FLOW-DRIVEN OPTIMIZATION METHODS FOR IMAGE INVERSE PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow matching is a recent state-of-the-art framework for generative modeling. A variety of optimization methods that integrate flow matching have been developed for image inverse problems. In this paper, a framework of gradient-related optimization methods is proposed that incorporates the flow matching model for image inverse problems. We introduce a novel regularization term that takes the form of a distance-based function. We prove that the negative gradient of this regularization term corresponds to the velocity field of the flow matching model. Based on this theoretical result, any gradient-related optimization method can be applied to solve optimization problems involving this new regularization term, provided that the data consistency term is differentiable. We propose flow-driven two-step gradient descent (TSGD), a flow-driven fast version of TSGD, flow-driven BCD, and flow-driven ADMM algorithms. Numerically, we evaluate these gradient-related methods on four inverse problems across three datasets. Numerical results demonstrate that FlowDPS, FlowBCD, and FlowADMM achieve superior performance, with FlowBCD and FlowADMM standing out in particular.

1 INTRODUCTION

Image inverse problems seek to recover a clean image from degraded or incomplete observations in tasks such as super-resolution, deblurring, and inpainting. The observation model is commonly written as

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{u}) + \boldsymbol{n},$$

where $\boldsymbol{u}$ is an unknown image, $\boldsymbol{y}$ is the observation, $\mathcal{A}$ is the forward degradation operator, and $\boldsymbol{n}$ denotes measurement noise. Information loss can make reconstruction nonunique, while ill-conditioning can make it highly sensitive to measurement noise. Just sticking to the observations alone usually doesn't guarantee a unique and stable solution. Image priors complement the observation model by constraining aspects of the reconstruction that the measurements do not fully determine (Tikhonov & Arsenin, 1977).

Maximum A Posteriori (MAP) estimation provides a probabilistic basis for combining the observation model with prior information, which forms a foundation of classical Bayesian image restoration. The Bayesian posterior satisfies $p(\boldsymbol{u} \mid \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{u})p(\boldsymbol{u})$, where the likelihood $p(\boldsymbol{y} \mid \boldsymbol{u})$ describes the observation process and the prior $p(\boldsymbol{u})$ captures image statistics. Maximizing the posterior density is equivalent to minimizing the negative log-likelihood and negative log-prior:

$$\hat{\boldsymbol{u}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{u}} p(\boldsymbol{u} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{u}} \{-\log p(\boldsymbol{y} \mid \boldsymbol{u}) - \log p(\boldsymbol{u})\}.$$

For additive noise independent of $\boldsymbol{u}$ and distributed as $\mathcal{N}(\boldsymbol{0}, \sigma_y^2 I)$, with a prior of the form $p(\boldsymbol{u}) \propto \exp[-\lambda R(\boldsymbol{u})]$, this yields

$$\hat{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}} \left\{ \frac{1}{2\sigma_y^2} \|\mathcal{A}(\boldsymbol{u}) - \boldsymbol{y}\|_2^2 + \lambda R(\boldsymbol{u}) \right\}.$$

This objective balances a data consistency term, which penalizes disagreement with the observations, against a regularization term, which encodes prior information about the image.

A variety of approaches have been developed in the literature to design this regularization term, such as the ℓ_p-norm with $p \in \{0, \frac{1}{2}, 1, 2, 3\}$, Tikhonov regularization, and the total variation norm, among

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

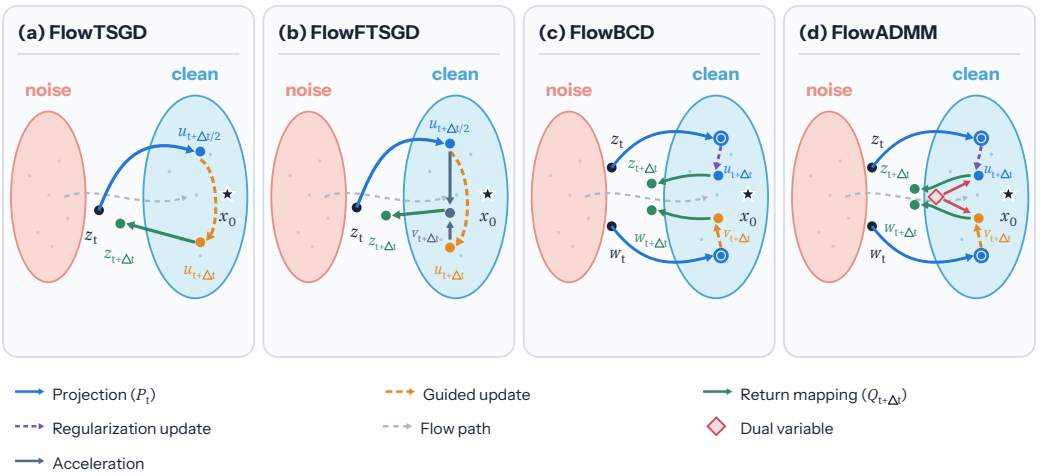

Figure 1: Flowcharts of the FlowTSGD, FlowFTSGD, FlowBCD, and FlowADMM methods.

others. More recently, generative models such as diffusion models (Ho et al., 2020; Song et al., 2020) and flow matching (Lipman et al., 2022; Liu et al., 2022) have been introduced for image inverse problems, leveraging unconditionally pretrained generative models to address conditional generation tasks. Such methods correct the per-step sampling of a stochastic process conditioned on an observation (Song et al., 2022; Meng & Kabashima, 2025; He et al., 2023; Tao et al., 2026). Furthermore, a variety of optimization methods that integrate diffusion models or flow matching have been developed (Zhu et al., 2023; Song et al., 2024; Patel et al., 2025; Martin et al., 2025; Pourya et al., 2026; Cai et al., 2026). In this work, we propose flow-driven optimization methods for image inverse problems and interpret the velocity field as the negative gradient of the regularization term. Our contributions are summarized as follows:

We introduce a novel regularization term, which takes the form of a distance-based function. The negative gradient of this regularization term corresponds to the velocity field of the flow matching model.

Any gradient-related optimization methods can be applied to solve optimization problems involving this new regularization term. When the data consistency term is differentiable, these optimization problems can be solved using gradient-related methods. We propose flow-driven two-step gradient descent (FlowTSGD), a flow-driven fast version of TSGD (FlowFTSGD), flow-driven BCD (FlowBCD), and flow-driven ADMM (FlowADMM) algorithms. The illustrations of the FlowTSGD, FlowFTSGD, FlowBCD and FlowADMM methods are in Figure 1.

We evaluate these gradient-related methods numerically. We conduct experiments on four inverse problems across three datasets. Numerical results demonstrate that FlowDPS, FlowBCD, and FlowADMM achieve superior performance, with FlowBCD and FlowADMM standing out in particular.

The remainder of this paper is structured as follows: Section 2 reviews those related works. Section 3 defines distance functions and establishes the relationship between distance function and flow matching models. Based on the theoretical result in Section 3, Section 4 gives flow-driven optimization methods. Numerical implementations are discussed in Section 5. Finally, we conclude with our findings and highlight the limitations of this work in Section 6.

2 Related Works

Optimization-based image restoration combines data consistency with image priors to recover images from degraded observations. Classical methods encode smoothness, sparsity, or edge structure through explicit regularizers. Tikhonov regularization stabilizes reconstruction using a quadratic penalty (Tikhonov & Arsenin, 1977), while total variation promotes sparse image gradients and preserves edges (Rudin et al., 1992). Transform-domain sparse regularization encourages sparse coefficients in a chosen image representation (Jafarpour et al., 2009). Iterative Shrinkage-Thresholding

Algorithm (ISTA) combines gradient steps on the data consistency term with proximal updates for the regularizer (Blumensath & Davies, 2008), and a fast ISTA (FISTA) accelerates convergence under standard convexity and smoothness assumptions (Beck & Teboulle, 2009).

