STEERING RECTIFIED FLOW MODELS IN THE VECTOR FIELD FOR CONTROLLED IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing diffusion model-based methods for classifier guidance, inverse problems, and image editing often require additional training and are computationally expensive. We introduce **FlowChef**, a unified framework leveraging rectified flow models (RFMs) to guide the denoising process deterministically and without gradients. By exploiting the unique properties of RFMs, **FlowChef** achieves a significant reduction in computational overhead while maintaining high-quality image generation. This approach not only streamlines the process but also opens up new possibilities for real-time applications in image manipulation. **FlowChef** efficiently handles tasks such as inverse problems and image editing, achieving superior performance and setting new state-of-the-art benchmarks.

1 INTRODUCTION

Diffusion models have revolutionized AI-generated content, particularly in text-to-image (T2I) and text-to-video (T2V) applications Rombach et al. (2022); Saharia et al. (2022); Esser et al. (2024); Polyak et al. (2024); Patel et al. (2024a); Singer et al. (2023). While latent diffusion models (LDMs) Rombach et al. (2022) and classifier-free guidance Ho & Salimans (2022) have enabled remarkable advances in image editing and inverse problem solving, these methods incur high computational costs due to their stochastic nature and the need for numerous function evaluations (NFEs). However, the recent introduction of flow-based methods Lipman et al. (2023), especially rectified flow models (RFMs) Liu et al. (2023a); Lee et al. (2024), addresses these limitations to some extent by requiring fewer NFEs. Recent works have attempted to solve inverse problems by leveraging this property, focusing mainly on pixel models Ben-Hamu et al. (2024); Martin et al. (2024). While these approaches have improved computational time requirements, they are still not sufficiently efficient, as they require inversion and incur significant memory overhead. As a result, they cannot be extended to large state-of-the-art models like Flux or SD3 Esser et al. (2024).

In this paper, we introduce **FlowChef**, a novel method that significantly enhances controlled image generation by leveraging the unique characteristics of rectified flow models. We first standardize the objective of controlled synthesis, unifying various downstream tasks within a single framework. By revisiting the ordinary differential equations (ODEs) that govern these models, we analyze their error dynamics both theoretically and empirically. We discover that in nonlinear ODEs with stochasticity or trajectory crossovers, error terms emerge that hinder convergence due to inaccuracies in estimating denoised samples or improper gradient approximations.

Contrary to diffusion models, rectified flow models exhibit straight trajectories and avoid significant trajectory crossovers due to their linear interpolation between noise and data distributions (see Figure **??**(b-c)). We theoretically demonstrate and empirically validate that RFMs can achieve higher convergence rates without additional computational overhead by capitalizing on this key property. Building on this understanding, we present **FlowChef**, that proposes to steer the trajectories towards the target in the vector field by gradient skipping. This allows us to steer/navigate the vector field in a deterministic manner.

We conduct extensive evaluations of **FlowChef** across tasks such as pixel-level classifier guidance, image editing, and classifier-guided style transfer. Our results demonstrate that **FlowChef** not only surpasses baseline methods but does so with greater computational efficiency and without the need for inversion. **FlowChef** efficiently addresses a variety of tasks such as inverse problems, image editing, style transfer, etc. For perspective, **FlowChef** handles the linear inverse problems within

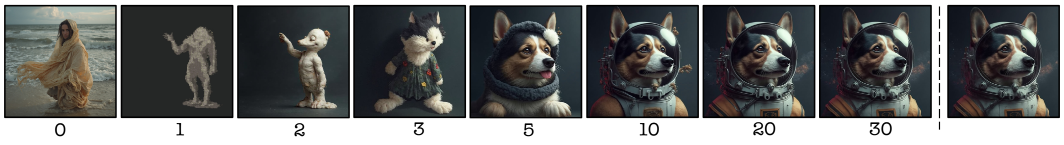


Figure 1: Illustration of impact of guided control step on Flux.1[Dev] with mean squared error as cost function ($\mathcal{L} = ||\hat{x}_0 - x_0^{ref}||_2^2$). This shows that **`FlowChef`** could guide the rectified flow models on the fly without requiring either the gradients through the Flux model or inversion. Importantly, the convergence speed is slowed down for illustration purposes.

