D-Flow: Differentiating through Flows for Controlled Generation

Heli Ben-Hamu [1 2 *] **Omri Puny** [2] **Itai Gat** [1] **Brian Karrer** [1] **Uriel Singer** [1] **Yaron Lipman** [1 2]

Abstract

Taming the generation outcome of state of the art Diffusion and Flow-Matching (FM) models without having to re-train a task-specific model unlocks a powerful tool for solving inverse problems, conditional generation, and controlled generation in general. In this work we introduce *D-Flow*, a simple framework for controlling the generation process by differentiating through the flow, optimizing for the source (noise) point. We motivate this framework by our key observation stating that for Diffusion/FM models trained with Gaussian probability paths, differentiating through the generation process projects gradient on the data manifold, implicitly injecting the prior into the optimization process. We validate our framework on linear and non-linear controlled generation problems including: image and audio inverse problems and conditional molecule generation reaching state of the art performance across all.

1. Introduction

Controlled generation from generative priors is of great interest in many domains. Various problems such as conditional generation, inverse problems, sample editing etc., can all be framed as a controlled generation problem. In this work we focus on controlled generation from diffusion/flow generative models (Song & Ermon, 2019; Ho et al., 2020; Lipman et al., 2023) as they are the current state-of-the-art generative approaches across different data modalities.

There are three main approaches for controlled generation from diffusion/flow models: (i) conditional training, where the model receives the condition as an additional input during training (Song et al., 2020; Dhariwal & Nichol, 2021; Ho & Salimans, 2022b), although performing very well this approach requires task specific training of a generative model which in cases may be prohibitive; (ii) training-free approaches that modify the generation process of a pre-trained model, adding additional guidance (Bar-Tal et al., 2023; Yu et al., 2023). The guidance is usually built upon strong assumptions on the generation process that can lead to errors in the generation and mostly limit the method to observations that are linear in the target (Kawar et al., 2022; Chung et al., 2022; Song et al., 2023a; Pokle et al., 2023); lastly, (iii) adopt a variational perspective, framing the controlled generation as an optimization problem (Graikos et al., 2023; Mardani et al., 2023; Wallace et al., 2023; Samuel et al., 2023b), requiring only a differentiable cost to enforce the control. This paper belongs to this third class.

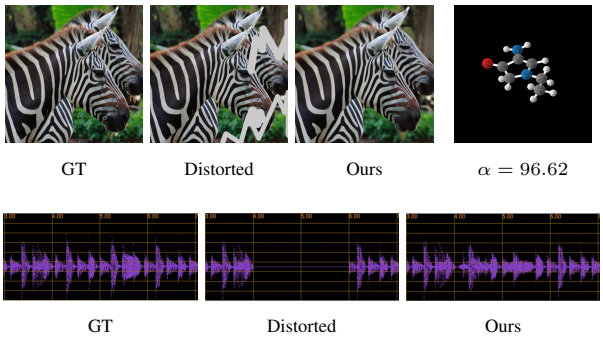


Figure 1: Free-form inpainting with a latent T2I FM model (Ground truth image is taken from the MS-COCO validation set), conditionally generated molecule and audio inpainting using D-Flow.

The goal of this paper is to introduce a framework for adding controlled generation to a pre-trained Diffusion or Flow-Matching (FM) model based on *differentiation through the ODE sampling process*. Our key observation is that for Diffusion/FM models trained with standard Gaussian probability paths, differentiating an arbitrary loss $\mathcal{L}(x)$ through the generation process of x with respect to the initial point, x_0, projects the gradient $\nabla_x \mathcal{L}$ onto the "data manifold", i.e., onto major data directions at x, implicitly injecting a valuable prior. Based on this observation we advocate a simple general algorithm that minimizes an arbitrary cost function $\mathcal{L}(x)$, representing the desired control, as a function of the source noise point x_0 used to generate x. That is,

$$\min_{x_0} \mathcal{L}(x). \tag{1}$$

[*]Work done as a research intern at FAIR, Meta. [1]Meta [2]Weizmann Institute of Science. Correspondence to: Heli Ben-Hamu <heli.benhamu@weizmann.ac.il>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Algorithm 1 D-Flow.

Require: cost $\mathcal{L}$, pre-trained flow model $u_t(x)$
Initialize $x_0^{(0)} = x_0$
for $i = 1, \dots, N$ **do**
  $x^{(i)}(1) \leftarrow \texttt{solve}(x_0^{(i)}, u_t)$
  $x_0^{(i+1)} \leftarrow \texttt{optimize_step}(x_0^{(i)}, \nabla_{x_0}\mathcal{L}(x^{(i)}(1)))$
end for
Return: $x^N(1)$

Differentiating through a generator of a GAN or a normalizing flow was proven generally useful for controlled generation (Bora et al., 2017; Asim et al., 2020; Whang et al., 2021) and counterfactual examples (Dombrowski et al., 2021; 2024). Recently, (Wallace et al., 2023; Samuel et al., 2023b) have been suggesting to differentiate through a discrete diffusion solver for the particular tasks of incorporating classifier guidance and generating rare concepts. In this paper we generalize this idea in two ways: (i) we consider general flow models trained with Gaussian probability paths, including Diffusion and Flow-Matching models; and (ii) we demonstrate, both theoretically and practically, that the inductive bias injected by differentiating through the flow is applicable to a much wider class of problems modeled by general cost functions.

We experiment with our method on a variety of settings and applications: Inverse problems on images using conditional ImageNet and text-2-image (T2I) generative priors, conditional molecule generation with QM9 unconditional generative priors, and audio inpainting and super-resolution with unconditional generative prior. In all application we were able to achieve state of the art performance without carefully tuning the algorithm across domains and applications. One drawback of our method is the relative long time for generation (usually $5 - 15$ minutes on ImageNet-128 on an NVidia V100 GPU) compared to some baselines, however the method's simplicity and its superior results can justify its usage and adaptation in many use cases. Furthermore, we believe there is great room for speed improvement.

To summarize, our contributions are:

- We formulate the controlled generation problem as a simple source point optimization problem using general flow generative models.
- We show that source point optimization of flows trained with Gaussian probability paths inject an implicit bias exhibiting a data-manifold projection behaviour to the cost function's gradient.
- We empirically show the generality and the effectiveness of the proposed approach for different domains.

2. Preliminaries

Flow models. Generative flow models, including Continuous Normalizing Flows (CNFs) (Chen et al., 2018; Lipman et al., 2023) and (deterministic sampling of) Diffusion Models (Song et al., 2020) generate samples $x(1) \in \mathbb{R}^d$ by first sampling from some source (noise) distribution $x(0) \sim p_0(x_0)$ and then solving an Ordinary Differential Equation (ODE),

$$\dot{x}(t) = u_t(x(t)), \tag{2}$$

from time $t = 0$ to time $t = 1$[1], using a predetermined velocity field $u : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$. We denote by p_1 the distribution and density function of $x(1)$ given $x(0) \sim p_0(x_0)$.

3. Controlled Generation via Source Point Optimization

Given a pre-trained (frozen) flow model, $u_t(x)$, represented by a neural network and some cost function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}_+$, our goal is to find likely samples x that provide low cost $\mathcal{L}(x)$ and are likely under the flow model's distribution p_1. We advocate a general framework formulating this problem as the following optimization problem

$$\min_{x_0} \quad \mathcal{L}(x(1)) \tag{3}$$

where in general $\mathcal{L}$ can also incorporate multiple costs including potentially a regularization term that can depend on x_0 and u,

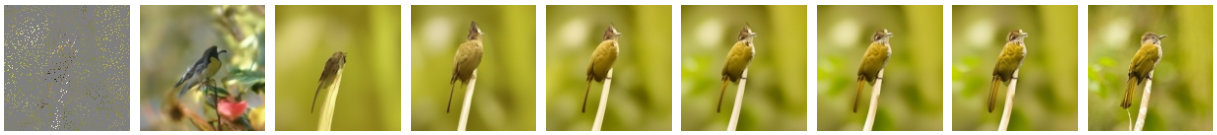
$$\tilde{\mathcal{L}}(x) = \mathcal{L}(x) + \mathcal{R}(x_0, u). \tag{4}$$

In this formulation, the sample $x(1)$ is constrained to be a solution of the ODE 2 with initial boundary condition $x(0) = x_0$, where x_0 is the only optimized quantity and $\mathcal{L}$ is the desired cost function. Optimizing equation 3 is done by computing the gradients of the loss w.r.t. the optimized variable x_0 as listed in Algorithm 1. We call this method *D-Flow*. To better understand the generality of this framework we next consider several instantiations of equation 3.

3.1. Cost Functions

Reversed sampling. First, consider the simple case where $\mathcal{L}(x) = \|x - y\|^2$. In this case, the solution of 3 will be the x_0 that that has an ODE trajectory that reaches y at $t = 1$, i.e., $x(1) = y$. Note that since (under some mild assumptions on $u_t(x)$) equation 2 defines a diffeomorphism $\mathbb{R}^d \to \mathbb{R}^d$, for an arbitrary $y \in \mathbb{R}^d$, there exists a unique solution $x_0 \in \mathbb{R}^d$ to equation 3.

[1] In this paper we use the convention of $t = 0$ corresponds to noise, and $t = 1$ to data.



Distorted | Initial $x(1)$ | step 2 | step 4 | step 6 | step 8 | step 10 | step 12 | GT

Figure 2: Intermediate $x(1)$ during optimization. Given a distorted image and randomly initialized x_0 defining the initial $x(1)$, our optimization travels close to the natural image manifold passing through in-distribution images on its way to the GT sample from the face-blurred ImageNet-128 validation set.

Inverse problems. In this case we have access to some known corruption function $H : \mathbb{R}^d \to \mathbb{R}^n$ and a corrupted sample from an unknown ground truth signal x_*,

$$y = H(x_*) + \epsilon, \tag{5}$$

where $\epsilon \sim \mathcal{N}(\epsilon)$ is an optional additive noise. The goal is to recover an x that produces y and the cost function is usually

$$\mathcal{L}(x) = \|H(x) - y\|^2 , \tag{6}$$

where the norm can be some arbitrary L_p norm or even a general loss $\ell(H(x), y)$ comparing $H(x)$ and y. Specific choices of the corruption function H can lead to common applications: *Image inpainting* corresponds to choosing the corruption function H to sub-sample known $n < d$ pixels out of d total pixels; *Image deblurring* corresponds to taking $H : \mathbb{R}^d \to \mathbb{R}^d$ to be a blurring function, e.g., a convolution with a blurring kernel; *Super-resolution* corresponds to $H : \mathbb{R}^d \to \mathbb{R}^{d/k}$ lowering the dimension by a factor of k.