Since simple hand-crafted regularizers capture only limited aspects of natural image statistics, denoisers provide an alternative means of incorporating richer prior information. The Plug-and-Play (PnP) method replaces the proximal update associated with the regularizer by an existing denoiser, without requiring an explicit regularization function (Venkatakrishnan et al., 2013). The Regularization by Denoising (RED) method defines a denoiser-based regularizer and employs the denoising residual as a regularizing correction (Romano et al., 2017). In both frameworks, reconstruction quality still depends on the quality of the denoiser. When neural denoisers are trained on specific image distributions and noise conditions, generalization to other degradation types or noise levels becomes difficult. For general denoisers, both convergence and optimization interpretation depend on the properties of the denoiser (Reehorst & Schniter, 2019).

Generative priors have been successfully employed for image inverse problems, integrating unconditional diffusion models (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2020) or flow matching (Lipman et al., 2022; Liu et al., 2022) for conditional tasks. Motivated by the generative process in denoising diffusion probabilistic models (DDPM), a number of filtering-based methods have been proposed, such as iterative latent variable refinement (ILVR) (Choi et al., 2021), filtering posterior sampling (FPS) (Dou & Song, 2024), and frequency-guided posterior Sampling (FGPS) (Thaker et al., 2025), among others. Additional methods are based on Tweedie's formula. The manifold constrained gradient (MCG) method introduces correction terms inspired by manifold constraints so as to keep the iterative process close to the original data manifold (Chung et al., 2022b). In pseudo-inverse guided diffusion (ΠGDM), Song et al. (2022) adopt pseudo-inverse guidance to invert the measurement model, thereby improving approximation accuracy. In diffusion posterior sampling (DPS), Chung et al. (2022a) employ the predicted clean image to approximate the measurement likelihood at each noisy state and incorporate the gradient of the approximate log-likelihood into reverse sampling. FlowDPS employs a flow analogue of the Tweedie formula to express the sampling update in terms of clean image and noise estimates (Kim et al., 2025). Analogous to DDIM, the Denoising Diffusion Restoration Models (DDRM) construct a non-Markovian process that enables flexible skip-step sampling while preserving conditioning for solving inverse problems (Kawar et al., 2022). Rather than the Tweedie's formula, Tao et al. (2026) adopt a novel MAP-based method to update the guided term by incorporating a Gaussian-type prior of natural images. Because these pixel-space methods can be computationally expensive for high-resolution restoration, Rombach et al. (2022) propose a latent diffusion model (LDM), which reduces the cost of generation by performing diffusion in the compressed latent space of an autoencoder. They also design a general-purpose conditioning mechanism based on cross-attention, enabling multi-modal training. Rout et al. (2023) present a framework for solving linear inverse problems that leverages pretrained latent diffusion models. Their theoretical analysis establishes provable sample recovery in a linear model setting.

More recently, a variety of optimization methods that integrate diffusion models or flow matching have been developed. Based on the hypothesis of a $(d-1)$-dimensional linear subspace manifold, He et al. (2023) formulate an optimization problem involving tangent spaces and propose manifold preserving guided diffusion (MPGD). DiffPIR incorporates half-quadratic splitting into diffusion sampling within a plug-and-play framework (Zhu et al., 2023). It alternates between prior-based denoising and data consistency optimization, correcting the denoised estimate through a data proximal subproblem before adding noise to continue sampling. ReSample (Song et al., 2024) is an algorithm for solving general inverse problems with pretrained latent diffusion models, in which data consistency is enforced by solving an optimization problem during the reverse sampling process. In FlowChef (Patel et al., 2025), the authors derive the mathematical relationship between the gradients of rectified flow ODEs by leveraging the straightness and smooth Jacobian properties, extending the theoretical findings to solve linear inverse problems. PnP-Flow (Martin et al., 2025) uses a pretrained flow matching model and alternates between gradient descent steps on the data-fidelity term, reprojections onto the learned flow matching path, and denoising. Flower (Pourya et al., 2026) similarly combines clean image correction with renoising, but uses a proximal update of the predicted clean endpoint rather than the data consistency gradient step in PnP-Flow. This proximal update is well suited to linear inverse problems, but may be difficult to extend to nonlinear inverse problems. Cai et al. (2026) reformulate classifier-free guidance (CFG) sampling as a homotopy optimization

with a manifold constraint. To improve computational efficiency and stability, they further enhance this iterative process with Anderson acceleration without requiring additional model evaluations.

3 DISTANCE FUNCTION

Let $\mathcal{M}$ be a data manifold. We first define a distance function to $\mathcal{M}$.

Definition 3.1. *Let $\mathcal{M} \subseteq \mathbb{R}^d$. Then the distance of $\boldsymbol{z} \in \mathbb{R}^d$ to $\mathcal{M}$ can be defined as*

$$\mathrm{dist}_{\mathcal{M}}(\boldsymbol{z}) = \inf_{\boldsymbol{u}\in\mathcal{M}} \|\boldsymbol{z}-\boldsymbol{u}\|.$$

Corresponding to $\frac{1}{2}\mathrm{dist}^2_{\mathcal{M}}(\boldsymbol{z})$, we define a smoothed squared-distance function with a smoothing parameter $\sigma > 0$:

$$f_{\mathcal{M}}(\boldsymbol{z},\sigma) = -\sigma^2 \log\left(\int_{\boldsymbol{u}\in\mathcal{M}} \exp\left(-\frac{\|\boldsymbol{z}-\boldsymbol{u}\|^2}{2\sigma^2}\right)\right).$$

$f_{\mathcal{M}}(\boldsymbol{z},\sigma)$ is a smooth approximation of $\frac{1}{2}\mathrm{dist}^2_{\mathcal{M}}(\boldsymbol{z})$. Without loss of generality, we assume the data is uniformly distributed on $\mathcal{M}$, then $\lim_{\sigma\to 0} f_{\mathcal{M}}(\boldsymbol{z},\sigma) = \frac{1}{2}\mathrm{dist}^2_{\mathcal{M}}(\boldsymbol{z})$. Compared with $\frac{1}{2}\mathrm{dist}^2_{\mathcal{M}}(\boldsymbol{z})$, the function $f_{\mathcal{M}}(\boldsymbol{z},\sigma)$ is always differentiable, which is suitable for gradient-related methods. Next, we will establish the relationship between the above distance function and the flow matching model, where the velocity field of the flow matching model exactly corresponds to the gradient of a distance-based function. We have the following theorem.

Theorem 3.1. *Let $\mathcal{M}$ be a data manifold. For any $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$ and $\boldsymbol{u} \in \mathcal{M}$, define $\boldsymbol{z} = a_t\boldsymbol{u} + b_t\epsilon$, where $a_t, b_t \geq 0$. Let $v_\theta(\boldsymbol{z}, t)$ be the pre-trained neural network to estimate the velocity field in the flow matching model (see the overview in Appendix A). Set $\ell_t(\boldsymbol{z}) = \mu_t f_{\mathcal{M}}(c_t\boldsymbol{z}, \sigma_t) - \frac{\eta_t}{2}\|\boldsymbol{z}\|^2$. When the neural network is optimal trained–that is, when $\int_0^1 \mathbb{E}_\epsilon \mathbb{E}_{\boldsymbol{u}} \|\dot{a}_t\boldsymbol{u} + \dot{b}_t\epsilon + v_\theta(a_t\boldsymbol{u} + b_t\epsilon, t)\|^2 dt = 0$, it follows that*

$$\nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z}) = -v_\theta(\boldsymbol{z}, t),$$

where

$$c_t = \frac{1}{a_t}, \quad \sigma_t = \frac{b_t}{a_t}, \quad \mu_t = \frac{a_t}{b_t}(a_t\dot{b}_t - \dot{a}_t b_t), \quad \eta_t = \frac{\mu_t}{a_t^2} - \frac{\dot{b}_t}{b_t}.$$

In practice, $\int_0^1 \mathbb{E}_\epsilon \mathbb{E}_{\boldsymbol{u}} \|\dot{a}_t\boldsymbol{u} + \dot{b}_t\epsilon + v_\theta(a_t\boldsymbol{u} + b_t\epsilon, t)\|^2 dt > 0$ is a small constant after training, $v_\theta(\boldsymbol{z}_t, t)$ approximates the negative gradients of the function $\ell_t(\boldsymbol{z})$ with respect to $\boldsymbol{z}$.

