbol{x}}_\gamma(k)) = \boldsymbol{S}_k\mathcal{P}_k(\hat{\boldsymbol{z}}_\gamma(k)) \in \mathbb{R}^k$ and the first $k$ coordinates of the solution of Eq. (11) are

$$\begin{aligned}\mathcal{P}_k(\hat{\boldsymbol{z}}_\gamma(k)) &= (\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\mathcal{P}_k(y) \\ &= (\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T(\boldsymbol{S}_k\mathcal{P}_k(z_0) + \mathcal{P}_k(\boldsymbol{\eta})).\end{aligned}$$

Thus,

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\mathcal{P}_k(\hat{\boldsymbol{x}}_\gamma(k) - \boldsymbol{x}_0)\|^2 &= \mathbb{E}_{\boldsymbol{z_0},\boldsymbol{\eta}}\|\boldsymbol{S}_k\mathcal{P}_k(\hat{\boldsymbol{z}}_\gamma(k)) - \boldsymbol{S}_k\mathcal{P}_k(\boldsymbol{z_0})\|^2 \\ &= \mathbb{E}_{\boldsymbol{z_0},\boldsymbol{\eta}}\|\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\mathcal{P}_k(y) - \boldsymbol{S}_k\mathcal{P}_k(\boldsymbol{z_0})\|^2 \\ &= \mathbb{E}_{\boldsymbol{z_0},\boldsymbol{\eta}}\|\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T(\boldsymbol{S}_k\mathcal{P}_k(z_0) + \mathcal{P}_k(\boldsymbol{\eta})) \\ &\quad - \boldsymbol{S}_k\mathcal{P}_k(\boldsymbol{z_0})\|^2 \\ &= \mathbb{E}_{\boldsymbol{z_0},\boldsymbol{\eta}}\|(\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\boldsymbol{S}_k - \boldsymbol{S}_k)\mathcal{P}_k(\boldsymbol{z_0}) \\ &\quad + \boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\mathcal{P}_k(\boldsymbol{\eta})\|^2 \end{aligned} \tag{14}$$

The last term above Eq. (14) can be decomposed

$$\mathbb{E}_{\boldsymbol{z_0}}\left\|\left(\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\boldsymbol{S}_k - \boldsymbol{S}_k\right)\mathcal{P}_k(\boldsymbol{z_0})\right\|^2 \tag{15}$$

$$+ 2\mathbb{E}_{\boldsymbol{z_0},\boldsymbol{\eta}}\left\langle\left(\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\boldsymbol{S}_k - \boldsymbol{S}_k\right)\mathcal{P}_k(\boldsymbol{z_0}), \boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\mathcal{P}_k(\boldsymbol{\eta})\right\rangle \tag{16}$$

$$+ \mathbb{E}_{\boldsymbol{\eta}}\left\|\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\mathcal{P}_k(\boldsymbol{\eta})\right\|^2 \tag{17}$$

The expectation of the inner product term Eq. (16) is zero because $\boldsymbol{z}_0$ and $\boldsymbol{\eta}$ are independent and have zero means. Furthermore, we apply Theorem A.3 to Eq. (15), yielding

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z_0}}\|(\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\boldsymbol{S}_k - \boldsymbol{S}_k)\mathcal{P}_k(\boldsymbol{z_0})\|^2 &= \|\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\boldsymbol{S}_k - \boldsymbol{S}_k\|_F^2 \\ &= \sum_{i=1}^{k}\left|\frac{s_i^3}{s_i^2+\gamma} - s_i\right|^2 \\ &= \sum_{i=1}^{k}\frac{s_i^2\gamma^2}{(s_i^2+\gamma)^2}.\end{aligned} \tag{18}$$

Then we apply Theorem A.4 to Eq. (17)

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\eta}}\|\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\mathcal{P}_k(\boldsymbol{\eta})\|^2 &= \text{Tr}(\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T\sigma^2\boldsymbol{I}_k\boldsymbol{S}_k(\boldsymbol{S}_k\boldsymbol{S}_k^T + \gamma\boldsymbol{I}_k)^{-1}\boldsymbol{S}_k^T) \\ &= \sum_{i=1}^{k}\frac{s_i^4\sigma^2}{(s_i^2+\gamma)^2}.\end{aligned} \tag{19}$$

This concludes the calculation of the first term on the right-hand side of Eq. (13). Now we apply Theorem A.3 to the second term on the right-hand side

$$\mathbb{E}_{\boldsymbol{x}_0}\|\mathcal{P}_k^{\perp}(\boldsymbol{x}_0)\|^2 = \mathbb{E}_{z_0}\|\boldsymbol{S}_k^{\perp}\mathcal{P}_k^{\perp}(z_0)\|^2 = \|\boldsymbol{S}_k^{\perp}\|_F^2 = \sum_{j=k+1}^{n} s_j^2. \tag{20}$$

Lastly, we combine Eqs. (18) to (20) and simplify

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\hat{\boldsymbol{x}}_\gamma(k) - \boldsymbol{x}_0\|^2 = \sum_{i=1}^{k} \frac{s_i^2(s_i^2\sigma^2+\gamma^2)}{(s_i^2+\gamma)^2} + \sum_{j=k+1}^{n} s_j^2.$$

For $\gamma = 0$, the solution above gives $\mathcal{P}_k(\hat{\boldsymbol{z}}_\gamma(k)) = \boldsymbol{S}_k^{-1}\mathcal{P}_k(\boldsymbol{y})$. Returning to the original signal coordinates,

$$\hat{\boldsymbol{x}}_\gamma(k) = \hat{\boldsymbol{x}}(k) = \boldsymbol{G}_k\hat{\boldsymbol{z}}(k) = \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{y}.$$

Each retained-coordinate term in the error formula then equals $\sigma^2$, giving