18 seconds on the latent-space model, while SOTA takes 1-3 minutes per image. Furthermore, we explore its practical applicability to large-scale models to tackle both linear inverse problems and image editing together without inversion and within 30 NFEs at billions of parameter scales.

2 Proposed Method

In this section, we introduce our method, **`FlowChef`**, which enables free-form control for rectified flow models by presenting an efficient gradient approximation during guided sampling. We begin by analyzing the error dynamics of general ordinary differential equations (ODEs) and then explain how the inherent properties of rectified flow models mitigate existing approximation issues. Building on these insights, we derive **`FlowChef`**, an intuitive yet theoretically grounded approach for free-form controlled image generation applicable to various downstream tasks, including those involving pretrained latent models.

2.1 Error Dynamics of the ODEs

Understanding why existing methods often fail and require computationally intensive strategies is crucial. In ODE-based generative models, guiding the sampling process toward a desired target typically involves computing the gradient of a loss function with respect to the model's parameters or state variables. As noted in Eq. (6), even though the denoised output can be estimated using $\hat{x}_0 \leftarrow Sample(x_t, u_\theta(x_t, t))$, backpropagation through the ODE solver is still necessary to obtain $\nabla_{x_t}\mathcal{L}$. This raises the question: *Why is backpropagation through the ODE solver necessary?*

Approximating gradient computations is a common approach to reduce computational overhead He et al. (2024); Song et al. (2023). However, in models governed by nonlinear ODEs, unregulated gradient approximations can introduce significant errors into the system dynamics. This issue is formalized in the following proposition:

Proposition 2.1. *Let $p_1 \sim \mathcal{N}(0, \mathcal{I})$ be the noise distribution and p_0 be the data distribution. Let x_t denote an intermediate sample obtained from a predefined forward function q as $x_t = q(x_0, x_1, t)$, where $x_0 \sim p_0$ and $x_1 \sim p_1$. Define an ODE sampling process $dx(t) = f(x_t, t)dt$ and quadratic $\mathcal{L} = ||\hat{x}_0 - x_0^{ref}||_2^2$, where $f : \mathcal{R}^d \times [0, T] \to \mathcal{R}^d$ is an ODESolver. Then, the error dynamics of ODEs for controlled image generation is governed by:*

$$\frac{dE(t)}{dt} = -4sE(t) + 2e(t)^T \epsilon(t),$$

where $e(t) = \hat{x}_0 - x_0^{ref}$, $E(t) = e(t)^T e(t)$ is the squared error magnitude, $s > 0$ is the guidance strength, and $\epsilon(t)$ represents the accumulated errors due to non-linearity and trajectory crossovers.

The proof of Proposition 2.1 is provided in the Appendix D. The term $-4sE(t)$ denotes the exponential decay of error due to guidance, while $2e(t)^\top \epsilon(t)$ captures the impact of non-linearity and trajectory crossovers. In diffusion models, curved sampling trajectories lead to larger $\epsilon(t)$, hindering convergence. In contrast, rectified flow models exhibit straight trajectories with minimal crossovers, causing $\epsilon(t)$ to approach zero and allowing error to decrease exponentially.

To validate our findings, we conduct a toy study comparing classifier guidance on two ODE sampling methods using pretrained IDDPM and Rectified Flow++ (RF++) models on the ImageNet 64x64. As reported in Table 3 (Appendix), skipping the gradient in DDIM-based sampling increases the FID score, indicating significant $\epsilon(t)$. Conversely, RF++ converges well and improves the FID score. These empirical evidences further bolster our hypothesis that Rectified Flow models observe smooth vector field with the help of Proposition 2.1. Although backpropagating through the ODESolver further improves performance, it incurs higher computational costs as highlighted.