Proof. The proof is deferred to Appendix B. □

4 FLOW-DRIVEN OPTIMIZATION METHODS

As shown in the previous section, the pretrained neural network approximates the negative gradients of the function $\ell_t(\boldsymbol{z})$ with respect to $\boldsymbol{z}$. That is, $\nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z}) \approx -v_\theta(\boldsymbol{z}, t)$. We now turn to employing this function as a regularizer in the context of image inverse problems. Since $\boldsymbol{z}$ lies on the path connecting $\boldsymbol{u}$ and ϵ, where $\boldsymbol{u} \in \mathcal{M}$ belongs to the data manifold $\mathcal{M}$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$, define $\mathcal{M}_t = \{\boldsymbol{z} | \boldsymbol{z} = a_t\boldsymbol{u} + b_t\epsilon, \boldsymbol{u} \in \mathcal{M}, \epsilon \sim \mathcal{N}(\mathbf{0}, I)\}$ and $\mathcal{M}_{0,t} = \{\boldsymbol{u} | \epsilon \sim \mathcal{N}(\mathbf{0}, I), a_t\boldsymbol{u} + b_t\epsilon \in \mathcal{M}_t\}$. Here we require $a_0 = b_1 = 0$ to ensure $\mathcal{M}_0 = \{\epsilon | \epsilon \sim \mathcal{N}(\mathbf{0}, I)\}$ and $\mathcal{M}_1 = \mathcal{M}$. It is obvious that $\mathcal{M}_1 \subseteq \mathcal{M}_{0,t}$. When $t = 1$, $\mathcal{M}_1 = \mathcal{M}_{0,t}$.

First, we define the Maps between $\mathcal{M}_t$ and $\mathcal{M}_{0,t}$. The following definition is based on the Tweedie's Formula (Kim et al., 2025, Proposition 1).

Definition 4.1. *Let $\mathcal{P}_t : \mathcal{M}_t \mapsto \mathcal{M}_{0,t}$ be a denoising map from $\mathcal{M}_t$ to $\mathcal{M}_{0,t}$ and $\mathcal{Q}_{t+\Delta t} : \mathcal{M}_{0,t} \times \mathcal{M}_t \mapsto \mathcal{M}_{t+\Delta t}$ be a noise map from $\mathcal{M}_{0,t} \times \mathcal{M}_t$ back to $\mathcal{M}_{t+\Delta t}$. Then*

$$\mathcal{P}_t(\boldsymbol{z}) = \mathbb{E}[\boldsymbol{u}|\boldsymbol{z}] = \left[a_t - \dot{a}_t\frac{b_t}{\dot{b}_t}\right]^{-1}\left(\boldsymbol{z} - \frac{b_t}{\dot{b}_t}v_\theta(\boldsymbol{z}, t)\right), \ \forall \boldsymbol{z} \in \mathcal{M}_t$$

$$\mathcal{Q}_{t+\Delta t}(\boldsymbol{u}, \boldsymbol{z}) = a_{t+\Delta t}\boldsymbol{u} + b_{t+\Delta t}E_t(\boldsymbol{z}), \ \forall \boldsymbol{u} \in \mathcal{M}_{0,t}, \boldsymbol{z} \in \mathcal{M}_t,$$

where $E_t(\boldsymbol{z}) = \mathbb{E}[\epsilon|\boldsymbol{z}] = \left[b_t - \dot{b}_t\frac{a_t}{\dot{a}_t}\right]^{-1}\left(\boldsymbol{z} - \frac{a_t}{\dot{a}_t}v_\theta(\boldsymbol{z}, t)\right)$. Numerically, $E_t(\boldsymbol{z})$ is replaced by $\tilde{E}_t(\boldsymbol{z}) = \gamma_t E_t(\boldsymbol{z}) + (1-\gamma_t)\epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, I)$ and $\gamma_t \in [0, 1]$.

A number of gradient-related optimization methods can be applied to solve image inverse problems, including Two-Step Gradient Descent (TSGD), Block Gradient Descent (BCD), and Alternating Direction Method of Multipliers (ADMM). Next, we formulate the flow-driven versions of those optimization methods. For any $t \in [0,1]$, any $\boldsymbol{u} \in \mathcal{M}_{0,t}$, set $\boldsymbol{z} = a_t\boldsymbol{u} + b_t e_t \in \mathcal{M}_t$, with given $\boldsymbol{z}_{t-\Delta t} \in \mathcal{M}_{t-\Delta t}$ and $e_t = E_{t-\Delta t}(\boldsymbol{z}_{t-\Delta t})$. Let $R_t(\boldsymbol{u}) := \frac{b_t}{a_t}\ell_t(\boldsymbol{z}) + \frac{1}{2(1-a_t)\xi_t}\|\boldsymbol{u}-\boldsymbol{z}\|^2 = \frac{\mu_t a_t}{b_t} f_{\mathcal{M}_{0,t}}(c_t\boldsymbol{z}, \sigma_t) - \frac{\eta_t}{2}\|\boldsymbol{z}\|^2 + \frac{1}{2(1-a_t)\xi_t}\|\boldsymbol{u}-\boldsymbol{z}\|^2$, then

$$\begin{aligned}\nabla_{\boldsymbol{u}} R_t(\boldsymbol{u}) =& \frac{b_t}{a_t}\nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z})\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{u}} + \frac{1}{(1-a_t)\xi_t}(\boldsymbol{u}-\boldsymbol{z})\left(1 - \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{u}}\right) \\ =& b_t\nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z}) + \frac{1}{(1-a_t)\xi_t}(\boldsymbol{u}-\boldsymbol{z})(1-a_t) \approx -b_t v_\theta(\boldsymbol{z}, t) + \frac{1}{\xi_t}(\boldsymbol{u}-\boldsymbol{z}).\end{aligned}$$

To this end, we formulate a path of optimization problems parameterized by t:

$$\min_{\boldsymbol{u}\in\mathcal{M}_{0,t}} R_t(\boldsymbol{u}) + \lambda_t F(\boldsymbol{u}),\ \forall t \in [0,1], \tag{4.1}$$

where $F(\boldsymbol{u})$ is the data-fidelity term with $\lambda_t > 0$ and $R_t(\boldsymbol{u})$ is the regularization term associated with the flow matching models.