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\hat{\boldsymbol{x}}(k) - \boldsymbol{x}_0\|^2 = k\sigma^2 + \sum_{j=k+1}^{n} s_j^2,$$

which recovers Theorem 4.1. □

*Proof of Corollary A.2.*
For notational convenience, let $E(k) = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\eta}}\|\hat{\boldsymbol{x}}_\gamma(k) - \boldsymbol{x}_0\|^2$, and let

$$C = \{k \in [n] \mid s_k \geq \sqrt{\sigma^2 - 2\gamma}\}.$$

From our assumptions, $\sigma^2 - 2\gamma \geq 0$ and $s_1 \geq \sigma \geq \sqrt{\sigma^2-2\gamma}$, so $1 \in C$ and $k^\star = \max C$ is well defined. Since $s_1 \geq s_2 \geq \cdots \geq s_n$, we have $C = \{1, \ldots, k^\star\}$. For $k \in \{2, \ldots, n\}$, the condition $E(k-1) - E(k) \geq 0$ reads

$$\begin{aligned}
&\sum_{i=1}^{k-1} \frac{s_i^2(s_i^2\sigma^2+\gamma^2)}{(s_i^2+\gamma)^2} + \sum_{j=k}^{n} s_j^2 - \sum_{i=1}^{k} \frac{s_i^2(s_i^2\sigma^2+\gamma^2)}{(s_i^2+\gamma)^2} \\
&- \sum_{j=k+1}^{n} s_j^2 \geq 0 \\
&\iff \\
&s_k^2 - \frac{s_k^2(s_k^2\sigma^2+\gamma^2)}{(s_k^2+\gamma)^2} \geq 0 \\
&\iff \\
&s_k \geq \sqrt{\sigma^2-2\gamma},
\end{aligned}$$

so $E(k-1) \geq E(k)$ if and only if $k \in C$.

We first show that $E$ is nonincreasing on $C$. Since $s_i^2 \geq \sigma^2 - 2\gamma$ for all $i \in C$, it is clear that

$$s_i^4 + s_i^2(2\gamma - \sigma^2) \geq s_i^4 - s_i^2 \cdot s_i^2 = 0$$

is non-negative. From this, we deduce that $s_i^2\sigma^2 + \gamma^2 \leq (s_i^2+\gamma)^2$, hence

$$\Delta(i) \coloneqq 1 - \frac{s_i^2\sigma^2+\gamma^2}{(s_i^2+\gamma)^2} \geq 0$$

is also non-negative.

Let $P, Q \in C$ be such that $P \leq Q$. We will show $E(P) \geq E(Q)$ by showing the difference $E(P) - E(Q)$ is non-negative. It is easily computed by

$$E(P) - E(Q) = \sum_{i=P+1}^{Q} s_i^2\Delta(i).$$

The above shows that $\Delta(i)$ is non-negative for all $i$ in the range $P < i \leq Q$ since all such $i$ are elements of $C$. For $k^\star < i \leq n$, we have $s_i^2 < \sigma^2 - 2\gamma$, so $\Delta(i) < 0$ and $E(i) - E(i-1) = -s_i^2\Delta(i) > 0$. Thus $k^\star$ is the largest minimizer over $[n]$. □

**Lemma A.3.** *Let $\boldsymbol{x} \in \mathbb{R}^n$ satisfy $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_n)$. Then, for any matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, we have*

$$\mathbb{E}_{\boldsymbol{x}} \|\boldsymbol{M}\boldsymbol{x}\|^2 = \|\boldsymbol{M}\|_F^2.$$

*Proof of Lemma A.3.*
For matrices, let $\langle A, B \rangle_F = \text{Tr}(A^T B)$ denote the Frobenius inner product. Since $\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}\boldsymbol{x}^T] = \boldsymbol{I}_n$, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}} \|\boldsymbol{M}\boldsymbol{x}\|^2 &= \mathbb{E}_{\boldsymbol{x}} \langle \boldsymbol{M}\boldsymbol{x}, \boldsymbol{M}\boldsymbol{x} \rangle \\
&= \mathbb{E}_{\boldsymbol{x}} \langle \boldsymbol{M}^T\boldsymbol{M}, \boldsymbol{x}\boldsymbol{x}^T \rangle_F \\
&= \langle \boldsymbol{M}^T\boldsymbol{M}, \boldsymbol{I}_n \rangle_F \\
&= \|\boldsymbol{M}\|_F^2.
\end{aligned}$$

□

**Lemma A.4.** *Let $\boldsymbol{x} \in \mathbb{R}^n$ satisfy $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$. Then, for any matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, we have*

$$\mathbb{E}_{\boldsymbol{x}} \|\boldsymbol{M}\boldsymbol{x}\|^2 = \textit{Tr}(\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T).$$

*Proof of Lemma A.4.*
Let $\boldsymbol{Y} = \boldsymbol{M}\boldsymbol{x} \in \mathbb{R}^m$. Then $\boldsymbol{Y} \sim \mathcal{N}(0, \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T)$. Thus,

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}} \|\boldsymbol{M}\boldsymbol{x}\|^2 = \mathbb{E}_{\boldsymbol{Y}}[\boldsymbol{Y}^T\boldsymbol{Y}] \\
&= \sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{Y}}[\boldsymbol{Y}_i^2] = \sum_{i=1}^{m} \text{Var}(\boldsymbol{Y}_i) \\
&= \text{Tr}(\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T). \qquad (21)
\end{aligned}$$

□

# B Experimental Details and Additional Results

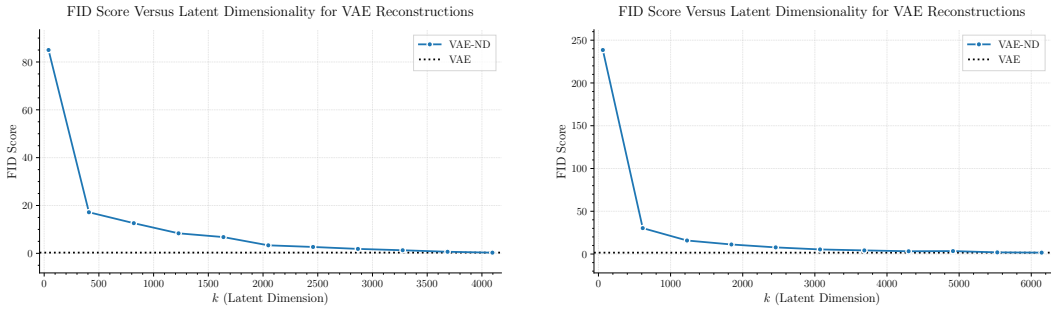
All FID scores reported in this appendix are computed with the `clean-fid` toolkit [38].