2.2 **FlowChef**: Steering Within the Vector Field

Rectified flow models inherently allow error dynamics to converge even with gradient approximations due to their straight-line trajectories and smooth vector fields, as discussed previously. Hence, vector field $u_\theta(x_t, t)$ is trained to be smooth, and this smoothness implies that u_θ changes gradually *w.r.t.* x_t. We formalize our approach with the following assumptions:

Assumption 1 (Local Linearity): Within the small neighborhoods around any point x_t along the sampling trajectory, the vector field $u_\theta(x_t, t)$ behaves approximately linearly with respect to x_t. Doing Taylor series expansion for small perturbations δ, we get:

$$u_\theta(x_t + \delta, t) \approx u_\theta(x_t, t) + J_{u_\theta}(x_t, t)\delta, \tag{1}$$

where $J_{u_\theta}(x_t, t) = \frac{du_\theta(x_t,t)}{dx_t}$ is the Jacobian matrix of u_θ with respect to x_t.

Assumption 2 (Constancy of the Jacobian): The Jacobian $J_{u_\theta}(x_t, t)$ varies slowly with respect to x_t within these small neighborhoods. Therefore, for small δ, it can be approximated as constant:

$$J_{u_\theta}(x_t + \delta, t) \approx J_{u_\theta}(x_t, t). \tag{2}$$

Under these assumptions, we derive the following gradient relationship between $\nabla_{x_t}\mathcal{L}$ and $\nabla_{\hat{x}_0}\mathcal{L}$:

Lemma 2.2 (Gradient Relationship). *Let $u_\theta : \mathcal{R}^d \times [0, T] \to \mathcal{R}^d$ be the velocity function with the parameter θ. Then the gradient of the cost function ($\nabla_{x_t}\mathcal{L}$) at any timestep t can be approximated as:*

$$\nabla_{x_t}\mathcal{L} = (I + t \cdot J_{u_\theta})^T \nabla_{\hat{x}_0}\mathcal{L}. \tag{3}$$

Therefore, we get $\epsilon(t) = t \cdot J_{u_\theta}(x_t, t)^T \nabla_{\hat{x}_0}\mathcal{L}$. Importantly, when $t \to 0$, the matrices $I + t \cdot J_{u_\theta}(x_t, t)$ are close to the identity matrix. We further provide empirical evidence about this on pretrained rectified flow models by analyzing the gradients and convergence *w.r.t.* denoising steps in the Appendix H. Where we observe that gradient direction improves linearly and quickly converges to x_0^{ref} as $t \to 0$. Under this approximation, the difference between the two error dynamics becomes negligible. Since the $\epsilon(t)$ introduces only a small correction, it leads to the convergence in error dynamics as $t \to 0$. Combining the results of Preposition 2.1, Assumption 1 and 2, and Lemma 2.2, we obtain the following theorem with straightforward proof that facilitates the controlled generation for rectified flow models in the most computationally efficient way:

Theorem 2.3. *(Informal) Given the above assumption and notations, the update rule for the vector field driven by u_θ for the inference-time steering is:*

$$x_{t-\Delta t} = x_t + \Delta t \cdot u_\theta(x_t, t) - s' \nabla_{\hat{x}_0}\mathcal{L}, \tag{4}$$

where s' is the guidance scale.

The formal statement and proof are provided in the Appendix E. This theorem forms the core of **FlowChef**, enabling controlled generation efficiently.

3 Experiments

We evaluate the **FlowChef** across two tasks: (1) Linear inversion problems on pixel and latent-space models, (2) Image editing, and In Table 1, we report three inverse problems with easy and hard difficulty scenarios. It can be observed that **FlowChef** significantly improves the performance on both easy and hard settings across the tasks and all metrics consistently. While the concurrent