4.1 Flow-driven TSGD

To solve Equation (4.1), we sample $\boldsymbol{z}_0 \sim \mathcal{M}_0$ and $\boldsymbol{u}_0 = \mathcal{P}_0(\boldsymbol{z}_0)$. The iterative scheme for Flow-driven TSGD (denoted as FlowTSGD) is as follows:

$$\begin{cases} \boldsymbol{u}_{t+\frac{\Delta t}{2}} = \boldsymbol{u}_t - \xi_t\nabla_{\boldsymbol{u}} R_t(\boldsymbol{u}_t) = \boldsymbol{z}_t - \xi_t b_t\nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z}_t), \\ \boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_{t+\frac{\Delta t}{2}} - \zeta_t\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_{t+\frac{\Delta t}{2}}), \end{cases} \tag{4.2}$$

where $\xi_t = 1$, $\zeta_t > 0$ and

$$\boldsymbol{z}_t = \begin{cases} \boldsymbol{z}_0 \sim \mathcal{M}_0, & t = 0, \\ \mathcal{Q}_t(\boldsymbol{u}_t, \boldsymbol{z}_{t-\Delta t}), & t = k\Delta t, k \in \{1, \cdots, 1/\Delta t\}. \end{cases}$$

The FlowChef method (Patel et al., 2025) exactly corresponds to FlowTSGD. we sample $\boldsymbol{z}_0 \sim \mathcal{M}_0$ and $\boldsymbol{v}_0 = \mathcal{P}_0(\boldsymbol{z}_0)$. An fast algorithm of Equation (4.2), denoted as FlowFTSGD, is as follows:

$$\begin{cases} \boldsymbol{u}_{t+\frac{\Delta t}{2}} = \boldsymbol{v}_t - \xi_t\nabla_{\boldsymbol{u}} R_t(\boldsymbol{v}_t) = \boldsymbol{z}_t - \xi_t b_t\nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z}_t), \\ \boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_{t+\frac{\Delta t}{2}} - \zeta_t\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_{t+\frac{\Delta t}{2}}), \\ \boldsymbol{v}_{t+\Delta t} = \tau_t\boldsymbol{u}_{t+\frac{\Delta t}{2}} + (1-\tau_t)\boldsymbol{u}_{t+\Delta t}, \end{cases} \tag{4.3}$$

where $\xi_t = 1$, $\zeta_t > 0$, $\tau_t \in [0,1]$ and

$$\boldsymbol{z}_t = \begin{cases} \boldsymbol{z}_0 \sim \mathcal{M}_0, & t = 0, \\ \mathcal{Q}_t(\boldsymbol{v}_t, \boldsymbol{z}_{t-\Delta t}), & t = k\Delta t, k \in \{1, \cdots, 1/\Delta t\}. \end{cases}$$

Equation (4.3) corresponds to FlowDPS (Kim et al., 2025) or PnP-Flow (Martin et al., 2025) with appropriate values of τ_t.

4.2 Flow-driven BCD and ADMM

We split the variables and introducing an auxiliary variable, the problem in Equation (4.1) is reformulated as follows:

$$\min_{\boldsymbol{u},\boldsymbol{v}\in\mathcal{M}_{0,t}} R_t(\boldsymbol{u}) + \lambda_t F(\boldsymbol{v}), \quad \text{s.t. } \boldsymbol{u} = \boldsymbol{v}, \quad \forall t \in [0,1], \tag{4.4}$$

Here, we focus on the flow-driven formulations of BCD and ADMM, termed FlowBCD and FlowADMM, which are detailed as follows. For any $\rho_t > 0$, we first convert Equation (4.4) to an unconditional optimization problem:

$$\min_{\boldsymbol{u},\boldsymbol{v}\in\mathcal{M}_{0,t}} R_t(\boldsymbol{u}) + \lambda_t F(\boldsymbol{v}) + \rho_t\|\boldsymbol{u}-\boldsymbol{v}\|^2/2. \tag{4.5}$$

We sample $\boldsymbol{z}_0, \boldsymbol{w}_0$ from $\mathcal{M}_0$. The iterative scheme for Flow-driven BCD (denoted as FlowBCD) is as follows:

$$\begin{cases} \boldsymbol{u}_t = \mathcal{P}_t(\boldsymbol{z}_t), \\ \boldsymbol{v}_t = \mathcal{P}_t(\boldsymbol{w}_t), \\ \boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t - \xi_t \nabla_{\boldsymbol{u}} R_t(\boldsymbol{u}_t) - \xi_t \rho_t(\boldsymbol{u}_t - \boldsymbol{v}_t), \\ \boldsymbol{v}_{t+\frac{\Delta t}{2}} = \boldsymbol{v}_t - \zeta_t \left[\lambda_t \nabla_{\boldsymbol{v}} F(\boldsymbol{v}_t) + \rho_t(\boldsymbol{v}_t - \boldsymbol{u}_{t+\Delta t})\right], \\ \boldsymbol{v}_{t+\Delta t} = \tau_t \boldsymbol{v}_{t+\frac{\Delta t}{2}} + (1-\tau_t)\boldsymbol{v}_t, \\ \boldsymbol{z}_{t+\Delta t} = \mathcal{Q}_{t+\Delta t}(\boldsymbol{u}_{t+\Delta t}, \boldsymbol{z}_t), \\ \boldsymbol{w}_{t+\Delta t} = \mathcal{Q}_{t+\Delta t}(\boldsymbol{v}_{t+\Delta t}, \boldsymbol{w}_t). \end{cases}$$

Similarly, we convert Equation (4.4) to an optimization problem with an augmented Lagrangian objective function for any $\rho_t > 0$:

$$\min_{\boldsymbol{u},\boldsymbol{v} \in \mathcal{M}_{0,t}, \Lambda} R_t(\boldsymbol{u}) + \lambda_t F(\boldsymbol{v}) + \langle \Lambda, \boldsymbol{v} - \boldsymbol{u} \rangle + \rho_t \|\boldsymbol{v} - \boldsymbol{u}\|^2/2. \tag{4.6}$$

We sample $\boldsymbol{z}_0, \boldsymbol{w}_0$ from $\mathcal{M}_0$ and give the value of Λ_0. The iterative scheme for the Flow-driven ADMM (denoted as FlowADMM) is as follows:

$$\begin{cases} \boldsymbol{u}_t = \mathcal{P}_t(\boldsymbol{z}_t), \\ \boldsymbol{v}_t = \mathcal{P}_t(\boldsymbol{w}_t), \\ \boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t - \xi_t \nabla R_t(\boldsymbol{u}_t) - \xi_t \rho_t(\boldsymbol{u}_t - \boldsymbol{v}_t) + \xi_t \Lambda_t, \\ \boldsymbol{v}_{t+\Delta t} = \boldsymbol{v}_t - \zeta_t \left[\lambda_t \nabla_{\boldsymbol{v}} F(\boldsymbol{v}_t) + \rho_t(\boldsymbol{v}_t - \boldsymbol{u}_{t+\Delta t})\right] - \zeta_t \Lambda_t, \\ \boldsymbol{v}_{t+\Delta t} = \tau_t \boldsymbol{v}_{t+\frac{\Delta t}{2}} + (1-\tau_t)\boldsymbol{v}_t, \\ \Lambda_{t+1} = \Lambda_t + \rho_t(\boldsymbol{v}_{t+\Delta t} - \boldsymbol{u}_{t+\Delta t}), \\ \boldsymbol{z}_{t+\Delta t} = \mathcal{Q}_{t+\Delta t}(\boldsymbol{u}_{t+\Delta t}, \boldsymbol{z}_t), \\ \boldsymbol{w}_{t+\Delta t} = \mathcal{Q}_{t+\Delta t}(\boldsymbol{v}_{t+\Delta t}, \boldsymbol{w}_t). \end{cases}$$

Remark 4.1. *In the two iterative algorithms above, the operator $\mathcal{P}_t$ maps from $\mathcal{M}_t$ to $\mathcal{M}_{0,t}$, while the operator $\mathcal{Q}_{t+\Delta t}(\cdot,\cdot)$ maps the one-step update of $\boldsymbol{u}_t, \boldsymbol{v}_t$ onto the manifold $\mathcal{M}_{t+\Delta t}$, thereby ensuring better satisfaction of the constraints $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{M}_{0,t+\Delta t}$.*

5 Numerical Implements

Table 1: Quantitative results for the Super-Resolution with the $\times 12$ Avgpool and Bicubic downsamplings. The best results are highlighted in bold, while the second-best results are underlined.