## B.1 VAE Training/Inversion

Our VAE follows the `AutoencoderKL` architecture from von Platen et al. [49]. Both VAEs are trained as described in the main text with the regularization weight $\lambda_{\text{drop}}$ fixed at $0.1$ and a nested-dropout success probability of $10^{-3}$, and use a VGG-based perceptual loss to encourage perceptual fidelity. For inversion, we use gradient descent on the latent vector, following the reference implementation accompanying Bora et al. [2].

**CelebA.** The images are $3 \times 64 \times 64$, and a latent dimensionality of $16 \times 16 \times 16$ is used, yielding a model with approximately $44$M parameters.

**MS COCO.** The images are $3 \times 64 \times 64$, and a latent dimensionality of $24 \times 16 \times 16$ is used.




(a) CelebA dataset

(b) MS COCO dataset

Figure 6: Reconstruction FID is plotted as a function of latent dimensionality $k$ for the tunable VAE on CelebA (a) and MS COCO (b). The vanilla VAE baseline (dotted line) operates only at full dimensionality. As $k$ increases, the tunable VAE approaches baseline performance while retaining the ability to reconstruct from lower-dimensional representations.

## B.2 LDM Training/Inversion

Here $(\cdot)_{\downarrow k}$ retains the first $k$ latent coordinates and zeros the rest, $\mathcal{D}$ is the autoencoder's decoder, and $\boldsymbol{\epsilon}_\theta$ is the trained noise-prediction network. We use $\bar{\alpha}_0 = 1$ and the previous-selected-timestep convention described in Section 3.2; $\sigma_t^{\text{DDPM}}$ is defined in Section 2.1. The positive correction scales $\tau_t$ are supplied separately from the sampling standard deviations, so the objective remains finite when the sampling variance is zero, including at the terminal step. Algorithm 2 is a simplified mathematical template with approximate optimization initialized from the reverse proposal.

Following von Platen et al. [49], all small-scale diffusion priors used in this section share a common training recipe: a `UNet2DModel` with approximately $200$M parameters, optimized with AdamW at learning rate $10^{-4}$ and batch size $128$ under the `DDPMScheduler`. We compute FID with the `DPMSolverMultistepScheduler` [51] at $100$ sampling steps, evaluating every $20{,}000$ iterations and retaining the three checkpoints with the lowest validation FID; the best-FID checkpoint is used for all subsequent inverse-problem evaluations.

**CelebA.** Training-set FID is computed over $50{,}000$ images and test-set FID over $10{,}000$ unseen images from the PyTorch CelebA split. CelebA images are $3 \times 64 \times 64$.

**CIFAR-10 [52] and MS COCO [46].** The same training recipe and FID protocol are used, applied to each dataset's standard train/test partitions. MS COCO images are $3 \times 64 \times 64$, and CIFAR-10 images are $3 \times 32 \times 32$.

For the inversion algorithms DPS [17] and LDPS/PSLD [34], we used the official codebases but adapted them to the Diffusers framework, which already provides pipelines. All hyperparameters were selected by grid search over the configuration files provided on GitHub, and we report results

**Algorithm 2** Tunable Posterior Sampling for Latent Diffusion (Simplified Template)

**Input:** $\boldsymbol{y}$, $\mathcal{A}$, $\mathcal{D}$, $\boldsymbol{\epsilon}_\theta$, steps $T$, schedule $\{\bar{\alpha}_t\}_{t=0}^{T}$, parameter $k$
Reverse sampler (DDIM or DDPM), DDIM $\eta \in [0,1]$, correction scales $\tau_t > 0$
**Output:** $\mathcal{D}(\boldsymbol{z}_0)$
$\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I}_d)$; $\boldsymbol{z}_T \leftarrow (\boldsymbol{z}_T)_{\downarrow k}$
**for** $t = T, \ldots, 1$ **do**
  $\hat{\boldsymbol{\epsilon}}_t \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$
  $\hat{\boldsymbol{z}}_0 \leftarrow (\boldsymbol{z}_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\boldsymbol{\epsilon}}_t)/\sqrt{\bar{\alpha}_t}$
  $\boldsymbol{\epsilon}'_t \sim \mathcal{N}(0, \boldsymbol{I}_d)$
  **if** reverse sampler is DDIM **then**
    $\sigma_t \leftarrow \eta\sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}(1-\alpha_t)}$
    $\boldsymbol{z}'_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{z}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\hat{\boldsymbol{\epsilon}}_t + \sigma_t\boldsymbol{\epsilon}'_t$
  **else** ▷ DDPM
    $\boldsymbol{z}'_{t-1} \leftarrow \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{z}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\boldsymbol{z}}_0 + \sigma_t^{\text{DDPM}}\boldsymbol{\epsilon}'_t$
  **end if**
  $\boldsymbol{z}_{t-1} \leftarrow \underset{\boldsymbol{z}\in\mathbb{R}^d}{\arg\min}\ \|\boldsymbol{y} - \mathcal{A}(\mathcal{D}(\boldsymbol{z}))\|_2^2 + \frac{\|\boldsymbol{z} - \boldsymbol{z}'_{t-1}\|_2^2}{2\tau_t^2}$
  *(solved approximately, initialized at $\boldsymbol{z}'_{t-1}$)*
  $\boldsymbol{z}_{t-1} \leftarrow (\boldsymbol{z}_{t-1})_{\downarrow k}$
**end for**
**return** $\mathcal{D}(\boldsymbol{z}_0)$




(a) Training set CelebA (b) Test set CelebA

Figure 7: FID is plotted as a function of latent dimensionality $k$ for models trained with different nested-dropout regularization weights $\lambda_{\text{drop}}$, on the training set (a) and the test set (b). The vanilla LDM baseline (dotted line) operates only at full dimensionality. As $k$ increases, the tunable models approach baseline performance while retaining the ability to generate from lower-dimensional representations. Training-set FID is computed over 50k images and test-set FID over 10k unseen images. FID scores computed with [38].

using the best-performing settings. Our public code repository provides training, evaluation, inverse-problem implementations, configurations, and environment instructions. In our experiments, DDIM sampling consistently yielded stronger reconstruction quality at intermediate latent dimensionalities compared to stochastic samplers, though a full theoretical understanding of this effect remains an open question. For Algorithm 2, the best-performing setup used the Adam optimizer [53] with the DDIM reverse equation, 3 optimization steps per update, 500 reverse steps, and DDIM stochasticity parameter $\eta = 0.1$. In these experiments, a small amount of DDIM stochasticity improved reconstruction: setting $\eta = 0$, *i. e.*, deterministic DDIM sampling, led to unstable or degraded reconstructions. This approach is particularly effective in our setting due to the relatively large latent space compared to the image space. The overall objective was to implement a straightforward inversion algorithm that can be readily applied across a wide range of problems.