Conditional sampling. Another important application is to guide the sampling process to satisfy some conditioning y. In this case we can take $\mathcal{L}(x)$ to encourage a classifier or some energy function to reach a particular class or energy y. For example, if $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ is some function and we would like to generate a sample from a certain level set $c \in \mathbb{R}$ we can use the loss

$$\mathcal{L}(x) = (\mathcal{F}(x) - c)^2 . \tag{7}$$

3.2. Initialization

The initialization of x_0 can have a great impact on the convergence of the optimization of equation 3. A natural choice will be to initialize x_0 with a sample from the source distribution $p_0(x_0)$. We find that for cases when an observed signal y provides a lot of information about the desired x, one can improve the convergence speed of the optimization. For example, in linear inverse problems on images, where the observed y has a strong prior on the structure of the image, it is beneficial to initialize x_0 with a blend of a sample from the source distribution and the backward solution of the ODE from $t = 1$ to $t = 0$ of y:

$$x_0 = \sqrt{\alpha} \cdot y(0) + \sqrt{1 - \alpha} \cdot z, \tag{8}$$

where $z \sim p_0(x_0)$ and $y(0) = y + \int_1^0 u(t, y(t))dt$.

3.3. Regularizations

The formulation in equation 3 allows including different regularizations $\mathcal{R}$ (equation 4) discussed next. Maybe the most intriguing of these regularizations, and the main point of this paper, is the *implicit regularization*, i.e., corresponding to $\mathcal{R} \equiv 0$, discussed last in what follows.

Regularizing the target $x(1)$. Maybe the most natural is incorporating the negative log likelihood (NLL) of the sample $x(1)$, i.e., $\mathcal{R} = -\log p_1(x(1))$ in equation 4. This prior can be incorporated by augmenting $x(t) \in \mathbb{R}^d$ with an extra coordinate $z \in \mathbb{R}$ and formulate equation 3 as

$$\min_{x_0} \quad \mathcal{L}(x(1)) - z(1) \tag{9a}$$

$$\text{s.t.} \quad \dot{x}(t) = u_t(x(t)), \qquad x(0) = x_0 \tag{9b}$$

$$\dot{z}(t) = -\text{div}\, u_t(x(t)), \; z(0) = \log p_0(x_0) \tag{9c}$$

Indeed, solving the ODE system defined by equations 9b and 9c for times $t \in [0, 1]$ provides $z(1) = \log p_1(x(1))$, see (Chen et al., 2018). However, aside from the extra complexity introduced by the divergence term in the ODE in equation 9c (see e.g., (Grathwohl et al., 2018) for ways to deal with this type of ODE) it is not clear whether likelihood is a good prior in deep generative models in high dimensions (Nalisnick et al., 2019); In Figure 3 we compare bits-per-dimension (BPD) of a test image of ImageNet-128 and a version of this image with a middle square masked with zeros, providing a more likely image according to our flow model trained on ImageNet.

BPD=2.02 BPD=1.84

Figure 3: BPD of two images in an ImageNet-128 model.

Regularizing the source $x(0) = x_0$. Another option is to regularize the source point $x(0) = x_0$. The first choice would again be to incorporate the NLL of the noise sample, i.e., $\mathcal{R} = -\log p_0(x_0)$, which for standard noise $p_0(x_0) = \mathcal{N}(x_0|0, I)$ would reduce to $\mathcal{R} = c + \frac{1}{2}\|x_0\|^2$, where c is a constant independent of x_0. This however, would attract x_0

towards the most likely all zero mean but far from most of the probability mass at norm $\sqrt{d}$.

Following (Samuel et al., 2023a) we instead prefer to make sure x_0 stays in the area where most mass of p_0 is concentrated and therefore use the χ^d distribution, which is defined as the probability distribution $p(r)$ of the random variable $r = \|x_0\|$ where $x_0 \sim \mathcal{N}(x_0|0, I)$ is again the standard normal distribution. The NLL of r in this case is

$$\mathcal{R} = -\log p(r) = c + (d-1)\log\|x_0\| - \frac{\|x_0\|^2}{2}, \quad (10)$$

where c is a constant independent of x_0.

Implicit regularization. Maybe the most interesting and potentially useful regularization in our formulation (equation 3) comes from the choice of optimizing the cost $\mathcal{L}(x(1))$ as a function of the source point $x(0) = x_0$. For standard diffusion/flow models that are trained to zero loss:

> Optimizing the cost $\mathcal{L}(x(1))$ with respect to x_0 follows the data distribution $p_1(x_1)$ by projecting the gradient $\nabla_{x(1)}\mathcal{L}(x(1))$ with the local data covariance matrix.

This is intuitively illustrated in Figure 4: while moving in direction of the gradient $\nabla_{x(1)}\mathcal{L}(x(1))$ generally moves away from the data distribution (in pink), differentiating w.r.t. $x(0)$ projects this gradient onto high variance data directions and consequently staying close to the data distribution. To exemplify this phenomena we show in Figure 2 optimization steps $x^{(0)}(1), x^{(2)}(1), x^{(4)}(1), \ldots$ of a loss $\mathcal{L}(x) = \|H(x) - H(x_*)\|^2$, where H is a linear matrix that subsamples a (random) subset of the image's pixels consisting of 90% of the total number of pixels, and x_* is a target image (different from the initial $x^{(0)}(1)$). The sampling process here is using an ImageNet trained flow model with the class condition 'bulbul'. As can be seen in this sequence of images, the intermediate steps of the optimization stay close to the distribution and pass through different sub-species of the bulbul bird. In the next section we provide a precise mathematical statement supporting this claim but for now let us provide some intuitive explanation.

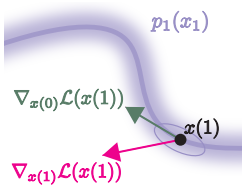


Figure 4: Implicit bias in differentiating through the solver.

3.4. Practical Implementation

The practical implementation of Algorithm 1 requires three algorithmic choices. First, one needs to decide how to initialize x_0. In all experiments we either initialize x_0 as a sample from the source distribution, i.e., normal Gaussian, or we use a variance preserving blend of a normal Gaussian with the backward solution from $t = 1$ to $t = 0$ of the observed signal when possible. Second, we need to choose the solver used to parameterize $x(1)$. To this end we utilize the `torchdiffeq` package (Chen, 2018), providing a wide class of differentiable ODE solvers. Backpropagating through the solver can be expensive in memory and we therefore use gradient checkpointing to reduce memory consumption at the cost of runtime. In most of our experiments we use the midpoint method with 6 function evaluations. Lastly, we need to choose the optimizer for the gradient step. Since the optimization we perform is not stochastic we choose to use the LBFGS algorithm with line search in all experiments. The runtime of the optimization depends on the problem but typically ranges from $5 - 15$ minutes per sample. For large text-2-image and text-2-audio models run times are higher and can reach $30 - 40$ minutes.

4. Theory

In this section we provide the theoretical support to the implicit regularization claim made in the previous section.

First, we revisit the family of Affine Gaussian Probability Paths (AGPP) taking noise to data that are used to supervise diffusion/flow models. When diffusion/flow models reach zero loss they reproduce these probability paths and we will therefore use them to analyze the implicit bias. Second, we use the method of adjoint dynamics to provide an explicit formula for the gradient $\nabla_{x_0}\mathcal{L}(x(1))$ under the AGPP assumption, and consequently derive the asymptotic change (velocity vector) in $x(1)$. Lastly, we interpret this velocity vector of $x(1)$ to demonstrate why it is pointing in the direction of the data distribution $p_1(x)$.

Affine Gaussian probability paths. Diffusion and recent flow based models use Affine Gaussian Probability Path (AGPP) to supervise their training. In particular, denoting $p_0 = \mathcal{N}(0, \sigma_0^2 I)$ the Gaussian noise (source) distribution and p_1 data (target) distribution, an AGPP is defined by

$$p_t(x) = \int p_t(x|x_1)p_1(x_1)dx_1, \quad (11)$$

where $p_t(x|x_1) = \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I)$ is a Gaussian kernel and $\alpha_t, \sigma_t : [0,1] \to [0,1]$ are called the *scheduler*, satisfying $\alpha_0 = 0$, $\sigma_1 \approx 0$, and $\alpha_1 = 1 = \sigma_0$, consequently guaranteeing that p_t interpolates (exactly or approximately) the source and target distributions at times $t = 0$ and $t = 1$, respectively. The velocity field that generates this probability path and coincide with the velocity field trained by diffusion/flow models at zero loss is (Lipman et al., 2023; Shaul et al., 2023)

$$u_t(x) = \int [a_t x + b_t x_1]\, p_t(x_1|x)dx_1 \quad (12)$$

where using Bayes' Theorem

$$p_t(x_1|x) = \frac{p_t(x|x_1)p_1(x_1)}{p_t(x)}, \tag{13}$$

and

$$a_t = \frac{\dot{\sigma}_t}{\sigma_t}, \quad b_t = \dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}. \tag{14}$$

One can also simplify the integral in equation 12 and write the marginal vector field in terms of the denoiser (Karras et al., 2022), $\hat{x}_{1|t}(x) = \int x_1 p_t(x_1|x) dx_1$:

$$u_t(x) = a_t x + b_t \hat{x}_{1|t}(x) \tag{15}$$

which for AGPP possesses a useful property we will use in Theorem 4.2, stated in the following proposition (proof in Appendix A.1):

Proposition 4.1. *For AGPP, the gradient of the denoiser $\hat{x}_{1|t}(x)$ w.r.t x is proportional to the variance of the random variable defined by $p_t(x_1|x)$, formally:*

$$D_x \hat{x}_{1|t}(x) = \frac{\alpha_t}{\sigma_t^2} \mathrm{Var}_{1|t}(x) \tag{16}$$

where

$$\mathrm{Var}_{1|t}(x) = \mathbb{E}_{p_t(x_1|x)} \left[x_1 - \hat{x}_{1|t}(x)\right] \left[x_1 - \hat{x}_{1|t}(x)\right]^T \tag{17}$$

Differentiating through the solver. When diffusion/flow models are optimized to a minimal loss they perfectly reproduce the AGPP velocity field, i.e., equation 12 (Lipman et al., 2023). For this velocity field we begin with an analysis of the differential of a solution (sample) $D_{x_0} x(1)$ for the continuous time exact case and a discrete time approximation.

Theorem 4.2. *For AGPP velocity field u_t (see equation 12) and $x(t)$ defined via equation 2 the differential of $x(1)$ as a function of x_0 is*

$$D_{x_0} x(1) = \sigma_1 \mathcal{T} \exp\left[\int_0^1 \gamma_t \mathrm{Var}_{1|t}(x(t)) dt\right], \tag{18}$$

where $\mathcal{T}\exp[\cdot]$ stands for a time-ordered exponential, $\gamma_t = \frac{1}{2}\frac{d}{dt}\mathrm{snr}(t)$ and we define $\mathrm{snr}(t) = \frac{\alpha_t^2}{\sigma_t^2}$.