Method	(a) Super-Resolution							
	Avgpool $\times 12$				Bicubic $\times 12$			
	PSNR↑	SSIM↑	FID↓	LPIPS↓	PSNR↑	SSIM↑	FID↓	LPIPS↓
AFHQ (1,000 images, 768×768)								
FlowChef	23.93	0.641	21.14	0.249	24.79	0.633	21.31	0.256
FlowDPS	<u>25.88</u>	0.661	**16.85**	0.198	<u>26.07</u>	0.662	**15.71**	<u>0.188</u>
FlowBCD	**26.88**	<u>0.722</u>	19.52	<u>0.182</u>	**27.14**	<u>0.723</u>	18.84	**0.174**
FlowADMM	**26.88**	**0.723**	<u>19.42</u>	**0.181**	**27.14**	**0.724**	<u>18.65</u>	**0.174**
FFHQ (1,000 images, 768×768)								
FlowChef	25.40	0.696	41.50	0.218	25.31	0.692	39.75	0.224
FlowDPS	**26.81**	0.703	**33.78**	**0.159**	**27.00**	0.702	**33.75**	**0.154**
FlowBCD	26.65	<u>0.728</u>	34.69	<u>0.161</u>	26.86	<u>0.727</u>	36.86	<u>0.158</u>
FlowADMM	<u>26.69</u>	**0.731**	<u>34.43</u>	<u>0.161</u>	<u>26.90</u>	**0.730**	<u>36.31</u>	<u>0.158</u>
DIV2K (800 images, 768×768)								
FlowChef	18.02	0.495	101.50	0.355	15.12	0.441	123.40	0.433
FlowDPS	20.84	0.488	58.29	**0.258**	<u>21.10</u>	0.492	**56.19**	**0.252**
FlowBCD	**20.98**	<u>0.508</u>	<u>55.44</u>	0.262	**21.26**	<u>0.511</u>	57.47	0.259
FlowADMM	<u>20.97</u>	**0.509**	**54.71**	<u>0.261</u>	**21.26**	**0.512**	<u>56.55</u>	<u>0.258</u>

In this section, we compare a family of flow-driven optimization methods, namely FlowTSGD, FlowFTSGD, FlowBCD, and FlowADMM. The main results are presented in Subsection 5.1. Ablation studies and computational complexity are provided in Subsections 5.2 and 5.3, respectively. Owing to space constraints, the experimental setups are deferred to Appendix C.1.

Table 2: Quantitative results for the Gaussian and Motion deblurring. The best results are highlighted in bold, while the second-best results are underlined.

Method	(b) Deblurring							
	Gaussian				Motion			
	PSNR↑	SSIM↑	FID↓	LPIPS↓	PSNR↑	SSIM↑	FID↓	LPIPS↓
AFHQ (1,000 images, 768×768)								
FlowChef	22.75	0.703	36.46	0.267	26.16	**0.732**	39.19	0.246
FlowDPS	25.33	0.653	**23.00**	0.238	25.61	0.657	**19.99**	0.222
FlowBCD	**26.34**	<u>0.714</u>	<u>24.39</u>	**0.220**	**26.20**	0.712	<u>22.73</u>	**0.211**
FlowADMM	<u>26.31</u>	**0.715**	24.76	<u>0.221</u>	<u>26.18</u>	<u>0.713</u>	22.95	<u>0.212</u>
FFHQ (1,000 images, 768×768)								
FlowChef	23.14	**0.767**	112.80	0.251	23.14	**0.772**	104.70	0.264
FlowDPS	**26.23**	0.702	41.31	0.197	**26.62**	0.699	38.14	**0.182**
FlowBCD	26.09	0.721	**40.34**	**0.195**	26.14	0.719	**36.89**	<u>0.187</u>
FlowADMM	<u>26.11</u>	<u>0.723</u>	<u>40.67</u>	<u>0.196</u>	<u>26.16</u>	<u>0.722</u>	<u>37.27</u>	0.188
DIV2K (800 images, 768×768)								
FlowChef	19.54	**0.520**	98.27	0.343	19.80	**0.546**	88.86	0.313
FlowDPS	20.46	0.473	72.13	0.319	21.04	0.495	62.68	0.291
FlowBCD	**20.65**	<u>0.494</u>	**60.76**	**0.302**	**21.08**	<u>0.522</u>	**56.04**	**0.270**
FlowADMM	<u>20.62</u>	<u>0.494</u>	<u>61.05</u>	<u>0.303</u>	<u>21.05</u>	<u>0.522</u>	<u>56.26</u>	<u>0.272</u>

5.1 Main Results

Tables 1 and 2 report quantitative results for four tasks across three datasets. As FlowChef and FlowDPS are exact counterparts of FlowTSGD and FlowFTSGD, respectively, we adopt the numerical results of FlowChef and FlowDPS in their place. The numerical results demonstrate that FlowDPS, FlowBCD, and FlowADMM attain superior performance.

FlowBCD and FlowADMM achieve competitive results across datasets, with trade-offs that depend on the task and metric. On AFHQ, both rank first or second in PSNR and LPIPS across all four tasks, as well as in SSIM for super-resolution and Gaussian deblurring, while FlowDPS consistently achieves the lowest FID. On FFHQ, FlowBCD and FlowADMM achieve the two highest SSIM scores in super-resolution, whereas FlowDPS leads in PSNR, FID, and LPIPS. For deblurring on FFHQ, both achieve the two lowest FID scores and, under Gaussian blur, the two lowest LPIPS scores; FlowDPS and FlowChef lead in PSNR and SSIM, respectively. On DIV2K, both methods rank first or second in PSNR, FID, and LPIPS for deblurring and in PSNR and SSIM for super-resolution. FlowChef achieves the highest SSIM in deblurring, while FlowDPS obtains the lowest LPIPS in super-resolution.

We next turn to qualitative comparisons. Figure 2 presents the visual results for the $\times 12$ SR task with Avgpool downsampling on the AFHQ dataset, the $\times 12$ SR task with bicubic downsampling on the FFHQ dataset, as well as the Gaussian and motion deblurring tasks on the DIV2K dataset.

Compared with FlowDPS, FlowBCD and FlowADMM produce more distinct muzzle fur on AFHQ and preserve clear hair boundaries and collar patterns on FFHQ, with fewer visible artifacts around the green cord. On DIV2K, both recover clearer grass edges and head contours under Gaussian blur and more distinct body plumage under motion blur. Fine fur and wing textures still differ from the ground truth, however, showing that greater local sharpness does not necessarily imply accurate texture recovery.

5.2 Ablation Study

In this subsection, we explore the effects caused by the parameter ρ_t in the FlowBCD and FlowADMM methods. We focus on $\times 12$ SR task with Avgpool downsampling under 1,000 fixed dog images on the AFHQ dataset. Since $\rho_t = \rho_0 \sigma_t$ with σ_t being a fixed parameter in the unconditional flow model, we focus on the exploration of various values of $\rho_0 \in \{35, 40, 45\}$. For other parameters, we choose the same values as the experimental setup in Appendix C.1.