**Tunable Priors on Various Datasets** Each problem is solved on 5 images randomly drawn from the held-out test set with a batch size of 12. The blue shaded region represents one standard deviation.

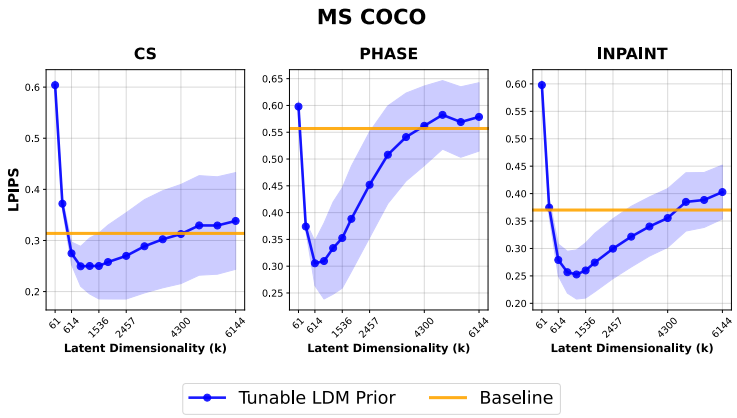



Figure 8: LPIPS (↓) versus latent dimensionality $k$ on MS COCO for compressed sensing, phase retrieval, and inpainting. The tunable LDM prior (blue; shaded region is one standard deviation) is compared with the fixed-complexity baseline (orange). The tunable prior attains lower LPIPS than the baseline at intermediate $k$.

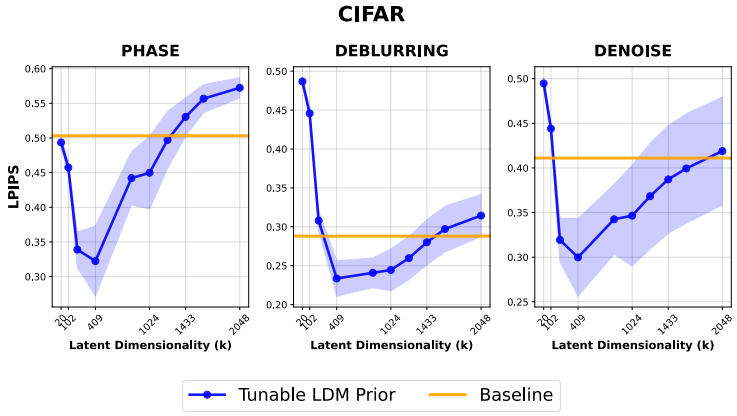



Figure 9: LPIPS (↓) versus latent dimensionality $k$ on CIFAR-10 for phase retrieval, deblurring, and denoising. The tunable prior attains lower LPIPS than the baseline at intermediate $k$.

Table 3: Quantitative results on CelebA-HQ ($3 \times 64 \times 64$ images) for compressed sensing, phase retrieval, random inpainting, and Gaussian deblurring, comparing each fixed-complexity prior (DPS, LDPS/PSLD, NF) with its tunable counterpart (Ours). Values are mean $\pm$ std over 100 images. **Bold** indicates best per column.

| | CS | | Phase Retrieval | | Inpainting (random) | | Deblur (Gauss.) | |
|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ | PSNR↑ | LPIPS↓ |
| DPS [17] | $25.65 \pm 2.67$ | $\mathbf{0.159 \pm 0.067}$ | $22.15 \pm 3.84$ | $0.267 \pm 0.061$ | $23.18 \pm 2.11$ | $0.146 \pm 0.042$ | $27.90 \pm 1.53$ | $0.093 \pm 0.019$ |
| Tunable LDM Prior (Ours) | $25.49 \pm 1.70$ | $0.179 \pm 0.045$ | $25.24 \pm 1.71$ | $\mathbf{0.168 \pm 0.046}$ | $25.30 \pm 1.76$ | $0.122 \pm 0.028$ | $\mathbf{29.52 \pm 1.11}$ | $\mathbf{0.085 \pm 0.024}$ |
| LDPS / PSLD [34] | $23.30 \pm 1.24$ | $0.248 \pm 0.063$ | $24.48 \pm 1.35$ | $0.169 \pm 0.038$ | $25.85 \pm 2.18$ | $0.136 \pm 0.036$ | $27.33 \pm 1.23$ | $0.165 \pm 0.037$ |
| Tunable LDPS / PSLD (Ours) | $24.69 \pm 1.37$ | $0.164 \pm 0.036$ | $21.95 \pm 1.31$ | $0.256 \pm 0.065$ | $\mathbf{26.75 \pm 1.71}$ | $\mathbf{0.095 \pm 0.028}$ | $28.50 \pm 1.05$ | $0.115 \pm 0.031$ |
| NF [4] | $21.31 \pm 1.94$ | $0.508 \pm 0.079$ | $20.67 \pm 2.32$ | $0.554 \pm 0.073$ | $20.07 \pm 2.42$ | $0.376 \pm 0.091$ | $21.43 \pm 2.11$ | $0.438 \pm 0.034$ |
| Tunable NF (Ours) | $\mathbf{27.16 \pm 1.56}$ | $0.238 \pm 0.058$ | $\mathbf{26.11 \pm 2.07}$ | $0.246 \pm 0.076$ | $23.24 \pm 2.43$ | $0.167 \pm 0.057$ | $24.85 \pm 2.21$ | $0.215 \pm 0.043$ |