The proof is given in Appendix A.2. In the exact case where $\sigma_1 = 0$ we also have $\int_0^1 \gamma_t dt = \infty$, nevertheless we show in Appendix A.2 that $D_{x_0} x(1)$ is the time-ordered exponential of a bounded time-dependent matrix. While a closed form expression to this integral is unknown, we note that the matrix-vector product $D_{x_0} x(1) v$ corresponds to an infinite sum of powers of the matrices $\gamma_t \mathrm{Var}_{1|t}(x(t))$ applied to v.

To gain better intuition and align our theory with practice, where discrete ODE solvers are used to obtain $x(1)$, we will now analyze the discrete time solver case. Let us consider an Euler ODE solver with N uniform steps of size $h = \frac{1}{N}$, then the differential of $x(1)$ as a function of x_0 is:

$$D_{x_0} x(1) = \prod_{m=0}^{N-1} \Big((1 + h a_{mh}) I + h \gamma_{mh} \mathrm{Var}_{1|mh}(x_{mh}) \Big) \tag{19}$$

note that the product is a time-ordered product, with m decreasing from right to left (derivation in Appendix A.3). The form of equation 19, consisting of powers of $\mathrm{Var}_{1|t}(x)$, provides insights as to why D-Flow works even with a low number of solver steps. Intuitively, the vector-matrix multiplication $\mathrm{Var}_{1|t}(x)v$ projects v on the major axes of the distribution of the data conditioned on x. As we will soon see, $D_{x_0} x(1)$ is key to understanding the implicit bias claim.

The dynamics of $x(1)$. Consider an optimization step updating the optimized variable x_0 with a gradient step, i.e., $x_0^\tau = x_0 - \tau \nabla_{x_0} \mathcal{L}(x(1))$, where the gradient $\nabla_{x_0} \mathcal{L}(x(1))$ can be now computed with the chain rule and equation 18,

$$\nabla_{x_0} \mathcal{L}(x(1)) = D_{x_0} x(1)^T \nabla_{x(1)} \mathcal{L}(x(1)), \tag{20}$$

We can now ask: *How is the sample $x(1)$ changing infinitesimally under this gradient step?* Denote by $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ the map taking initial conditions x_0 to solutions of equation 2 at $t = 1$, i.e., $\Phi(x_0) = x(1)$. The *variation* of $x(1)$ is

$$\begin{aligned} \delta x(1) &= \frac{d}{d\tau}\Big|_{\tau=0} \Phi\left(x_0 - \tau \nabla_{x_0} \mathcal{L}(x(1))\right) \\ &= -\left[D_{x_0} x(1) D_{x_0} x(1)^T\right] \nabla_{x(1)} \mathcal{L}(x(1)), \end{aligned}$$

where the first equality is the definition of variation and the second equality is using chain rule and equation 20. Indeed, the dynamics of $x(1)$ follow the projection of the gradient $\nabla_{x(1)} \mathcal{L}(x(1))$ with the operator $D_{x_0} x(1)$ that iteratively applies projection by the covariance matrix $\mathrm{Var}_{1|t}(x(t))$ at different times t (equations 18 and 19).

5. Related Work

Inverse Problems. A new line of works alter the diffusion generation process for training-free solutions of inverse problems. Most works can be viewed as building guidance strategies to the generation process of diffusion models. (Kawar et al., 2022) takes a variational approach deriving a solver for linear inverse problems. Similarly, (Chung et al., 2022; Wang et al., 2022) modify the generation process by enforcing consistency with the observations either via cost functions or projections (Choi et al., 2021; Wang et al., 2022; Lugmayr et al., 2022). Other approaches guide the sampling

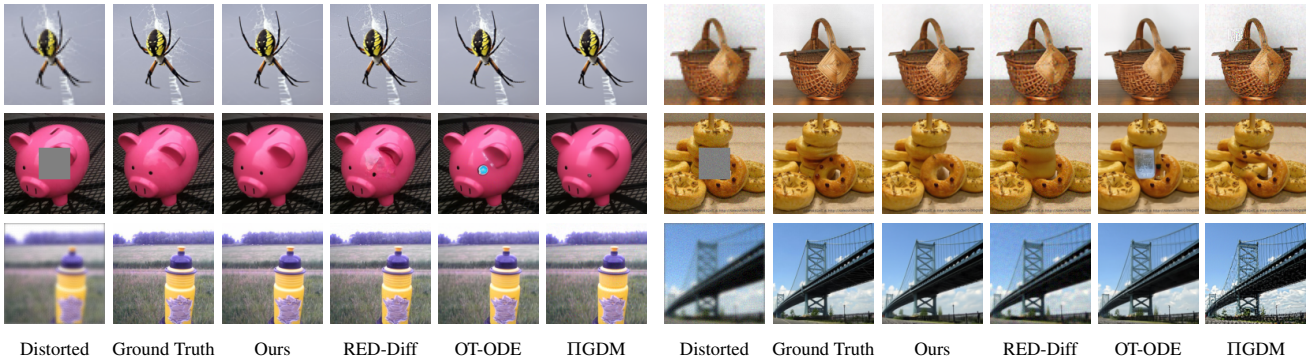


Figure 5: Qualitative comparison for linear inverse problems on ImageNet-128. GT samples from ImageNet-128 validation.

process with derivatives through the diffusion model at each denoising step (Ho et al., 2022; Chung et al., 2023; Song et al., 2023a; Pokle et al., 2023). A recent work by (Rout et al., 2023) extends the ideas for latent diffusion models by chained applications of encoder-decoder. Similar to our approach (Mardani et al., 2023) performs optimization of a reconstruction loss with score matching regularization.

Conditional sampling. Conditional sampling from diffusion models can be achieved by training an additional noise-aware condition predictor model (Song et al., 2020) or by incorporating the condition into the training process (Dhariwal & Nichol, 2021; Ho & Salimans, 2022a). These approaches however require task specific training. Plug-and-play approaches, on the other hand, utilize a pre-trained unconditional generative model as a prior. (Graikos et al., 2023) perform constrained generation via optimization of a reconstruction term regularized by the diffusion loss. (Liu et al., 2023) seeks for optimal control optimizing through the generation process to learn guiding controls. Our method formulates a similar optimization problem like earlier works on GANs (Bora et al., 2017) and normalizing flows (Asim et al., 2020; Dombrowski et al., 2021; Chávez, 2022). While (Asim et al., 2020) provides an analysis of a simplified linear model, (Dombrowski et al., 2021) analyzes the manifold preserving properties of diffeomorphic generative models. Our work provides a novel theoretical analysis of the gradient of differentiable functionals with respect to initial values of diffusion/flow generative processes with affine Gaussian paths. Our analysis unravels a fresh perspective on the implicit regularization implemented by differentiating through the generation process, even with a few number of steps (Appendix A.3), that aligns with the denoising attributes of diffusion/flow models. We note that using gradients through the solver for the case of discrete diffusion models was first used by (Wallace et al., 2023) for classifier guidance and by (Samuel et al., 2023b) to generate rare samples.

6. Experiments

We test D-Flow on the tasks: linear inverse problems on images, inverse problems with latent flow models and conditional molecule generation. For all the inverse problems experiments, where the observed signal provides structural information, we use a blend initialization to our algorithm speeding up convergence and often improving performance. Furthermore, in most experiments we find that there is no need in adding an explicit regularizing term in the optimization. The only cases where we found regularization helpful was in the noisy case for linear inverse problems and molecule generation. Additional details are in Appendix B.

6.1. Linear Inverse Problems on Images

We validate our method on standard linear inverse problems with a known degradation model on images. The tasks we consider are center-crop inpainting, super-resolution and Gaussian deblurring both in the noiseless and noisy case. In all cases we stop the optimization at a task dependent target PSNR. For the noisy case we choose the target PSNR to be the PSNR corresponding to the known added noise.

Tasks. We follow the same settings as in (Pokle et al., 2023): (i) For center-crop inpainting, we use a 40×40 centered mask; (ii) for super-resolution we use bicubic interpolation to downsample the images by $\times 2$; and lastly (iii) for Gaussian deblur we apply a Gaussian blur kernel of size 61×61 with intensity 1. For each task we report results for the noiseless and noisy (Gaussian noise of $\sigma_y = 0.05$, see equation 5) cases. Further implementation details can be found in the Appendix B.1.

Metrics. Following the evaluation protocol of prior works (Chung et al., 2022; Kawar et al., 2022) we report Fréchet Inception Distance (FID) (Heusel et al., 2018), Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM).

Table 1: Quantitative evaluation of linear inverse problems on face-blurred ImageNet-128.

Method	Inpainting-Center				Super-Resolution X2				Gaussian deblur			
	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑
$\sigma_y = 0$												
ΠGDM (Song et al., 2023a)	5.73	0.096	36.89	0.908	6.01	0.104	34.31	0.911	4.27	0.066	37.61	0.961
OT-ODE (Pokle et al., 2023)	5.65	0.094	37.00	0.893	4.28	0.097	33.88	0.903	2.04	0.048	37.44	0.959
RED-Diff (Mardani et al., 2023)	5.40	0.068	38.91	0.928	3.05	0.091	33.74	0.900	1.62	0.055	35.18	0.937
Ours	4.14	0.072	37.67	0.922	2.50	0.069	34.88	0.924	2.37	0.035	39.47	0.976
$\sigma_y = 0.05$												
ΠGDM (Song et al., 2023a)	7.99	0.122	34.57	0.867	4.38	0.148	32.07	0.831	30.30	0.328	29.96	0.606
OT-ODE (Pokle et al., 2023)	6.25	0.119	35.01	0.882	4.61	0.149	32.59	0.862	4.84	0.175	31.94	0.821
RED-Diff (Mardani et al., 2023)	14.63	0.171	32.42	0.820	10.54	0.182	31.82	0.852	21.43	0.229	31.41	0.807
Ours	4.76	0.102	34.609	0.890	4.26	0.146	32.35	0.858	5.35	0.167	31.99	0.820

Table 2: Quantitative evaluation of free-form inpainting on MS-COCO with T2I latent model.

Method	Inpainting-Free-Form				
	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	Clip score ↑
RED-Diff (Mardani et al., 2023)	23.31	0.327	33.28	0.813	0.882
Ours	16.92	0.327	32.34	0.759	0.892

Datasets and baselines. We use the face-blurred ImageNet-128 dataset and report our results on the $10k$ split of the face-blurred ImageNet dataset used by (Pokle et al., 2023). We compare our method to three recent state of the art methods: ΠGDM (Song et al., 2023a), OT-ODE (Pokle et al., 2023) and RED-Diff (Mardani et al., 2023). We use the implementation of (Pokle et al., 2023) for all the baselines. All methods, including ours, are evaluated with the same Cond-OT flow-matching class conditioned model trained on the face-blurred ImageNet-128 unless the reults we produced were inferior to the ones reported in (Pokle et al., 2023). In that case, we use the reported numbers from (Pokle et al., 2023).