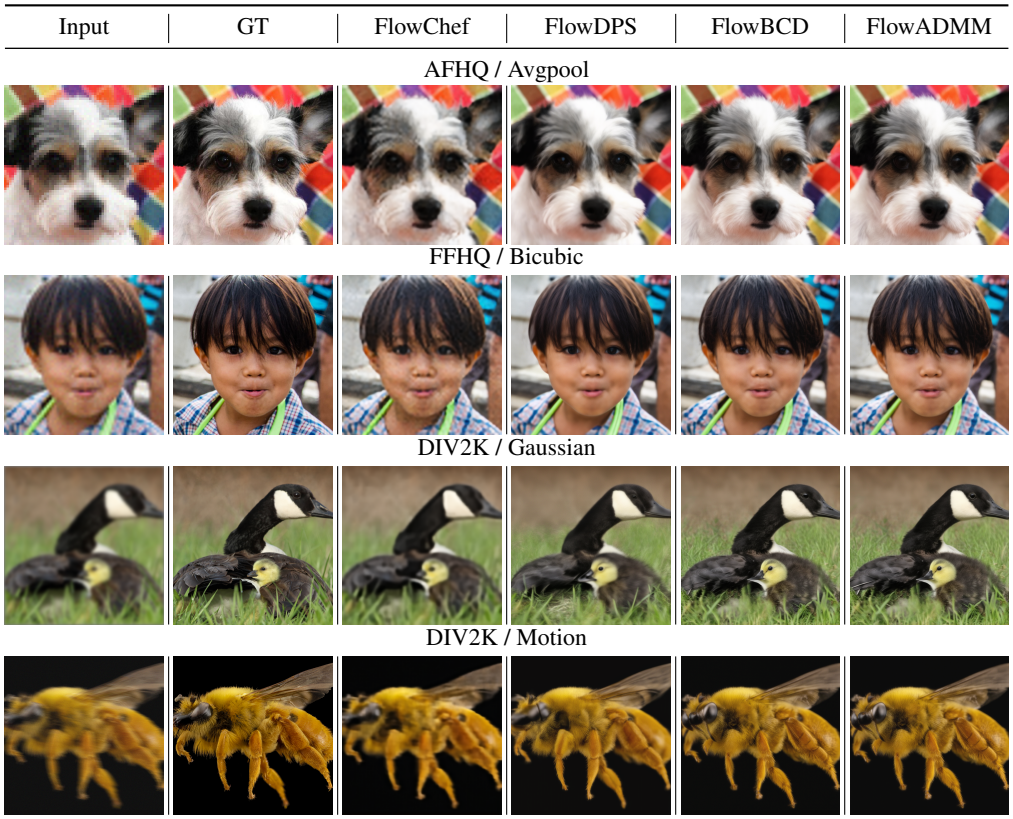

Figure 2: Qualitative comparisons. The first row presents $\times 12$ SR with Avgpool downsampling on the AFHQ dataset, and the second row presents $\times 12$ SR with bicubic downsampling on the FFHQ dataset. The third and fourth rows present Gaussian and motion deblurring on the DIV2K dataset, respectively, ordered from top to bottom. Each row displays the input, ground truth (GT), FlowChef, FlowDPS, FlowBCD, and FlowADMM images for the same sample.

Table 3: Ablation studies of ρ_0 for the $\times 12$ SR task with Avgpool downsampling under 1,000 dog images on the AFHQ dataset.

Method	ρ_0	PSNR↑	SSIM↑	FID↓	LPIPS↓	Method	ρ_0	PSNR↑	SSIM↑	FID↓	LPIPS↓
FlowBCD	35	**27.01**	**0.725**	**18.53**	0.186	FlowADMM	35	**27.03**	**0.727**	**18.55**	0.186
FlowBCD	40	26.88	0.722	19.52	**0.182**	FlowADMM	40	26.88	0.723	19.42	**0.181**
FlowBCD	45	26.47	0.706	23.03	0.190	FlowADMM	45	26.47	0.708	23.04	0.190

Table 3 evaluates the FlowBCD and FlowADMM methods using PSNR, SSIM, FID, and LPIPS. Results show that $\rho_0 = 35$ yields the highest PSNR and SSIM and the lowest FID for both methods, whereas $\rho_0 = 40$ achieves the lowest LPIPS. Increasing ρ_0 from 40 to 45 worsens all four metrics relative to 40, indicating that a larger penalty does not consistently improve reconstruction quality. The two methods obtain similar scores at each parameter value, and neither exhibits a consistent advantage across all metrics.

In addition, we illustrate the comparisons of the eye structure and fur recovery for the FlowBCD and FlowADMM in Figure 3, which correspond to three values of ρ_0.

As ρ_0 increases from 35 to 45, additional individual fur strands become visible in the illustrated images. Among the recoveries produced by FlowBCD, $\rho_0 = 40$ produces eye contours that resemble the ground truth more closely, whereas the additional fur detail at $\rho_0 = 45$ is not accompanied by a corresponding improvement in eye structure. A very similar trend can be observed in the recoveries produced by FlowADMM as ρ_0 varies from 35 to 45.

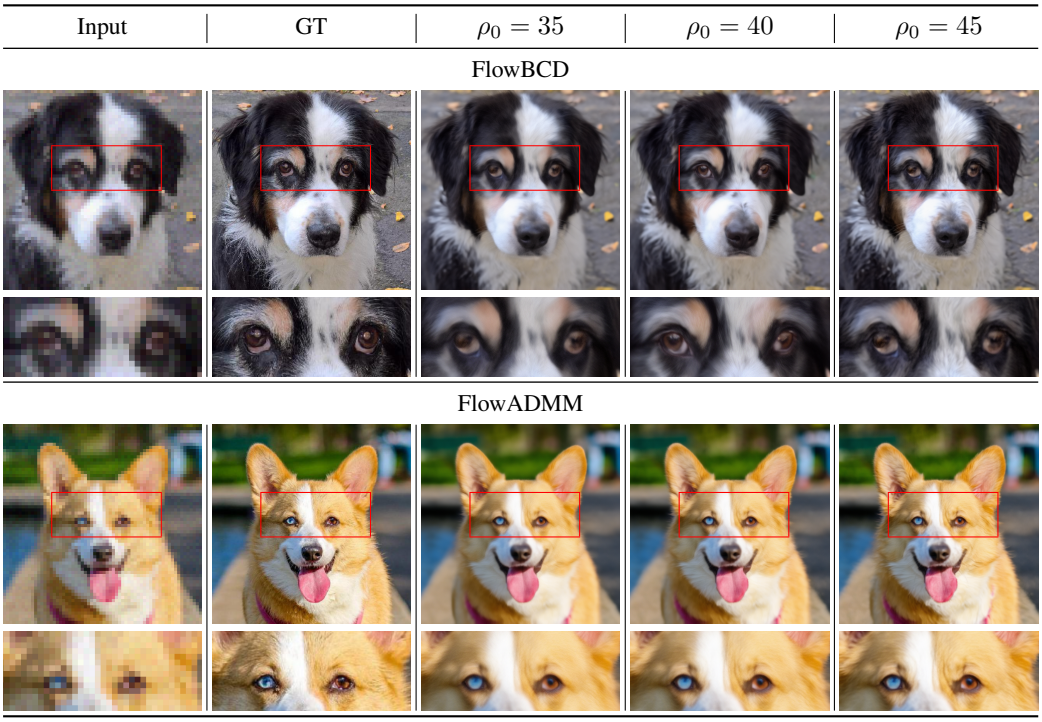

Figure 3: Effects of ρ_0 on FlowBCD and FlowADMM on the $\times 12$ SR task with the Avgpool downsampling (GT denotes ground truth). Red boxes indicate the eyes and surrounding fur, which correspond to the same region in all five columns.

5.3 Computational Complexity

Owing to space constraints, we present only the results here, while descriptions of the experiments and the data analysis are deferred to Appendix C.2.

Table 4: Comparisons of runtime and the Peak GPU memory on the Gaussian deblurring task under 100 DIV2K images at 768×768 resolution. Runtime is reported by mean and standard deviation (mean $\pm$ standard deviation); the Peak GPU memory is an average across images.