**FFHQ and ImageNet Model Configurations** For the FFHQ and ImageNet experiments we train two diffusion U-Nets that operate in a shared $24 \times 64 \times 64$ latent space produced by a KL autoencoder ($256 \times 256$ RGB images, $4\times$ spatial downsampling). The autoencoder is trained for 100,000 steps at batch size 32 with a VGG-based perceptual loss, a small KL weight of $10^{-6}$, an adversarial discriminator (weight 0.1, ramped in over the first 300 steps), and a nested-dropout regularizer with strength $\lambda_{\text{drop}} = 0.1$. Both diffusion models are trained for 1000 DDPM timesteps with min-SNR-$\gamma$ weighting ($\gamma = 5$), AdamW at learning rate $10^{-4}$, gradient accumulation of 2, and a linear warmup

followed by cosine decay; we sample with the `DPMSolverMultistepScheduler` for FID evaluation and with DDIM for inverse problems.

**FFHQ.** The FFHQ U-Net has approximately 520M parameters and is used for *both* the tunable and fixed-complexity (baseline) prior; the two variants share an identical architecture and differ only in whether nested-dropout is applied to the latent during training. The network has four resolution stages ($64\to32\to16\to8$) with channel widths $(320, 640, 960, 1280)$, two residual blocks per stage, attention at every stage with head dimension 64, and a scale–shift time embedding. We train for 80,000 optimizer steps at a per-GPU batch size of 100 (effective batch 200 with gradient accumulation) with 10,000 warmup steps, achieving an FID of 8 on a held-out validation set of 5,000 images.

**ImageNet.** The ImageNet U-Net has approximately 720M parameters and is class-conditional over the 1000 ImageNet classes plus a learned null embedding to support classifier-free guidance [54]. At training time we replace the class label with the null token with probability 0.1; at sampling time we use a guidance scale of 4.0. The network has six resolution stages ($64\to32\to16\to8\to4\to2$) with channel widths $(320, 320, 640, 640, 1280, 1280)$; the top two stages use plain residual blocks while the bottom four add attention with head dimension 64, and each stage has two residual blocks. We train for 120,000 optimizer steps at a per-GPU batch size of 160 (effective batch 320 with accumulation) with 5,000 warmup steps, achieving an FID of 18 on a held-out validation set of 5,000 images. Both models are trained from scratch on a single NVIDIA H200, and inverse-problem inference is run on a single NVIDIA A100 (40 GB) or H200 GPU; the latent-dimensionality sweeps below used one H200 per job.

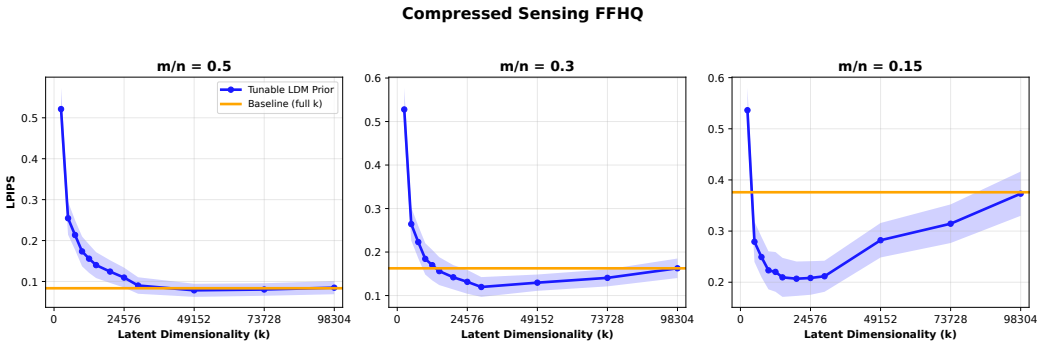



Figure 10: Compressed sensing on FFHQ ($256 \times 256$) with ReSample at measurement ratios $m/n = 0.5$, 0.3, and 0.15. LPIPS (↓) versus latent dimensionality $k$ for the tunable LDM prior (blue; shaded region is one standard deviation) and the same model at full $k$ (orange).

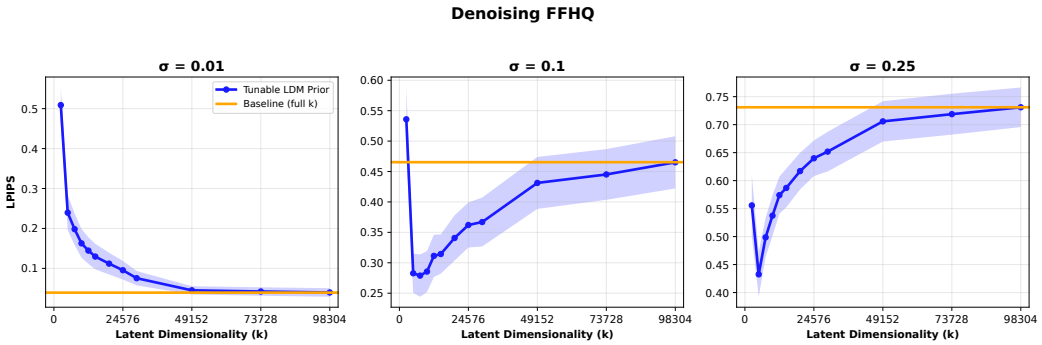



Figure 11: Denoising on FFHQ ($256 \times 256$) with ReSample at noise levels $\sigma = 0.01$, 0.1, and 0.25. LPIPS (↓) versus latent dimensionality $k$ for the tunable LDM prior (blue; shaded region is one standard deviation) and the same model at full $k$ (orange).