Results. As shown in Table 1, our method shows strong performance across all tasks, Figure 5 shows samples for each type of distortion. For inpainting and super-resolution our method improves upon state of the art in most metrics. We believe that our method's ability to reach images with higher fidelity to the ground truth is attributed to the source point optimization, which, differently from guided sampling approaches such as (Song et al., 2023a; Pokle et al., 2023), iteratively correct the sampling trajectory to better match the observed signal. We further note that compared to RED-Diff, which is also an optimization approach, our method does not struggle in the noisy case and achieves SOTA performance. We show more samples in Figures 8,9.

6.2. Inverse Problems with Latent Flow Models

6.2.1. Image Inpainting

We demonstrate the capability of our approach for non-linear inverse problems by applying it to the task of free-form inpainting using a latent T2I FM model.

Metrics. To quantitatively assess our results we report standard metrics used in T2I generation: PSNR, FID (Heusel et al., 2018), and Clip score (Ramesh et al., 2022).

Datasets and baselines. The T2I model we use was trained on a proprietary dataset of $330m$ image-text pairs. It was trained on the latent space of an autoencoder as in (Rombach et al., 2022). The architecture is based on GLIDE (Nichol et al., 2022) and uses a T5 text encoder (Raffel et al., 2023). We evaluate on a subset of $1k$ samples from the validation set of the COCO dataset (Lin et al., 2015). We compare our method to RED-Diff (Mardani et al., 2023) as it is also not limited to linear inverse problems like the other baselines we used in the previous section. We tested different hyper-parameters for RED-Diff and report results with the best.

Results. Table 2 reports metrics for the baseline and our method. The metrics indicate that while RED-Diff better matches the unmasked areas, achieving superior performance for structural metrics (PSNR, SSIM) our method produces more semantically plausible image completion winning in perceptual metrics. We do observe that RED-Diff often produces artifacts for this task. Results are visualized in Figure 10.

6.2.2. Audio Inpainting and Super-Resolution

We evaluate our method on the tasks of music inpainting and super-resolution, utilizing a latent flow-matching music generation model. For this, we used a trained Cond-OT flow-matching text conditioned model with a transformer architecture of 325m parameters that operates on top of EnCodec representation (Défossez et al., 2022). The model's performance aligns with the current state-of-the-art scores in text-conditional music generation, achieving a Fréchet Audio Distance (FAD) score of 3.13 (Kilgour et al., 2018) on MusicCaps and FAD of 0.72 on in-domain data. The model is trained to generate ten-seconds samples. In the following, we evaluate the performance of inpainting and super-resolution using our method and RED-Diff as baseline, we report FAD and PSNR metrics.

Table 3: Quantitative evaluation of music generation with latent flow models.

Method	Inpainting (10%) FAD ↓	Inpainting (10%) PSNR ↑	Inpainting (20%) FAD ↓	Inpainting (20%) PSNR ↑	Super-Resolution X2 FAD ↓	Super-Resolution X2 PSNR ↑	Super-Resolution X4 FAD ↓	Super-Resolution X4 PSNR ↑	Super-Resolution X8 FAD ↓	Super-Resolution X8 PSNR ↑
In-domain										
RED-Diff (Mardani et al., 2023)	0.75	31.19	0.78	29.99	0.93	35.27	1.63	33.51	1.73	29.12
Ours	0.22	31.02	0.49	29.57	0.22	44.51	0.50	42.64	1.01	36.50
MusicCaps										
RED-Diff (Mardani et al., 2023)	3.59	32.81	3.72	30.39	3.07	37.13	3.51	34.99	3.97	30.49
Ours	1.19	31.78	1.31	31.08	1.25	38.93	1.42	35.83	2.09	32.20

Datasets and baselines. For evaluation, we use the MusicCaps benchmark, which comprises of 5.5K pairs of music and a textual description and an internal (in-domain) evaluation set of 202 samples, similar to (Copet et al., 2023; Ziv et al., 2024). Similar to prior work, we compute FAD metric using VGGish. We compare our method to RED-Diff (Mardani et al., 2023).

Results. Table 3 studies our method in inapinting and super resolution tasks. This experiment demosntrates the ability of our method to work in non-linear setup, where the flow model is trained over a neural representation and the cost function is evaluated on the post-decoded signal (neural representation after decoding). In the inpainting task, we center crop the signal by 10% and 20%, i.e., for a ten-seconds signal, we mask out two and four seconds respectively. In the super-resolution task we upscale a signal by factors of two, four, and eight,i.e., from 4kHz, 8kHz, 16kHz to 32kHz respectively. Overall, our method improves upon the baseline. Specifically, in all experiments, our method obtain the lowest FAD metric. In the inpainting task our method obtains a slightly lower PSNR from the baseline. Audio samples are attached in a supplementray material. Additional implementation details appear in Appendix B.2.2.

6.3. Conditional Molecule Generation on QM9

In this experiment we illustrate the application of our method for controllable molecule generation, which is of practical significance in the fields of material and drug design. The properties targeted for conditional generation (c in equation 7) include polarizability α, orbital energies $\varepsilon_{HOMO}, \varepsilon_{LUMO}$ and their gap $\Delta\varepsilon$, Diople moment μ, and heat capacity C_v. To assess the properties of the molecules generated, we used a property classifier ($\mathcal{F}$ in equation 7) for each property. Those classifiers were trained following the methodology outlined in (Hoogeboom et al., 2022). Further details are in Appendix B.3.

Metrics. To assess conditional generation, we calculate the Mean Absolute Error (MAE) between the predicted property value of the generated molecule by the property classifier, (Satorras et al., 2022), and the target property value. According to the conditional training protocol from (Hoogeboom et al., 2022), the property classifier is trained over half of the QM9 train set (50K) while the remaining half is used for training the conditional generative models. Additionally, we appraise the quality of the generated molecules by evaluating atom stability (the percentage of atoms with correct valency), molecule stability (the percentage of molecules where all atoms are stable), validity (as defined in RDKit (Landrum, 2016)), and the uniqueness of the generated molecules.

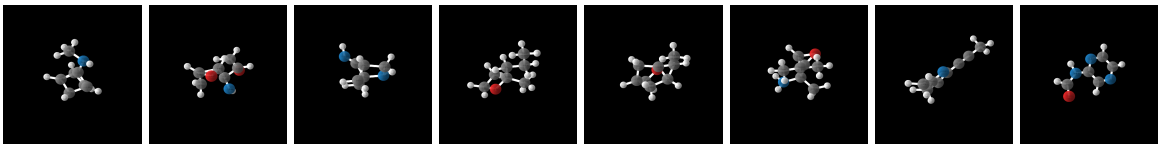
Table 4: Quantitative evaluation of conditional molecule generation. Values reported in the table are MAE (over 10K samples) for molecule property predictions (lower is better).

Property Units	α Bohr^2	$\Delta\varepsilon$ meV	ε_{HOMO} meV	ε_{LUMO} meV	μ D	C_v $\frac{\text{cal}}{\text{mol}}$K
QM9*	0.10	64	39	36	0.043	0.040
EDM	2.76	655	356	584	1.111	1.101
EquiFM	2.41	591	337	530	1.106	1.033
GeoLDM	2.37	587	340	522	1.108	1.025
Ours	1.39	344	182	330	0.300	0.784

Table 5: Stability and validity evaluation of D-flow on conditional molecule generation (10K samples).

Property	α	$\Delta\varepsilon$	ε_{HOMO}	ε_{LUMO}	μ	C_v
Molecule Stability (%)	56.2	59.4	60.2	59.4	60.7	57.9
Atom Stability (%)	93.6	93.9	94.1	93.8	94.2	93.6
Validity (%)	77.4	79.4	80.2	79.4	81.1	78.9
Validity & Uniqueness (%)	77.4	79.4	80.2	79.4	81.1	78.9

Dataset and baselines. The generative models used for this experiment are trained using the QM9 dataset (Ramakrishnan et al., 2014), a commonly used molecular dataset containing small molecules with up to 29 atoms. The model we use as prior in these experiments is an unconditional equivariant Flow-Matching model with CondOT path (Lipman et al., 2023), trained on the train set half used in (Hoogeboom et al., 2022) for conditional training. We compare our method to several state of the art *conditional models*: conditional EDM, Equivariant Flow-Matching (EquiFM) (Song et al., 2023b), and Geometric Latent Diffusion Model (GeoLDM)(Xu et al., 2023) an equivariant latent diffusion model. Additionally, we report the test MAE of each property classifier (denoted as QM9* in Table 4), which serves as an empirical lower bound. It is important to note that for each specific property of conditional generation, the



52.22 63.60 72.63 79.36 81.33 88.88 95.13 102.40

Figure 6: Qualitative visualization of controlled generated molecules for various polarizability (α) levels.

baseline methods utilized a distinct conditional model, each individually trained for generating that particular property while we used a single unconditional model.

Results. Table 4 demonstrates that our approach significantly outperforms all other baseline methods in the quality of conditional molecule generation. This superior performance is attributed to our direct optimization of the conditional generation. Table 5 presents the stability and validity metrics for our method. In comparison with conditional EDM, which achieves an average molecular stability of 82.1% across different properties, our method reveals a disparity in the stability of the generated molecules. This gap is a consequence of two factors. First, the trained Flow Matching unconditional model achieved inferior performance compared to EDM reaching molecular stability of 72.2%. Second, the optimization with respect to the property predictor does not achieve the same quality of generation as regular sampling. We further verify that the gain in MAE that D-Flow presents is not due to the degradation in the percentage of stable molecules and report both MAE values for stable and non-stable molecules withing the $10k$ generated sample, in Table 8. The MAE values for both stable and non-stable molecules are on par and improve by a large margin the existing baselines. Figure 6 visualize the controlled generation for different polarizability α values; all molecules in the figure are valid and stable with a classifier error lower than 1.

7. Discussion, Limitations and Future Work

We have presented a simple and general framework for controlled generation from pre-trained diffusion/flow models and demonstrated its efficacy on a wide range of problems from various domains and data types ranging from images, and audio to molecules. The main limitation of our approach is in its relatively long runtimes (see Section 3.4, and Appendix B) which stems from the need to back-propagate through multiple compositions of the velocity field (equivalently, the diffusion model). Our theoretical analysis and empirical evidence show however that computing gradients through the ODE solution have a desirable implicit bias, producing state of the art results on common conditional generation tasks. Consequently, an interesting future direction is to utilize the implicit bias but with potentially cheaper computational overhead, and draw connections to other biases used in other controlled generation paradigms.

Acknowledgments

OP is supported by a grant from Israel CHE Program for Data Science Research Centers and the Minerva Stiftung.

Impact Statement

In this paper we introduce a general approach for controlled data generation from generative priors. This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anderson, B., Hy, T.-S., and Kondor, R. Cormorant: Covariant molecular neural networks, 2019.

Asim, M., Daniels, M., Leong, O., Ahmed, A., and Hand, P. Invertible generative models for inverse problems: mitigating representation error and dataset bias, 2020.

Bar-Tal, O., Yariv, L., Lipman, Y., and Dekel, T. Multidiffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113*, 2023.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. Compressed sensing using generative models, 2017.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Chen, R. T. Q. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`.