Method	Runtime (s)↓	Peak GPU memory (GiB)↓
FlowChef	7.481 ± 2.815	10.705
FlowDPS	17.479 ± 0.054	10.712
FlowBCD	19.475 ± 0.077	10.713
FlowADMM	19.680 ± 2.147	10.714

6 Conclusion

This paper proposes a framework of gradient-related optimization methods that incorporate the flow matching model for image inverse problems. We introduce a novel regularization term in the form of a distance-based function. From a theoretical perspective, we prove that the negative gradient of this regularization term coincides with the velocity field of the flow matching model. Consequently, any gradient-related optimization method can be applied to solve optimization problems involving this novel regularization term, provided that the data consistency term is differentiable. We propose four flow-driven gradient-related optimization methods and validate them numerically. Our methods mainly focus on image processing problems, and a valuable future direction is to extend them to video generation. Moreover, the maps between $\mathcal{M}_t$ and $\mathcal{M}_{0,t}$ are based on Tweedie's formula, and alternative mappings warrant further investigation.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing and assistance in creating the flowchart in Figure 1. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. doi: 10.1137/080716542.

Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5):629–654, 2008.

Jian-Feng Cai, Haixia Liu, Zhengyi Su, and Chao Wang. Improving classifier-free guidance of flow matching via manifold projection. *arXiv preprint arXiv:2601.21892*, 2026.

Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *International Conference on Computer Vision*, pp. 14347–14356, 2021.

Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022a.

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022b.

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.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Behnam Jafarpour, Vivek K Goyal, Dennis B McLaughlin, and William T Freeman. Transform-domain sparsity regularization for inverse problems in geosciences. *Geophysics*, 74(5):R69–R83, 2009.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

Jeongsol Kim, Bryan Sangwoo Kim, and Jong Chul Ye. Flowdps: Flow-driven posterior sampling for inverse problems. *arXiv preprint arXiv:2503.08136*, 2025.

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

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

Ségolène Martin, Anne Gagneux, Paul Hagemann, and Gabriele Steidl. PnP-Flow: Plug-and-play image restoration with flow matching. In *International Conference on Learning Representations*, 2025.

Xiangming Meng and Yoshiyuki Kabashima. Diffusion model based posterior sampling for noisy linear inverse problems. In *Asian Conference on Machine Learning*, pp. 623–638, 2025.

Maitreya Patel, Song Wen, Dimitris N. Metaxas, and Yezhou Yang. FlowChef: Steering of rectified flow models for controlled generations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Mehrsa Pourya, Bassam El Rawas, and Michael Unser. Flower: A flow-matching solver for inverse problems. In *International Conference on Learning Representations*, 2026.

Edward T. Reehorst and Philip Schniter. Regularization by denoising: Clarifications and new interpretations. *IEEE Transactions on Computational Imaging*, 5(1):52–67, 2019. doi: 10.1109/TCI.2018.2880326.

Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017. doi: 10.1137/16M1102884.

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*, pp. 10684–10695, 2022.

Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alexandros G. Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268, 1992. doi: 10.1016/0167-2789(92)90242-F.

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 *International Conference on Learning Representations*, 2024.

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

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2022.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Pingping Tao, Haixia Liu, and Jing Su. Map-based problem-agnostic diffusion model for inverse problems. *Signal Processing: Image Communication*, pp. 117532, 2026.

Darshan Thaker, Abhishek Goyal, and René Vidal. Frequency-guided posterior sampling for diffusion-based image restoration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12873–12882, 2025.

Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of Ill-Posed Problems*. Winston, Washington, DC, 1977.

Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pp. 945–948, 2013. doi: 10.1109/GlobalSIP.2013.6737048.

Rongyuan Wu, Tao Yang, Lingchen Sun, Zhengqiang Zhang, Shuai Li, and Lei Zhang. Seesr: Towards semantics-aware real-world image super-resolution. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25456–25467. IEEE, 2024.

Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1219–1229, 2023.

A FLOW MODELS

Flow Matching (Lipman et al., 2022), also referred to as rectified flow (Liu et al., 2022), is a generative modeling framework that learns to map a simple noise distribution to complex data by directly modeling the continuous evolution of data points over time. Given $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$ and $\boldsymbol{u} \in \mathcal{M}$ belongs to the data manifold $\mathcal{M}$, define a path $\boldsymbol{z} = a_t\boldsymbol{u} + b_t\epsilon$ for any $t \in [0,1]$, where $a_t, b_t \geq 0$ and $a_0 = b_1 = 0$. The rectified flow induced by $(\epsilon, \boldsymbol{u})$ is an ordinary differential equation (ODE) defined over time $t \in [0,1]$:

$$d\boldsymbol{z} = v(\boldsymbol{z}, t)\, dt,$$

which transforms $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$ into $\boldsymbol{u} \in \mathcal{M}$ when t varies from 0 to 1. The drift term $v : \mathbb{R}^d \times \mathbb{R}^1 \to \mathbb{R}^d$ is designed to align the flow with the direction $\dot{\boldsymbol{z}} = \dot{a}_t\boldsymbol{u} + \dot{b}_t\epsilon$ of the path connecting ϵ to $\boldsymbol{u}$. This is achieved by solving the following least squares problem:

$$\min_{\theta} \int_0^1 \mathbb{E}_{\epsilon,\boldsymbol{u}} \left[\|\dot{a}_t\boldsymbol{u} + \dot{b}_t\epsilon + v_\theta(a_t\boldsymbol{u} + b_t\epsilon, t)\|^2 \right] dt.$$

B PROOF OF THEOREM 3.1

Proof. Let

$$p(\boldsymbol{u}, \boldsymbol{z}, t) = \exp\left(-\frac{\|c_t\boldsymbol{z} - \boldsymbol{u}\|^2}{2\sigma_t^2} \right).$$

For

$$\ell_t(\boldsymbol{z}) = \mu_t f_{\mathcal{M}}(c_t\boldsymbol{z}, \sigma_t) - \frac{\eta_t}{2}\|\boldsymbol{z}\|^2 = -\mu_t\sigma_t^2 \log\left(\int_{\boldsymbol{u}} p(\boldsymbol{u}, \boldsymbol{z}, t) d\boldsymbol{u} \right) - \frac{\eta_t}{2}\|\boldsymbol{z}\|^2.$$

Our aim is to find a neural network $v_\theta(\boldsymbol{z}, t)$ such that $-v_\theta(\boldsymbol{z}, t) \approx \nabla_{\boldsymbol{z}}\ell_t(\boldsymbol{z})$ (that is, $\eta_t\boldsymbol{z} - v_\theta(\boldsymbol{z}, t) \approx \mu_t\nabla_{\boldsymbol{z}} f_{\mathcal{M}}(\boldsymbol{z})$). To train the neural network $v_\theta(\boldsymbol{z}, t)$, we use the following loss function $g_1(\theta)$ with $\kappa_t = \frac{1}{(\sqrt{2\pi}a_t)^d b_t}$, where