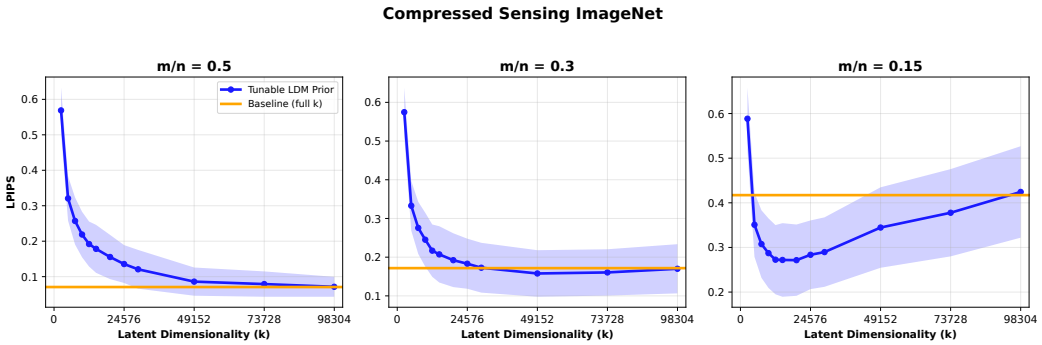



Figure 12: Compressed sensing on ImageNet ($256 \times 256$) with ReSample at measurement ratios $m/n = 0.5$, 0.3, and 0.15. LPIPS (↓) versus latent dimensionality $k$ for the tunable LDM prior (blue; shaded region is one standard deviation) and the same model at full $k$ (orange).

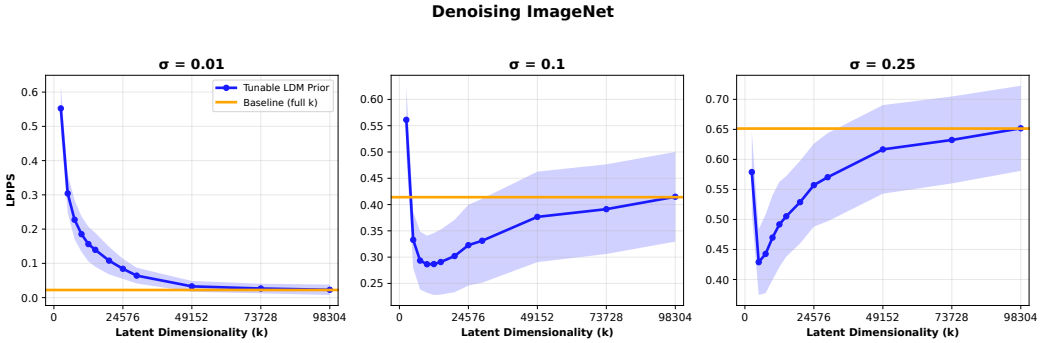



Figure 13: Denoising on ImageNet ($256 \times 256$) with ReSample at noise levels $\sigma = 0.01$, 0.1, and 0.25. LPIPS (↓) versus latent dimensionality $k$ for the tunable LDM prior (blue; shaded region is one standard deviation) and the same model at full $k$ (orange).

**Selecting $k$ when validation data are available.** We select $k$ as the latent dimensionality that achieves the lowest LPIPS reconstruction error on a validation set of 12 images.

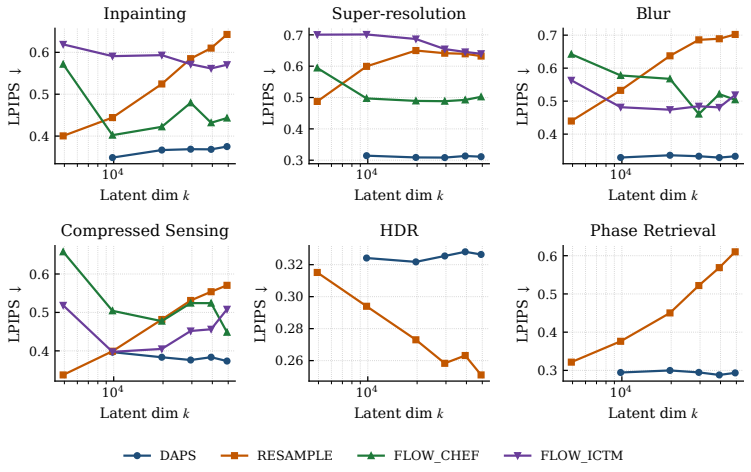



Figure 14: LPIPS (↓) is plotted against $k$ on the 12-image validation set for six inverse problems and four inversion methods (DAPS, ReSample, Flow-CHEF, Flow-ICTM). The best $k$ differs across methods and inverse problems, and the minimizer of each validation curve is the $k$ that would be selected for the test set.

## C Normalizing Flow Inverse Problems

**Naive Approach to Solving Tunable Prior** We trained a family of injective flows following Ross and Cresswell [18] with latent dimensionalities $\{16, 32, 64, 100, 112, 128, 200, 256, 384, 456\}$, each on MNIST ($N = 1024$) for 1,000 epochs using the authors official GitHub Repository, without parameter sharing. All models had the same number of parameters and were used as priors for compressed sensing with Gaussian random measurements $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and additive Gaussian noise $\boldsymbol{\eta}$ normalized to $\sqrt{\mathbb{E}\|\boldsymbol{\eta}\|^2} = 0.1$. These experimental details correspond to the results shown in Figure 1.

**Normalizing Flows** We assume the practitioner has trained a tunable normalizing flow with the following objective of Bekasov and Murray [23]. Our formulation is for latent-variable generative priors. For simplicity assume the measurements have the following form: the forward operator $\mathcal{A} = \boldsymbol{A} \in \mathbb{R}^{m \times n}$ and the additive noise term is i.i.d. $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_m)$, as in the compressed sensing case. Thus the measurements are generated by $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\eta}$, where $\boldsymbol{x} \in \mathbb{R}^n$ is an unknown signal.

Consider a fixed invertible neural network $G \colon \mathbb{R}^n \to \mathbb{R}^n$. Define for each $k \in [n] = \{1, \ldots, n\}$ the map

$$\begin{aligned} G_k \colon \mathbb{R}^k &\longrightarrow \mathbb{R}^n \\ z &\longmapsto G\left((\boldsymbol{z}, \quad 0)^\top\right). \end{aligned}$$

We note that $k$ is the parameter that governs the complexity of the prior, and $(\boldsymbol{z}, \quad 0)^\top \in \mathbb{R}^n$ is obtained from the latent representation $\boldsymbol{z} \in \mathbb{R}^k$. Then the signal representation is given by $\boldsymbol{x} = G((\boldsymbol{z}^T, 0))$ where $\boldsymbol{x} \in \mathbb{R}^n$ is as above.