Method	BoxInpaint			Deblurring			Super Resolution		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Easy Scenarios									
Degraded	21.79	74.76	10.92	20.17	54.03	22.20	24.68	77.57	11.67
OT-ODE	19.11	77.86	13.49	21.86	62.51	15.14	21.64	62.23	26.64
PnP-Flow	22.12	68.02	14.70	22.00	65.79	15.95	22.42	68.06	14.91
D-FLow	20.37	70.06	13.67	20.22	61.99	14.51	21.60	69.89	12.29
FreeDoM	20.87	74.79	13.92	20.21	69.73	13.22	21.15	77.54	12.12
DPS	23.61	74.79	9.35	22.49	69.73	10.23	23.94	77.54	8.46
FlowChef (ours)	**26.32**	**87.70**	**3.36**	**27.69**	**86.43**	**2.66**	**26.00**	**80.15**	**4.43**
Hard Scenarios									
Degraded	18.75	65.12	22.54	16.83	30.02	54.04	20.77	55.85	38.16
OT-ODE	16.37	67.35	19.22	17.89	34.02	29.68	18.19	39.43	36.84
PnP-Flow	20.44	61.96	17.53	19.50	50.54	22.00	21.35	61.78	17.78
D-FLow	18.34	62.62	19.94	16.93	34.13	25.31	20.01	56.46	17.64
FreeDoM	18.88	65.07	16.83	16.50	34.88	18.91	19.58	55.84	14.12
DPS	20.68	65.06	13.06	17.58	34.89	15.86	21.52	55.90	10.31
FlowChef (ours)	**21.45**	**78.75**	**7.73**	**20.31**	**52.73**	**10.64**	**21.62**	**60.33**	**10.18**

Table 1: Pixel-space model-based evaluations for tackling the linear inverse problems.

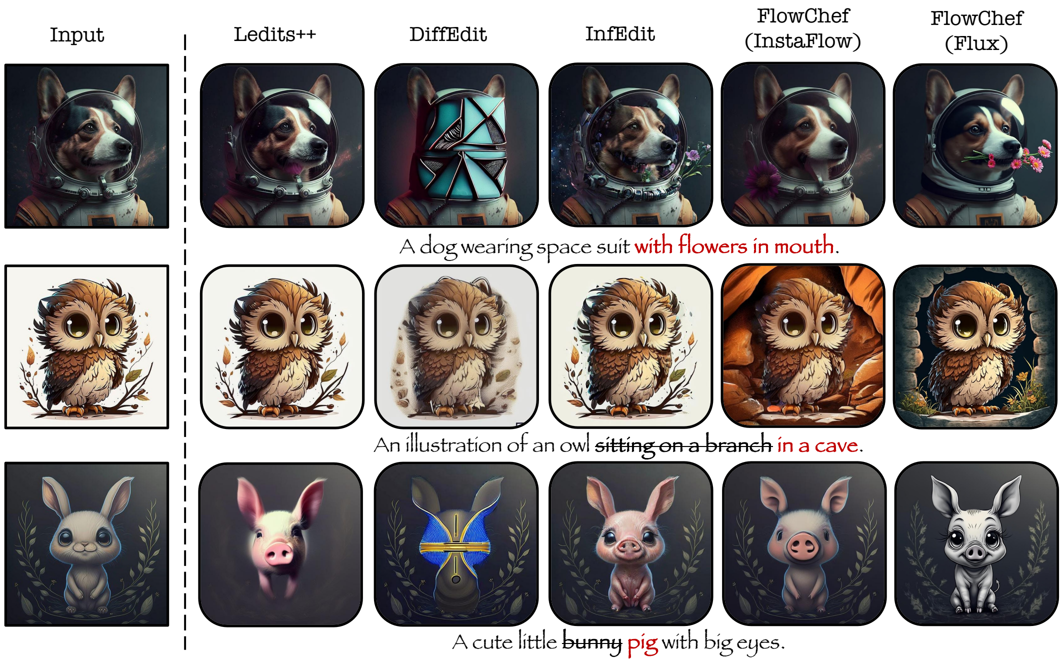

Input
Ledits++
DiffEdit
InfEdit
FlowChef (InstaFlow)
FlowChef (Flux)
A dog wearing space suit with flowers in mouth.
An illustration of an owl ~~sitting on a branch~~ in a cave.
A cute little ~~bunny~~ pig with big eyes.

Figure 2: **Qualitative results on image editing.** As illustrated, our method attains the SOTA performance on comparison inversion-free methods.

gradient-free work, PnP-Flow, outperforms many other baselines, **FlowChef** leads the benchmark. Figure 2 shows that **FlowChef** consistently outperforms the baselines without requiring inversion and our Flux variant further improves the performance. We further provide detailed experiments and it's setup in the appendix along with experiments on new tasks such as classifier guidance.