$$\begin{aligned}
g_1(\theta) &:= \int_0^1 \kappa_t dt \int_{\boldsymbol{z}} \|-v_\theta(\boldsymbol{z}, t) - (\mu_t\nabla_{\boldsymbol{z}} f_{\mathcal{M}}(c_t\boldsymbol{z}, \sigma_t) - \eta_t\boldsymbol{z})\|^2 \left[\int_{\boldsymbol{u}} p(\boldsymbol{u}, \boldsymbol{z}, t) d\boldsymbol{u} \right] p(\boldsymbol{z}|t) d\boldsymbol{z} \\
&= \int_0^1 \kappa_t dt \int_{\boldsymbol{z}} \left\| -v_\theta(\boldsymbol{z}, t) + \eta_t\boldsymbol{z} - \mu_t\sigma_t^2 \frac{\int_{\boldsymbol{u}} \left[p(\boldsymbol{u}, \boldsymbol{z}, t) \frac{\boldsymbol{z} - a_t\boldsymbol{u}}{b_t^2} \right] d\boldsymbol{u}}{\int_{\boldsymbol{u}} p(\boldsymbol{u}, \boldsymbol{z}, t) d\boldsymbol{u}} \right\|^2 \left[\int_{\boldsymbol{u}} p(\boldsymbol{u}, \boldsymbol{z}, t) d\boldsymbol{u} \right] p(\boldsymbol{z}|t) d\boldsymbol{z} \\
&\leq \int_0^1 \kappa_t \int_{\boldsymbol{z}} \left[\int_{\boldsymbol{u}} p(\boldsymbol{u}, \boldsymbol{z}, t) \left\| -v_\theta(\boldsymbol{z}, t) - \frac{\dot{b}_t\boldsymbol{z} + (\dot{a}_t b_t - a_t\dot{b}_t)\boldsymbol{u}}{b_t} \right\|^2 d\boldsymbol{u} \right] p(\boldsymbol{z}|t) d\boldsymbol{z} dt \\
&= \int_0^1 \int_{\boldsymbol{z}} \left[\int_{\boldsymbol{u}} \left\| \frac{\dot{b}_t\boldsymbol{z} + (\dot{a}_t b_t - a_t\dot{b}_t)\boldsymbol{u}}{b_t} + v_\theta(\boldsymbol{z}, t) \right\|^2 \kappa_t p(\boldsymbol{u}, \boldsymbol{z}, t) d\boldsymbol{u} \right] p(\boldsymbol{z}|t) d\boldsymbol{z} dt \\
&= \int_0^1 \iint \left\| \frac{\dot{b}_t\boldsymbol{z} + (\dot{a}_t b_t - a_t\dot{b}_t)\boldsymbol{u}}{b_t} + v_\theta(\boldsymbol{z}, t) \right\|^2 \frac{1}{(\sqrt{2\pi}b_t)^d} \exp\left[-\frac{\|\frac{\boldsymbol{z} - a_t\boldsymbol{u}}{b_t}\|^2}{2} \right] p(\boldsymbol{u}) \frac{1}{b_t} d\boldsymbol{z} d\boldsymbol{u} dt \\
&= \int_0^1 \iint \left\| \dot{a}_t\boldsymbol{u} + \dot{b}_t\epsilon + v_\theta(a_t\boldsymbol{u} + b_t\epsilon, t) \right\|^2 \frac{1}{(\sqrt{2\pi}b_t)^d} \exp\left[-\frac{\|\epsilon\|^2}{2} \right] p(\boldsymbol{u}) d\epsilon d\boldsymbol{u} dt \\
&= \int_0^1 \mathbb{E}_\epsilon \mathbb{E}_{\boldsymbol{u}} \left\| \dot{a}_t\boldsymbol{u} + \dot{b}_t\epsilon + v_\theta(a_t\boldsymbol{u} + b_t\epsilon, t) \right\|^2 dt := g_2(\theta).
\end{aligned}$$

When $g_2(\theta) = 0$, we have $g_1(\theta) = 0$, which implies $\eta_t\boldsymbol{z} - v_\theta(\boldsymbol{z}, t) = \nabla_{\boldsymbol{z}} f_{\mathcal{M}}(c_t\boldsymbol{z}, \sigma_t)$. In practice, $g_2(\theta) > 0$ is a small constant after training, we are reduced to the approximation alone $\eta_t\boldsymbol{z} - v_\theta(\boldsymbol{z}, t) \approx \nabla_{\boldsymbol{z}} f_{\mathcal{M}}(c_t\boldsymbol{z}, \sigma_t)$. □

C SUPPLEMENTARY EXPERIMENTS

C.1 EXPERIMENTAL SETUP

The experiments are performed on the AFHQ, FFHQ, and DIV2K datasets, comprising 1,000, 1,000, and 800 images, respectively. Across all datasets, images are resized and center-cropped to 768×768. To ensure a fair comparison, text conditioning follows the protocol of FlowDPS (Kim et al., 2025): the prompt is "a photo of a closed face of a dog" for AFHQ and "a photo of a closed face" for FFHQ. For DIV2K, we employ DAPE (Wu et al., 2024) to extract a distinct prompt from each degraded observation, with the prompt for each image held fixed across all methods and parameter settings.

For evaluation, we consider four image inverse tasks: $\times 12$ super-resolution with average pooling, $\times 12$ super-resolution with bicubic downsampling, Gaussian deblurring with 61×61 kernel and a standard deviation of 3.0, and motion deblurring with a 61×61 kernel and an intensity of 0.5. Gaussian noise with a standard deviation of 0.03 is added to all observations. Performance is measured in terms of pixel fidelity via PSNR↑ and SSIM↑, and perceptual quality via FID↓ and LPIPS↓.

As both FlowDPS and PnP-Flow constitute special cases of FlowFTSGD under suitable parameter choices, the results presented for FlowFTSGD in this work are identical to those of FlowDPS (Kim et al., 2025). For FlowBCD and FlowADMM, we adopt the pretrained Stable Diffusion 3 Medium model without further training. During reverse sampling, we employ 28 sampling steps, a CFG scale of 2.0, and a scheduler shift of 4.0. The step size is set to $\zeta_t = 20$, and the schedule is given by $\rho_t = \rho_0 \sigma_t$ with $\rho_0 = 40$, where σ_t represents the scheduler noise level.Here, σ_t decreases from 1 to 0 over the course of generation, which means that the penalty coefficient ρ_t also decreases accordingly. This schedule is consistent with the intuition that stronger data-fidelity guidance is needed in the early stages of generation, whereas the unconditional flow model should play a greater role in recovering fine image details at later stages. As for the update of the regularization term, it corresponds to the unconditional flow model, and the coefficient of ξ_t coincides exactly with the value specified by the flow model.

C.2 ADDITIONAL COMPUTATIONAL COMPLEXITY

Computational complexity is typically measured by the number of neural function evaluations (NFEs) in generative models. In generative model-driven image inverse problems, a correction is introduced based on an input. In the following, we use the runtime and the Peak GPU memory to evaluate the computational complexity.

We evaluate all four methods on an NVIDIA A100-SXM4-80GB GPU for Gaussian deblurring using 100 DIV2K images at 768×768 resolution and a batch size of one. We record the total time of the complete sampling function, including initialization, iterative updates, and final decoding, but exclude model loading, prompt precomputation, image loading, preprocessing, and result saving. We conduct CUDA synchronization before and after each timed call to ensure that measurements include all GPU operations within the sampling function.

In Table 4, We report the mean and standard deviation of per-image runtime and the average of the per-image peak GPU memory.

FlowChef incurs a shorter runtime. The primary difference between FlowChef and the remaining methods lies in the number of iterations for the data consistency term: FlowChef performs a single update, whereas the remaining methods perform three, yielding substantial time savings. In addition, an accelerated step (Equation (4.4), Line 4) is incorporated into FlowDPS. Although both FlowBCD and FlowADMM invoke the neural network twice per step, their mean runtimes exceed that of FlowDPS by only 11.4% and 12.6%, respectively. This additional cost stems primarily from the second forward evaluation of the flow model, while the decoding and gradient computations for data consistency are not doubled. Consequently, neural function evaluation counts alone do not reflect the full sampling cost.

For the Peak GPU memory, there is little variation across methods. The computations of the neural networks execute sequentially without retaining intermediate activations for backpropagation. The

additional latent variables, together with the dual variable in FlowADMM, occupy little memory relative to the overall footprint, consistent with the small increase in peak GPU memory usage.