We are interested in recovering a signal $\boldsymbol{x} \in \mathbb{R}^n$ given a set of noisy measurements $\boldsymbol{y} \in \mathbb{R}^m$. Our prior has a valid density over the entire signal space when $\boldsymbol{x} \in \text{Range}(G_k)$. A natural attempt to solve the given inverse problem is therefore maximum *a posteriori* estimation:

$$\begin{aligned} \hat{\boldsymbol{x}}_{\text{MAP}}(k) &:= \underset{\boldsymbol{x} \in \text{Range}(G_k)}{\arg\min} \; -\log p(\boldsymbol{x}|\boldsymbol{y}) \\ &= \underset{\boldsymbol{x} \in \text{Range}(G_k)}{\arg\min} \; -\log p(\boldsymbol{y}|\boldsymbol{x}) - \log p_{G_k}(\boldsymbol{x}) \\ &= \underset{\boldsymbol{x} \in \text{Range}(G_k)}{\arg\min} \; \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 - \sigma^2 \log p_{G_k}(\boldsymbol{x}) \end{aligned}$$

where $p_{G_k}$ is the density function on $\boldsymbol{x}$ induced by $G_k$, $\|\cdot\|$ is the Euclidean norm, and $\sigma$ is given by the model noise. Similar to previous work of [4, 11], we optimize over the latent space given by:

$$\hat{\boldsymbol{z}}_{\text{MAP}}(k) := \underset{\boldsymbol{z} \in \mathbb{R}^k}{\arg\min} \; \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{A}G\left((\boldsymbol{z}, \quad 0)^\top\right)\right\|^2 - \sigma^2 \log p_{G_k}\left(G\left((\boldsymbol{z}, \quad 0)^\top\right)\right)$$

The proposed optimization problem above is solved via gradient descent with the Adam optimizer [53] and is initialized at $\boldsymbol{z} = 0$. Depending on the specific inverse problem at hand the tunable parameter $k$ can be selected to change the dimensionality of the model, so the first $k$ elements of $\boldsymbol{z}$ will be optimized over and the rest will be set to zero. Our formulation relies on a density function $p_{G_k}$, and work by [4, 11] has shown that a smoothing parameter on a density function improves performance when solving the optimization problem above. We empirically observe the same phenomenon, and as a result, we replace $\sigma^2$ with a hyperparameter $\gamma$ and optimize the modified MAP estimate

$$\hat{\boldsymbol{z}}_{\text{MAP}}(k) := \underset{\boldsymbol{z} \in \mathbb{R}^k}{\arg\min} \; \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{A}G\left((\boldsymbol{z}, \quad 0)^\top\right)\right\|^2 - \gamma \log p_{G_k}\left(G\left((\boldsymbol{z}, \quad 0)^\top\right)\right).$$

For the VAE we follow the same formulation as [2], which is similar to the one above, but instead of having a tractable density, we approximate it with an $\ell_2$ penalty in the latent space.

**Generative Priors with Tunable Complexity on Various Architectures** In these experiments we want to demonstrate that generative priors with tunable complexities are not tied to one specific normalizing-flow architecture. We train three architectures: Rational Quadratic Spline Flow [20], GLOW [55], and a GLOW-inspired variant called GLOW-ADD that uses an additive coupling layer instead of an affine one. Each architecture is trained on the SVHN dataset [56] under matched hyperparameters to enable a controlled comparison.

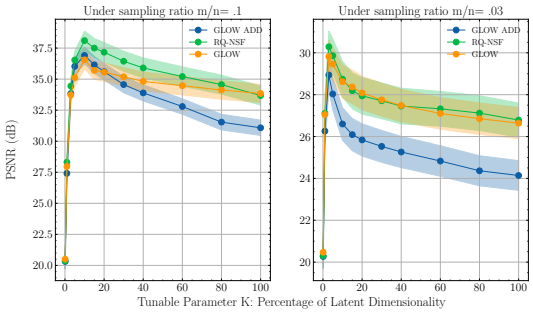



Figure 15: Reconstruction performance across three normalizing-flow architectures on SVHN; all benefit from a prior with tunable complexity.

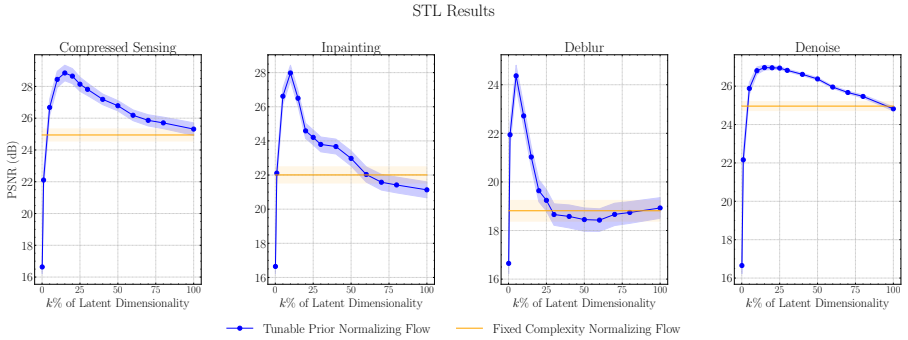



Figure 16: Performance of a generative prior with tunable complexity (Tunable Prior) and its fixed-complexity counterpart (Normalizing Flow) for compressed sensing, inpainting, deblurring, and denoising on the STL dataset. The tunable prior demonstrates a range of tunable parameters $k$ that lead to a better estimate of the target signal, as measured by PSNR, than its baseline.