Choi, J., Kim, S., Jeong, Y., Gwon, Y., and Yoon, S. Ilvr: Conditioning method for denoising diffusion probabilistic models, 2021.

Chung, H., Sim, B., Ryu, D., and Ye, J. C. Improving diffusion models for inverse problems using manifold constraints, 2022.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems, 2023.

Chávez, J. A. Generative flows as a general purpose solution for inverse problems, 2022.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. Simple and controllable music generation. In *NeurIPS*, 2023.

Detlefsen, N. S., Borovec, J., Schock, J., Jha, A. H., Koker, T., Liello, L. D., Stancl, D., Quan, C., Grechkin, M., and Falcon, W. Torchmetrics - measuring reproducibility in pytorch. *Journal of Open Source Software*, 7(70): 4101, 2022. doi: 10.21105/joss.04101. URL https://doi.org/10.21105/joss.04101.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

Dombrowski, A.-K., Gerken, J. E., and Kessel, P. Diffeomorphic explanations with normalizing flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. URL https://openreview.net/forum?id=ZBR9EpEl6G4.

Dombrowski, A.-K., Gerken, J. E., Müller, K.-R., and Kessel, P. Diffeomorphic counterfactuals with generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3257–3274, 2024. doi: 10.1109/TPAMI.2023.3339980.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Evans, L. C. An introduction to mathematical optimal control theory. *Lecture Notes, University of California, Department of Mathematics, Berkeley*, 3:15–40, 2005.

Graikos, A., Malkin, N., Jojic, N., and Samaras, D. Diffusion models as plug-and-play priors, 2023.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models, 2018.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022a.

Ho, J. and Salimans, T. Classifier-free diffusion guidance, 2022b.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models, 2022.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Kawar, B., Elad, M., Ermon, S., and Song, J. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. Fréchet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

Landrum, G. Rdkit: Open-source cheminformatics software. 2016. URL https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. Microsoft coco: Common objects in context, 2015.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling, 2023.

Liu, X., Wu, L., Zhang, S., Gong, C., Ping, W., and Liu, Q. Flowgrad: Controlling the output of generative odes with gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24335–24344, June 2023.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Gool, L. V. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.

Magnus, W. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673, 1954. doi: https://doi.org/10.1002/cpa.3160070404. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160070404.

Mardani, M., Song, J., Kautz, J., and Vahdat, A. A variational perspective on solving inverse problems with diffusion models, 2023.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know?, 2019.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022.

Pokle, A., Muckley, M. J., Chen, R. T. Q., and Karrer, B. Training-free linear image inversion via flows, 2023.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

Ramakrishnan, R., Dral, P., Rupp, M., and von Lilienfeld, A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 08 2014. doi: 10.1038/sdata.2014.22.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2022.

Rout, L., Raoof, N., Daras, G., Caramanis, C., Dimakis, A. G., and Shakkottai, S. Solving linear inverse problems provably via posterior sampling with latent diffusion models, 2023.

Sakurai, J. J. and Napolitano, J. *Modern Quantum Mechanics*. Cambridge University Press, 3 edition, 2020.

Samuel, D., Ben-Ari, R., Darshan, N., Maron, H., and Chechik, G. Norm-guided latent space exploration for text-to-image generation, 2023a.

Samuel, D., Ben-Ari, R., Raviv, S., Darshan, N., and Chechik, G. Generating images of rare concepts using pre-trained diffusion models, 2023b.

Satorras, V. G., Hoogeboom, E., Fuchs, F. B., Posner, I., and Welling, M. E(n) equivariant normalizing flows, 2022.

Shaul, N., Chen, R. T., Nickel, M., Le, M., and Lipman, Y. On kinetic optimal probability paths for generative models. In *International Conference on Machine Learning*, pp. 30883–30907. PMLR, 2023.

Song, J., Vahdat, A., Mardani, M., and Kautz, J. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=9_gsMA8MRKQ.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Song, Y., Gong, J., Xu, M., Cao, Z., Lan, Y., Ermon, S., Zhou, H., and Ma, W.-Y. Equivariant flow matching with hybrid probability transport, 2023b.

Wallace, B., Gokul, A., Ermon, S., and Naik, N. End-to-end diffusion latent optimization improves classifier guidance, 2023.

Wang, Y., Yu, J., and Zhang, J. Zero-shot image restoration using denoising diffusion null-space model, 2022.

Whang, J., Lei, Q., and Dimakis, A. G. Solving inverse problems with a flow-based noise model, 2021.

Xu, M., Powers, A., Dror, R., Ermon, S., and Leskovec, J. Geometric latent diffusion models for 3d molecule generation, 2023.

Yu, J., Wang, Y., Zhao, C., Ghanem, B., and Zhang, J. Freedom: Training-free energy-guided conditional diffusion model, 2023.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

Ziv, A., Gat, I., Lan, G. L., Remez, T., Kreuk, F., Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. Masked audio generation using a single non-autoregressive transformer. 2024.

A. Proofs and Theorems

A.1. Proof of Proposition 4.1

We restate Proposition 4.1 here:

Proposition A.1. *For AGPP, the gradient of the denoiser $\hat{x}_{1|t}(x)$ w.r.t x is proportional to the variance of the random variable defined by $p_t(x_1|x)$, formally:*

$$D_x \hat{x}_{1|t}(x) = \frac{\alpha_t}{\sigma_t^2} \mathrm{Var}_{1|t}(x) \tag{21}$$

where

$$\mathrm{Var}_{1|t}(x) = \mathbb{E}_{p_t(x_1|x)} \left[x_1 - \hat{x}_{1|t}(x)\right] \left[x_1 - \hat{x}_{1|t}(x)\right]^T \tag{22}$$

Proof. We recall a general affine Gaussian path is defined by

$$p_t(x|x_1) = \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I), \qquad \text{conditional probability path} \tag{23}$$

$$p_t(x) = \int p_t(x|x_1) q(x_1) dx_1, \qquad \text{marginal probability path} \tag{24}$$

where (α_t, σ_t) define the scheduler and q is the dataset probability density. The velocity fields defining these paths are (Lipman et al., 2023):

$$u_t(x|x_1) = a_t x + b_t x_1, \qquad a_t = \frac{\dot{\sigma}_t}{\sigma_t}, \quad b_t = \dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t} \qquad \text{conditional velocity field} \tag{25}$$

$$u_t(x) = \int u_t(x|x_1) p_t(x_1|x) dx_1, \qquad p_t(x_1|x) = \frac{p_t(x|x_1) q(x_1)}{p_t(x)} \qquad \text{marginal velocity field} \tag{26}$$

The differential of the denoiser is then:

$$D_x \hat{x}_{1|t}(x) = D_x \int x_1 p_t(x_1|x) dx_1 = \int x_1 \nabla_x p_t(x_1|x) dx_1 \tag{27}$$

Since $p_t(x|x_1)$ is a Gaussian:

$$\nabla_x p_t(x|x_1) = \frac{\alpha_t x_1 - x}{\sigma_t^2} p_t(x|x_1) \tag{28}$$

and plugging into 24, we have:

$$\nabla_x p_t(x) = \int \frac{\alpha_t x_1 - x}{\sigma_t^2} p_t(x|x_1) q(x_1) dx_1 \tag{29}$$

using 26, we get:

$$\nabla_x p_t(x_1|x) = p_t(x_1|x) \frac{\alpha_t}{\sigma_t^2} \left(x_1 - \hat{x}_{1|t}(x)\right) \tag{30}$$

therefore, 27 takes the form:

$$\begin{aligned} D_x \hat{x}_{1|t}(x) &= \int \frac{\alpha_t}{\sigma_t^2} p_t(x_1|x) x_1 (x_1 - \hat{x}_{1|t}(x))^T dx_1 = \\ &= \int \frac{\alpha_t}{\sigma_t^2} p_t(x_1|x) (x_1 - \hat{x}_{1|t}(x))(x_1 - \hat{x}_{1|t}(x))^T dx_1 = \frac{\alpha_t}{\sigma_t^2} \mathrm{Var}_{1|t}(x) \end{aligned} \tag{31}$$

□

A.2. Proof of Theorem 4.2

We restate Theorem 4.2 here:

Theorem A.2. *For AGPP velocity field u_t (see equation 12) and $x(t)$ defined via equation 2 the differential of $x(1)$ as a function of x_0 is*

$$D_{x_0}x(1) = \sigma_1 \mathcal{T} \exp\left[\int_0^1 \gamma_t \mathrm{Var}_{1|t}(x(t))dt\right], \tag{32}$$

where $\mathcal{T}\exp[\cdot]$ stands for a time-ordered exponential, $\gamma_t = \frac{1}{2}\frac{d}{dt}\mathrm{snr}(t)$ and we define $\mathrm{snr}(t) = \frac{\alpha_t^2}{\sigma_t^2}$.

Proof. To compute the differential of $x(1)$ w.r.t the initial point x_0 we utilize adjoint dynamics.

Let us define the adjoint $p(t) = D_{x(t)}x(1)$. The dynamics of $p(t)$ are defined by the following ODE (Evans, 2005):

$$\dot{p}(t) = -D_x u_t(x(t))^T p(t) \tag{33}$$
$$p(1) = D_{x(1)}x(1) = I. \tag{34}$$

To compute $D_{x_0}x(1)$ we solve 33 from time $t = 1$ back to time $t = 0$. Then,

$$p(0) = D_{x_0}x(1). \tag{35}$$

First, we will use the properties of AGPPs to further analyze the differential of the velocity field, $D_x u_t(x(t))$, that defines the adjoint dynamics in equation 33.

We write the velocity field in terms of the denoiser by plugging 25 into 26:

$$u_t(x) = a_t x + b_t \hat{x}_{1|t}(x) \tag{36}$$

and using equation 31, the differential of the velocity field is:

$$D_x u_t(x) = a_t I + b_t D_x \hat{x}_{1|t}(x) = a_t I + b_t \frac{\alpha_t}{\sigma_t^2}\mathrm{Var}_{1|t}(x) \tag{37}$$

The adjoint dynamics are then given by:

$$\dot{p}(t) = A(t)p(t) \tag{38}$$
$$p(1) = D_{x(1)}x(1) = I. \tag{39}$$

where

$$A(t) = -\left(a_t I + \gamma_t \mathrm{Var}_{1|t}(x)\right)^T. \tag{40}$$

and we define:

$$\gamma_t = b_t \frac{\alpha_t}{\sigma_t^2} = \left(\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}\right)\frac{\alpha_t}{\sigma_t^2} = \frac{\dot{\alpha}_t \alpha_t \sigma_t^2 - \alpha_t^2 \dot{\sigma}_t \sigma_t}{\sigma_t^4} = \frac{1}{2}\frac{d}{dt}\left(\frac{\alpha_t^2}{\sigma_t^2}\right) = \frac{1}{2}\frac{d}{dt}\mathrm{snr}(t) \tag{41}$$

The adjoint ODE 38 is a non-autonomous linear ODE and together with the initial condition equation 39 its solution is given by (Sakurai & Napolitano, 2020):

$$p(t) = \mathcal{T}\exp\left[-\int_t^1 A(s)ds\right]p(1) \tag{42}$$

known as the time-ordered exponential, defined as follows:

$$\mathcal{T}\exp\left[\int_t^1 A(s)ds\right] = \sum_{n=1}^{\infty}\frac{(-1)^n}{n!}\int_t^1\cdots\int_t^1 \mathcal{T}\{A(s_1)A(s_2)\dots A(s_n)\}ds_1ds_2\dots ds_n \tag{43}$$

where $\mathcal{T}\{A(s_1)A(s_2)\dots A(s_n)\}$ orders the product of matrices such that the value of s_i decreases from right to left. Alternative equivalent solutions to this differential equation are the dyson series (Sakurai & Napolitano, 2020) and the Magnus expansion (Magnus, 1954).

When the matrices $\{A(s)\}_{s\in[1,t]}$ commute, i.e., $[A(s), A(s')] = A(s)A(s') - A(s')A(s) = 0$, the time-ordered exponential 43 reduces to the first term in the sum for $n = 1$ in 43. We can therefore separate the first term in 40, since I commutes with every matrix, and arrive at a simplified solution for $t = 0$:

$$D_{x_0}x(1) = \sigma_1\mathcal{T}\exp\left[\int_0^1 \gamma_t \mathrm{Var}_{1|t}(x)dt\right], \tag{44}$$

where $\exp\left[\int_0^1 a_t dt\right] = \sigma_1$ under the assumption that a_t is integrable, concluding the proof.

□

Next we show that the integral in equation 18 is defined also for $\sigma_1 = 0$.

Lemma A.3. *For a Lipschitz function $f : \mathbb{R}^d \to \mathbb{R}$ we have that $\int \mathcal{N}(x|y, \sigma^2 I)f(x)dx = f(y) + \mathcal{O}(\sigma)$.*

Proof.

$$\begin{aligned}\left|\int \mathcal{N}(x|y,\sigma^2 I)f(x)dx - f(y)\right| &\le \int \mathcal{N}(x|y,\sigma^2 I))\,|f(x)-f(y)|\,dx\\ &= \int \mathcal{N}(z|0,I)\,|f(\sigma z + y) - f(y)|\,dz\\ &\le K\sigma\int \mathcal{N}(z|0,I)\,|z|\,dz\\ &= \mathcal{O}(\sigma),\end{aligned}$$

where in the first equality we performed a change of variable $z = \frac{x-y}{\sigma}$, and in the second inequality we used the fact that f is Lipschitz with constant $K > 0$. □

Using this Lemma we prove (under the assumption that $p_1(x)$ and its derivatives is Lipschitz):

Proposition A.4. *The denoiser asymptotics at $t \to 1$ is*

$$\hat{x}_{1|t}(x) = \frac{x}{\alpha_t} + \mathcal{O}(\sigma_t) \tag{45}$$

Proof. First we note that we assume $\sigma_t \to 0$ and $\alpha_t \to 1$ as $t \to 1$,

$$\mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I) = c_t\mathcal{N}\left(x_1\middle|\frac{x}{\alpha_t}, \left(\frac{\sigma_t}{\alpha_t}\right)^2 I\right), \tag{46}$$

where c_t is some normalization constant such that $c_1 = 1$. Now,

$$p_t(x) = \int \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I)p_1(x_1)dx_1 \tag{47}$$

$$= c_t\int \mathcal{N}\left(x_1\middle|\frac{x}{\alpha_t}, \left(\frac{\sigma_t}{\alpha_t}\right)^2 I\right)p_1(x_1)dx_1 \tag{48}$$

$$= c_t p_1\left(\frac{x}{\alpha_t}\right) + \mathcal{O}(\sigma_t), \tag{49}$$

where in second equality we used equation 46 and the last equality Lemma A.3.

$$\hat{x}_{1|t}(x) = \int x_1 p_t(x_1|x) dx_1 \tag{50}$$

$$= \int x_1 \frac{\mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I) p_1(x_1)}{p_t(x)} dx_1 \tag{51}$$

$$= \int x_1 \frac{c_t \mathcal{N}\left(x_1 \middle| \frac{x}{\alpha_t}, \left(\frac{\sigma_t}{\alpha_t}\right)^2 I\right) p_1(x_1)}{p_t(x)} dx_1 \tag{52}$$

$$= \frac{c_t \frac{x}{\alpha_t} p_1\left(\frac{x}{\alpha_t}\right) + \mathcal{O}(\sigma_t)}{c_t p_1\left(\frac{x}{\alpha_t}\right) + \mathcal{O}(\sigma_t)} \tag{53}$$

$$= \frac{x}{\alpha_t} + \mathcal{O}(\sigma_t), \tag{54}$$

where in the second equality we used the definition of $p_t(x_1|x)$, in the third equality we used equation 46, and in the fourth equality we used Lemma A.3. □

Now we can show that $D_x u_t(x(t))$ is bounded as $t \to 1$

$$D_x u_t(x(t)) = a_t I + b_t D_x \hat{x}_{1|t}(x) \tag{55}$$

$$= a_t I + b_t \left(\frac{1}{a_t} I + \mathcal{O}(\sigma_t)\right) \tag{56}$$

$$= \frac{\dot{\alpha}_t}{\alpha_t} I + \mathcal{O}(1), \tag{57}$$

where in the first equality we used equation 37, in the second Proposition A.4 (and the fact that the derivatives of p_1 are Lipschitz for the derivation of the asymptotic rule), and in the last equality equation 25. Furthermore $D_x u_t(x(t))$ is bounded as $t \to 0$ as both a_0, b_0 are well defined. This means that $D_x u_t(x(t))$ is integrable over $[0, 1]$.

A.3. Discrete Time Analysis

Theorem 4.2 analyzes the continuous time case, providing intuition about the behavior of the dynamics of $x(1)$ when changing the initial condition x_0. The final expression 18, however, involves a time-ordered exponential which may be hard to interpret. Furthermore, our experiments show that even with a small number of discrete steps, differentiating $x(1)$ with respect to x_0 yields meaningful gradients, performing well in practice (see B).

Let us consider Euler solver with N uniform steps of size $h = \frac{1}{N}$, with initial point x_0. An intermediate point at time mh, x_{mh} is given by:

$$x_{(m+1)h} = x_{mh} + h u_{mh}(x_{mh}) \tag{58}$$

We are interested at the derivative of $x_{Nh} = x(1)$ w.r.t x_0.

By the chain rule and equation 37, one can write:

$$D_{x_0} x_1 = \prod_{m=0}^{N-1} D_{x_{mh}} x_{(m+1)h} = \prod_{m=0}^{N-1} \Big((1 + h a_{mh}) I + h \gamma_{mh} \mathrm{Var}_{1|mh}(x_{mh})\Big) \tag{59}$$

note that this is also a time-ordered product, with m decreasing from left to right.

For the CondOT probability path, where $\alpha_t = t, \sigma_t = 1 - t$, 59 takes the form:

$$D_{x_0} x_1 = \prod_{m=0}^{N-1} \Big(\frac{1 - (m+1)h}{1 - mh} I + \frac{mh^2}{(1 - mh)^3} \mathrm{Var}_{1|mh}(x_{mh})\Big) \tag{60}$$

A.4. On Flow-Matching, Denoisers and Noise Prediction

Consider a general affine conditional probability path defined by the following transport map:

$$x_t = \sigma_t x_0 + \alpha_t x_1$$

where $x_0 \sim p_0$ and $x_1 \sim p_1$.

For different choices of σ_t, α_t we can parametrize known diffusion and flow-matching paths. The corresponding conditional vector field on x_1 is:

$$u_t(x|x_1) = \frac{\dot{\sigma}_t}{\sigma_t}(x - \alpha_t x_1) + \dot{\alpha}_t x_1 = \frac{\dot{\sigma}_t}{\sigma_t}x - \left(\frac{\dot{\sigma}_t \alpha_t}{\sigma_t} - \dot{\alpha}_t\right) x_1$$

and the conditional vector field on x_0 is:

$$u_t(x|x_0) = \dot{\sigma}_t x_0 + \frac{\dot{\alpha}_t}{\alpha_t}(x - \sigma_t x_0) = \frac{\dot{\alpha}_t}{\alpha_t}x - \left(\frac{\dot{\alpha}_t \sigma_t}{\alpha_t} - \dot{\sigma}_t\right) x_0$$

where $\dot{f} = \frac{d}{dt}f$.

Consider the marginal velocity field:

$$u_t(x) = \int u_t(x|x_1)p_t(x_1|x)dx_1 = \int u_t(x|x_0)p_t(x_0|x)dx_0$$

One can express it in terms of the optimal *denoiser* function, $\hat{x}_{1|t}(x)$:

$$u_t(x) = \frac{\dot{\sigma}_t}{\sigma_t}\int x p_t(x_1|x)dx_1 - \left(\frac{\dot{\sigma}_t \alpha_t}{\sigma_t} - \dot{\alpha}_t\right)\int x_1 p_t(x_1|x)dx_1 = \frac{\dot{\sigma}_t}{\sigma_t}x - \left(\frac{\dot{\sigma}_t \alpha_t}{\sigma_t} - \dot{\alpha}_t\right)\hat{x}_{1|t}(x) \tag{61}$$

For Cond-OT:

$$u_t(x) = \frac{\hat{x}_{1|t}(x) - x}{1 - t} \tag{62}$$

Or, in terms of the optimal noise predictor, $\epsilon_t(x)$, like in DDPM:

$$u_t(x) = \frac{\dot{\alpha}_t}{\alpha_t}\int x p_t(x_0|x)dx_0 - \left(\frac{\dot{\alpha}_t \sigma_t}{\alpha_t} - \dot{\sigma}_t\right)\int x_0 p_t(x_0|x)dx_0 = \frac{\dot{\alpha}_t}{\alpha_t}x - \left(\frac{\dot{\alpha}_t \sigma_t}{\alpha_t} - \dot{\sigma}_t\right)\epsilon_t(x)$$

and for Cond-OT:

$$u_t(x) = \frac{x - \epsilon_t(x)}{t} \tag{63}$$

B. Implementation details

B.1. Linear Inverse Problems on Images

Optimization details. For all experiments in this section we used the LBFGS optimizer with 20 inner iterations for each optimization step with line search. Stopping criterion was set by a target PSNR value, varying for different tasks. The solver used was midpoint with 6 function evaluations. The losses, regularizations, initializations and stopping criterions of our algorithm for the linear inverse problems are listed in Table 6. In the Table χ^d regularization corresponds to equation 10 and λ denotes the coefficients used.

Table 6: Algorithmic choices for the ImageNet-128 linear inverse problems tasks.