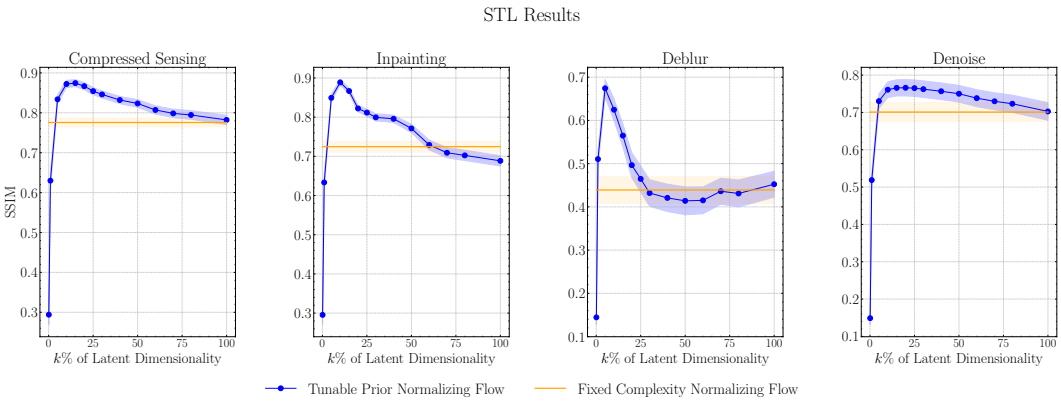



Figure 17: Performance of a generative prior with tunable complexity (Tunable Prior) and its fixed-complexity counterpart (Normalizing Flow) for compressed sensing, inpainting, deblurring, and denoising on the STL dataset. The tunable prior demonstrates a range of tunable parameters $k$, leading to a better estimate of the target signal, as measured by SSIM, than its baseline.

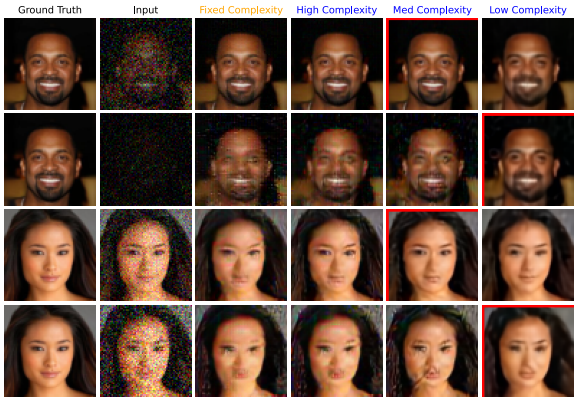



Figure 18: Results on inpainting and denoising across various measurement regimes for Normalizing Flow. The tuning parameter should be chosen based on the complexities of the given inverse problem. The red bounding box indicates which model achieved the lowest reconstruction error with respect to SSIM.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state three claims that map directly to results in the paper: (i) tunable latent priors can be built for VAEs, NFs, and LDMs (Appendix B.1, Appendix C, and Section 3.1, respectively); (ii) tunable priors empirically outperform fixed-complexity counterparts across compressed sensing, denoising, inpainting, and phase retrieval (Figure 3, Tables 1 and 2); and (iii) for linear denoising, an explicit MLE/MAP criterion characterizes the optimal complexity (Theorem 4.1, Corollary 4.2, Appendix A).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 explicitly discusses four limitations of the work: (i) the theoretical analysis (Theorem 4.1, Corollary 4.2) covers only linear generators under additive Gaussian noise, and extending analogous criteria for selecting the optimal latent dimension to richer model classes is left open; (ii) the tunability mechanism is specific to latent generative priors, where complexity is controlled by the number of retained coordinates, and extending it to pixel-space architectures would require a different mechanism; (iii) $k$ is selected using a held-out validation set with ground truth (or synthetic images), which is restrictive in practice, and developing measurement-only selection criteria remains an open problem; and (iv) experiments are limited to resolutions up to $256 \times 256$, leaving open how latent tunability scales to higher-resolution data and more complex reconstruction tasks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Theorem 4.1 and Corollary 4.2 are stated in Section 4 with assumptions. Full proofs, including the more general MAP formulation (Theorem A.1, Corollary A.2) and supporting lemmas (Lemmas A.3 and A.4), are in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Architectures, latent dimensionalities, parameter counts, optimizers, schedulers, training objectives, nested-dropout parameters, inversion algorithm details, and forward-operator specifications are all provided in Section 5 and Appendices B and C. Datasets are public.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The datasets are publicly available. Our public code repository provides training, evaluation, inverse-problem implementations, configurations, and environment instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details (optimizer, learning rate, batch size, schedulers, perceptual loss, nested-dropout parameters), validation procedure, inversion hyperparameters, and $k$-selection are all specified in Section 5 and Appendices B.1–B.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Figures 1, 8, 9, and 10–13 report shaded $\pm 1$ standard deviation regions or error bars across multiple test samples, and Table 3 reports mean $\pm$ standard deviation over 100 test images.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5 states that all models were trained on a single NVIDIA H200 GPU and all inverse problems were solved on a single NVIDIA A100 (40 GB) GPU; model sizes and training settings are given in Appendix B.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses only publicly available image datasets (CelebA, CelebA-HQ, FFHQ, ImageNet, MS COCO, CIFAR-10, SVHN, STL, MNIST) with established licenses, involves no human subjects or crowdsourcing, and conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper does not include a dedicated broader-impacts discussion, as the work is foundational methodology for inverse problems rather than a deployed application. We note here that improved generative priors for inverse problems can benefit medical imaging, scientific imaging, and computational photography by enabling higher-quality reconstructions from fewer or noisier measurements. As with all high-fidelity generative models, there is a risk that improved reconstruction quality could be misused to fabricate or alter images (e.g., deepfakes) or to denoise/enhance images obtained without consent. We do not release any model trained for identification or surveillance, and the face datasets used (CelebA, FFHQ) are widely studied benchmarks. Practitioners deploying tunable priors in sensitive domains should evaluate failure modes specific to their application.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The public repository provides research code for generative models trained on standard, publicly available image benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets are cited in the paper: CelebA, CelebA-HQ, FFHQ, ImageNet, MS COCO, CIFAR-10, SVHN, STL, MNIST, the Diffusers library, and pretrained baselines DPS, PSLD, ReSample [37], DAPS [40], CHEF, ICTM. CelebA and FFHQ are released for non-commercial research; COCO uses CC-BY 4.0; Diffusers uses Apache 2.0.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public code repository documents training, evaluation, inverse-problem implementations, configurations, and environment setup.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human-subjects research; no IRB approval is required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as an important, original, or non-standard component of the core methodology. Any use was limited to incidental writing/editing assistance, which per NeurIPS policy does not require declaration.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.