	Inpainting-Center		**Super-Resolution X2**		**Gaussian Deblur**	
	$\sigma_y = 0$	$\sigma_y = 0.05$	$\sigma_y = 0$	$\sigma_y = 0.05$	$\sigma_y = 0$	$\sigma_y = 0.05$
Loss	$-\text{PSNR}(Hx, y)$		$-\text{PSNR}(Hx, y)$		$-\text{PSNR}(H^\dagger Hx, H^\dagger y)$	$-\text{PSNR}(Hx, y)$
Regularization	None	χ^d, with $\lambda = 0.01$	None	χ^d, with $\lambda = 0.01$	None	χ^d, with $\lambda = 0.01$
Initialization	0.1 blend		0.1 blend		0.1 blend	
Target PSNR	45	32	55	32	55	32

Runtimes. For noiseless tasks: inpainting center crop took on avarage 10 minutes per image, super resolution took 12.5 minutes per image and Gaussian deblurring took 15.5 minutes per image. For the noisy tasks: inpainting center crop took on avarage 4 minutes per image, super resolution took 2.5 minute per image and Gaussian deblurring took 3.5 minutes per image. Experiments ran on 32GB NVIDIA V100 GPU.

Metrics are computed using the open source TorchMetrics library (Detlefsen et al., 2022).

RED-Diff baseline. To use the RED-Diff baseline with a FM cond-OT trained model we transform the velocity field to epsilon prediction according to 63. We searched for working parameters and reported results that outperformed the results that were produced by (Pokle et al., 2023) with an epsilon prediction model, otherwise we kept the number from (Pokle et al., 2023).

B.2. Inpainting with Latent Flow Models

B.2.1. Image inpainting

Optimization details. In this experiment we used the LBFGS optimizer with 20 inner iterations for each optimization step with line search. Stopping criterion was set by a runtime limit of 30 minutes, but optimization usually convergences before. The solver used was midpoint with 6 function evaluations and the loss was negative PSNR without regularization. We initialized the algorithm with a backward blend with $\alpha = 0.25$. To facilitate the backpropagation through a large T2I model we use gradient checkpointing.

The validation set of the COCO dataset, used for evaluation, was downloaded from http://images.cocodataset.org/zips/val2017.zip.

RED-Diff baseline. To adapt RED-Diff to a latent space diffusion model, let us recall the loss used in RED-Diff:

$$\ell(\mu) = \|y - f(\mu)\|^2 + \lambda_t(\texttt{sg}\left[\epsilon(x(t), t) - \epsilon\right])^T \mu \tag{64}$$

where f can be any differentiable function. In latent diffusion/flow model for inverse problems we can model f as $f = H(\texttt{decode}(\mu))$, where $\texttt{decode}$ applies the decoder of the autoencoder used in the latent diffusion/flow model and H is the corruption operator. We use $\text{lr} = 0.25, \lambda = 0.25$.

B.2.2. Audio inpainting

Optimization details. We follow the same setup described in B.2.1. Differently, we use 10 inner iterations and stop after 100 global iterations. We initialize the algorithm with a backward blend with $\alpha = 0.1$.

RED-Diff baseline. We follow the same adaptation described above in B.2.1. We use $\text{lr} = 0.05, \lambda = 0.5$.

B.3. Conditional Molecule Generation on QM9

Optimization details. In this section, we describe how Algorithm 1 was practically applied in the QM9 experiment. We initialized $x_0 \in \mathbb{R}^{n\times 9}$ for the experiment, where n represents the molecule's atom count and 9 the number of attributes per atom, using a standard Gaussian distribution. To enhance optimization process stability, we ensured x_0 had a feature-wise mean of zero and a standard deviation of one by normalizing it after every optimization step. We employed the midpoint method for the ode solver, with a total of 100 function evaluations, i.e.step size of $1/50$. The optimization technique utilized was LBFGS with line search, configured with 5 optimization steps and a limit of 5 inner iterations for each step. The learning rate was set to 1. On average, generating a single molecule took approximately 2.5 minutes using a single NVIDIA Quadro RTX8000 GPU.

Table 7: Comparison of generated molecules quality using different solvers and D-Flow.

Sample Method	NFE (#)	Molecule Stability (%)	Atom Stability (%)	Validity (%)	Validity & Uniqueness (%)
Dopri5 Adaptive Solver	-	72.03	96.14	85.00	83.84
Midpoint (50 steps)	100	72.10	96.18	85.56	84.39
Midpoint (50 steps) + optimization	100	58.97	93.87	79.38	79.38

In Table 8 below we report MAE values over the split to stable and non-stable molecules within the $10k$ generated samples. Other baselines and the result denoted as 'Ours' in the table report MAE on the entire $10k$ set of molecules without distinguishing between stable and non-stable ones. Since our method produces lower stability percentage, we also report the MAE on the stable and non-stable splits. It can be seen that our improved MAE performance is not due to producing non-stable molecules with lower MAE, but performance is also SOTA on generated stable molecules.

Table 8: Quantitative evaluation of conditional molecule generation. Values reported in the table are MAE (over 10K samples) for molecule property predictions (lower is better).

Property Units	α Bohr^2	$\Delta\varepsilon$ meV	ε_{HOMO} meV	ε_{LUMO} meV	μ D	C_v $\frac{\text{cal}}{\text{mol}}$K
QM9*	0.10	64	39	36	0.043	0.040
EDM	2.76	655	356	584	1.111	1.101
EQUIFM	2.41	591	337	530	1.106	1.033
GEOLDM	2.37	587	340	522	1.108	1.025
Ours	1.39	344	182	330	0.300	0.784
Ours-stable	1.40	347	180	337	0.287	0.835
Ours-non stable	1.38	340	186	318	0.321	0.714

QM9. The QM9 dataset (Ramakrishnan et al., 2014), a widely recognized collection, encompasses molecular characteristics and atomic positions for 130K small molecules, each containing no more than 9 heavy atoms (up to 29 atoms when including hydrogens). The train/validation/test partitions used are according to (Anderson et al., 2019) and consists of 100K/18K/13 samples per partition. We provide additional details regarding the properties used in the experiment:

- α Polarizabilty - Tendency of a molecule to acquire an electric dipole moment when subjected to anexternal electric field.
- ε_{HOMO} - Highest occupied molecular energy.
- ε_{LUMO} - Lowest unoccupied molecular energy.
- $\Delta\varepsilon$ - The difference between HOMO and LUMO.
- μ - Dipole moment.
- C_v - Heat capacity at 298.15K.

C. Additional Experiments and Ablations

C.1. Linear Inverse Problems on Images: Denoising

In this experiment we consider the task of denoising. The corrupted signal, y, is given by $y = x + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_y I)$, with $\sigma_y = 0.05$. Hyperparameters are the same as in Table 6 for the noisy experiments. Reported metrics are in Table 9.

Table 9: Quantitative evaluation of denoising inverse problem on face-blurred ImageNet-128.

	Denoising			
Method	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑
$\sigma_y = 0.05$				
ΠGDM (Song et al., 2023a)	9.60	0.107	35.11	0.903
OT-ODE (Pokle et al., 2023)	3.14	0.062	37.34	0.964
RED-Diff (Mardani et al., 2023)	9.19	0.105	32.52	0.895
Ours	2.83	0.060	36.05	0.952

C.2. Ablation: Regularization Coefficient

As reported in Table 6, on the tasks of linear inverse problems on images, we used the source point χ^d regularization, equation 10, for the noisy case. In the plot below, 7, we report the evaluation metrics (FID, LPIPS, PSNR, SSIM) for varying regularization coefficient values, λ, on the task of noisy super-resolution. All other hyperparameters are as reported in Table 6.

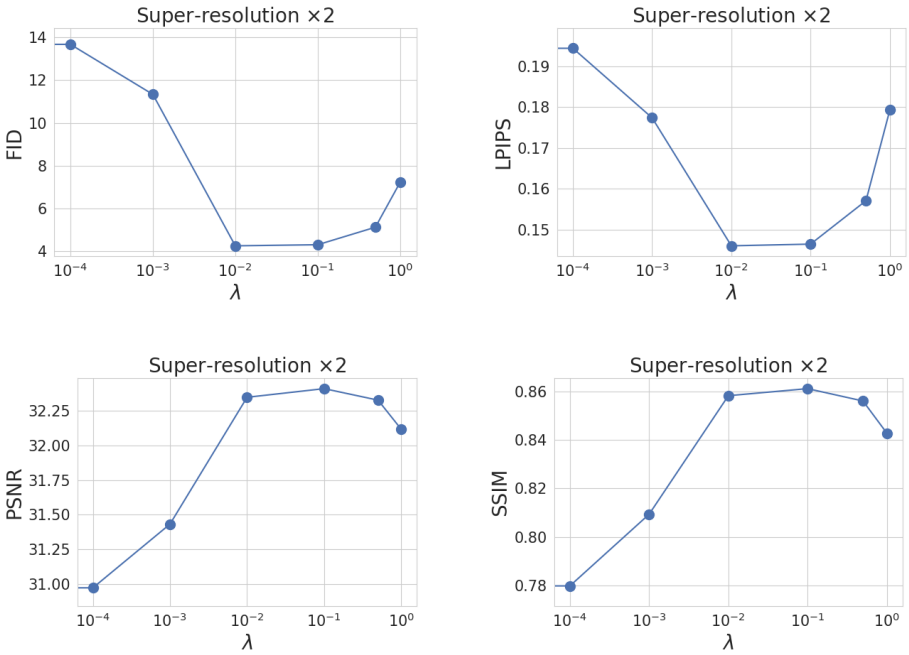


Figure 7: Evaluation metrics vs. regularization coefficient λ of χ^d regularization over x_0 for noisy super-resolution on ace-blurred ImageNet-128.

D. Additional Qualitative Results

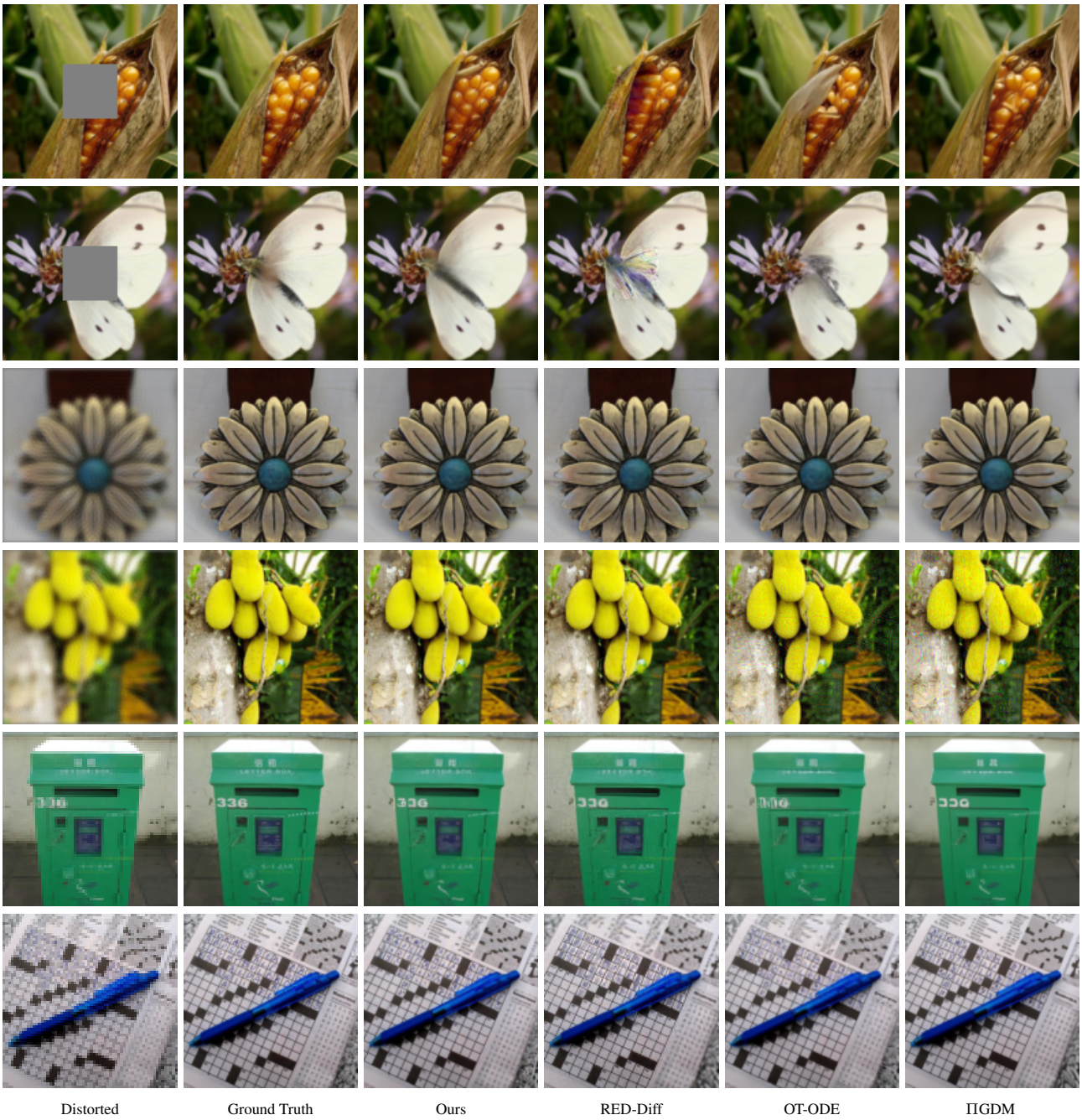
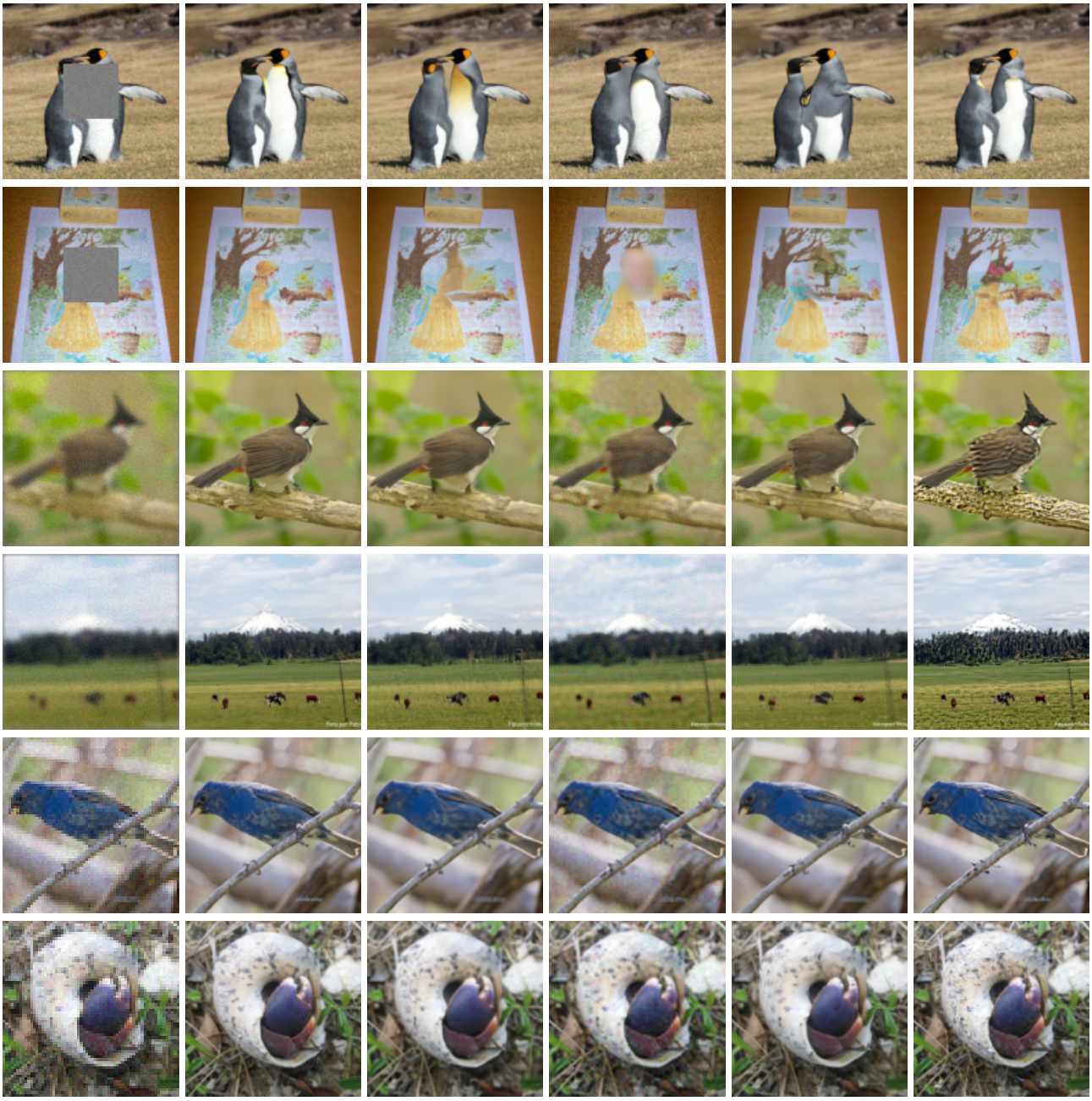
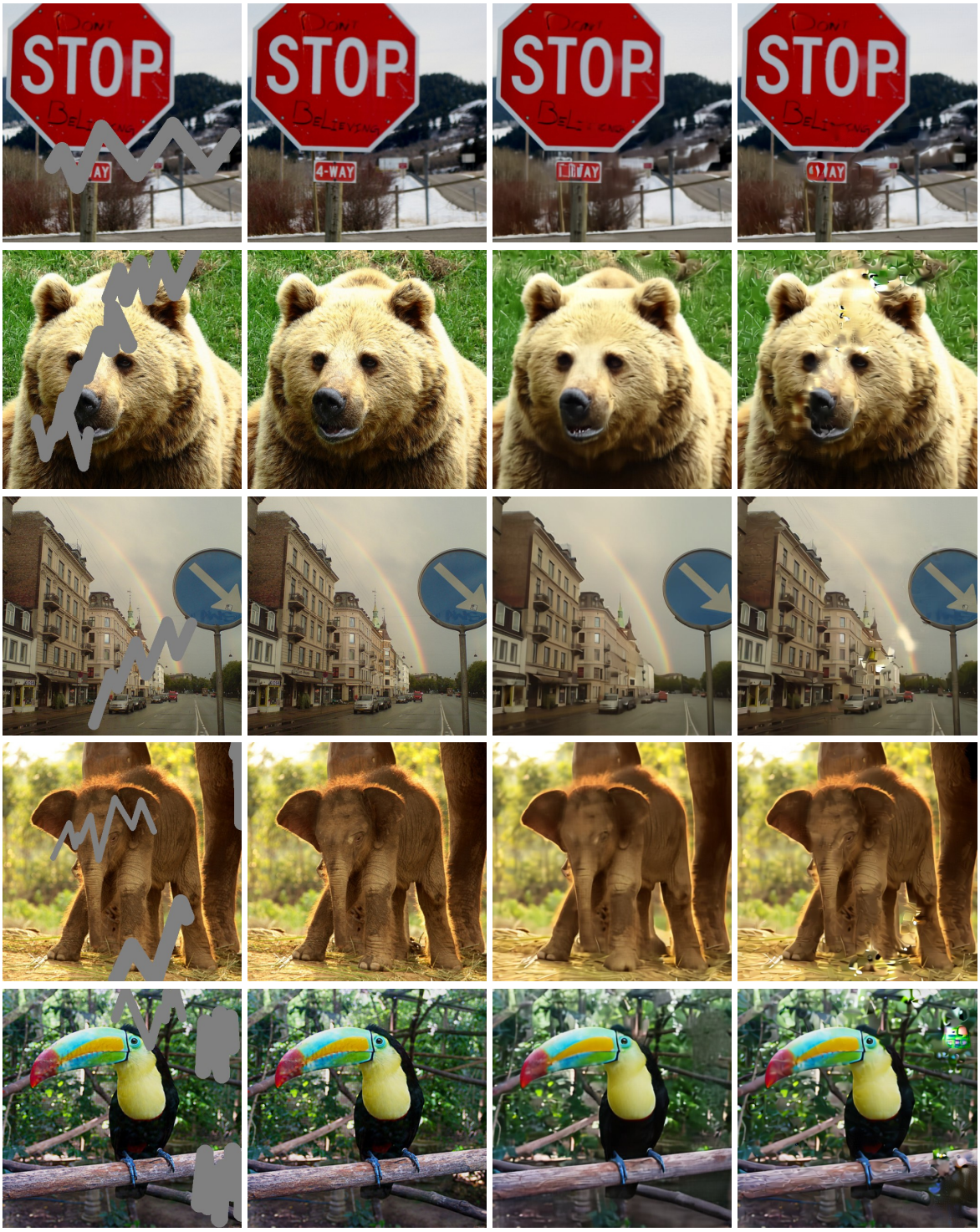


Figure 8: Qualitative comparison for linear inverse problems on ImageNet-128 for the noiseless case. GT samples come from the face-blurred ImageNet-128 validation set.



Distorted | Ground Truth | Ours | RED-Diff | OT-ODE | ΠGDM

Figure 9: Qualitative comparison for linear inverse problems on ImageNet-128 for the noisy case. GT samples come from the face-blurred ImageNet-128 validation set.



Distorted Ground Truth Ours RED-Diff

Figure 10: Qualitative comparison for free-form inpainting on the MS-COCO dataset using a T2I latent FM model. GT samples come from the MS-COCO validation set.