4 Conclusion

In this work, we introduced **FlowChef**, a versatile flow-based approach that unifies key tasks in controlled image generation, including linear inverse problems, image editing, and classifier-guided style transfer. Extensive experiments show that **FlowChef** outperforms baselines across all tasks, achieving state-of-the-art performance with reduced computational cost and memory usage. Notably, **FlowChef** enables inversion-free editing and scales to SOTA T2I models like Flux without memory issues. Our results demonstrate **FlowChef**'s adaptability and efficiency, offering a unified solution for both pixel and latent spaces across diverse architectures and practical constraints.

References

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=SE20BFqj6J`.

Manuel Brack, Felix Friedrich, Katharia Kornmeier, Linoy Tsaban, Patrick Schramowski, Kristian Kersting, and Apolinário Passos. Ledits++: Limitless image editing using text-to-image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8861–8870, 2024.

Andrew Brock. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

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*, 2022.

Hyungjin Chung, Jeongsol Kim, Sehui Kim, and Jong Chul Ye. Parallel diffusion models of operator and image for blind inverse problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6059–6069, 2023.

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=3lge0p5o-M-`.

Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems. *arXiv preprint arXiv:2410.00083*, 2024.

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua M. Susskind, and Navdeep Jaitly. Matryoshka diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=tOzCcDdH9O`.

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. In *The Twelfth International Conference on Learning Representations*, 2024.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=_CDixzkzeyb`.

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

Yi Huang, Jiancheng Huang, Yifan Liu, Mingfu Yan, Jiaxi Lv, Jianzhuang Liu, Wei Xiong, He Zhang, Shifeng Chen, and Liangliang Cao. Diffusion model-based image editing: A survey. *arXiv preprint arXiv:2402.17525*, 2024.

Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly ddpm noise space: Inversion and manipulations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12469–12478, 2024.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023.

Changhoon Kim, Kyle Min, Maitreya Patel, Sheng Cheng, and Yezhou Yang. Wouaf: Weight modulation for user attribution and fingerprinting in text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8974–8983, 2024a.

Changhoon Kim, Kyle Min, and Yezhou Yang. Race: Robust adversarial concept erasure for secure text-to-image diffusion model. *arXiv preprint arXiv:2405.16341*, 2024b.

Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. *arXiv preprint arXiv:2405.20320*, 2024.

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

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023a. URL `https://openreview.net/forum?id=XVjTT1nw5z`.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023b.

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

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Ségolène Martin, Anne Gagneux, Paul Hagemann, and Gabriele Steidl. Pnp-flow: Plug-and-play image restoration with flow matching. *arXiv preprint arXiv:2410.02423*, 2024.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=aBsCjcPu_tE`.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Maitreya Patel, Sangmin Jung, Chitta Baral, and Yezhou Yang. λ-eclipse: Multi-concept personalized text-to-image diffusion models by leveraging clip latent space. *ArXiv*, abs/2402.05195, 2024a. URL `https://api.semanticscholar.org/CorpusID:267547418`.

Maitreya Patel, Changhoon Kim, Sheng Cheng, Chitta Baral, and Yezhou Yang. Eclipse: A resource-efficient text-to-image prior for image generations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9069–9078, 2024b.

Xinyu Peng, Ziyang Zheng, Wenrui Dai, Nuoqian Xiao, Chenglin Li, Junni Zou, and Hongkai Xiong. Improving diffusion models for inverse problems using optimal posterior covariance. In *Forty-first International Conference on Machine Learning*, 2024.

Ashwini Pokle, Matthew J. Muckley, Ricky T. Q. Chen, and Brian Karrer. Training-free linear image inversion via flows, 2024. URL `https://openreview.net/forum?id=3JoQqW35GQ`.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=FjNys5c7VyY`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL `https://api.semanticscholar.org/CorpusID:231591445`.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

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, Yujia Chen, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Beyond first-order tweedie: Solving inverse problems using latent diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9472–9481, 2024a.

Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024b.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=nJfylDvgzlq`.

Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency. In *The Twelfth International Conference on Learning Representations*, 2024.

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

Zhicheng Sun, Zhenhao Yang, Yang Jin, Haozhe Chi, Kun Xu, Liwei Chen, Hao Jiang, Yang Song, Kun Gai, and Yadong Mu. Rectifid: Personalizing rectified flow with anchored classifier guidance. *arXiv preprint arXiv:2405.14677*, 2024.

Haofan Wang, Matteo Spinelli, Qixun Wang, Xu Bai, Zekui Qin, and Anthony Chen. Instantstyle: Free lunch towards style-preserving in text-to-image generation. *arXiv preprint arXiv:2404.02733*, 2024.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Zihui Wu, Yu Sun, Yifan Chen, Bingliang Zhang, Yisong Yue, and Katherine L Bouman. Principled probabilistic imaging using diffusion models as plug-and-play priors. *arXiv preprint arXiv:2405.18782*, 2024a.

Zongze Wu, Nicholas Kolkin, Jonathan Brandt, Richard Zhang, and Eli Shechtman. Turboedit: Instant text-based image editing. *arXiv preprint arXiv:2408.08332*, 2024b.

Sihan Xu, Yidong Huang, Jiayi Pan, Ziqiao Ma, and Joyce Chai. Inversion-free image editing with natural language. *arXiv preprint arXiv:2312.04965*, 2023.

Xiaofeng Yang, Cheng Chen, Xulei Yang, Fayao Liu, and Guosheng Lin. Text-to-image rectified flow as plug-and-play priors. *arXiv preprint arXiv:2406.03293*, 2024.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23174–23184, 2023.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

A Supplementary Overview

This supplementary material contains proofs, detailed results, discussion, and qualitative results:

- Section B: Preliminaries.
- Section C: Key Experiments.
- Section D: Proposition 2.1 proof.
- Section E: Theorem 2.3 proof.
- Section F: Numerical accuracy analysis.
- Section G: Extended related works.
- Section H: Empirical study of pixel and latent models.
- Section I: Detailed algorithms.
- Section J: Experimental setup details.
- Section K: Extended results.
- Section L: Hyperparameter study.
- Section M: Qualitative Results.
- Section N: Limitations & Future Work

B Preliminaries

Let $u_\theta : \mathcal{R}^d \times [0, T] \to \mathcal{R}^d$ represent a pretrained flow model estimating the drift $v = x_1 - x_0$ from x_t. The denoised sample $\hat{x}_0$ is obtained by integrating the drift u_θ over time from $t = T$ to $t = 0$, starting from $x_T \sim p_1$. With a target sample x_0^{ref}, we define a cost function $\mathcal{L} : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}_+$ that quantifies the cost of aligning $\hat{x}_0$ with x_0^{ref}, yielding the optimization problem:

$$\min_{\{\hat{x}_t\}_{t=0}^T} \quad \mathcal{L}(\hat{x}_t, x_0^{\text{ref}}), \tag{5}$$

where $\{\hat{x}_t\}_{t=0}^T$ represents the model-generated trajectory from x_T to x_0. The objective is to find the trajectory that minimizes $\mathcal{L}$, effectively steering the generated sample toward the target. This can be adapted for the denoising stage with either a noise-aware cost function at each timestep t or by estimating x_0 to refine the trajectory as needed. The gradient update is given by:

$$x_t \leftarrow x_t - s \cdot \nabla_{x_t} \mathcal{L}(\hat{x}_0, x_0^{ref}), \tag{6}$$

where s is guidance scale. This process requires estimating $\hat{x}_0$, backpropagating gradients through ODESolver (u_θ) to adjust x_t, and iteratively refining $x_{t-\Delta t}$. As it can be observed, this approach depends on accurate $\hat{x}_0$ estimation and substantial computation to ensure that the trajectory remains on the data manifold.

B.1 Cost Functions

Notably, explicit x_0^{ref} is unnecessary and can be approximated with appropriate cost functions depending on the downstream tasks. Assuming initial Gaussian noise x_T leads to $\hat{x}_0$, the cost function can be defined as:

$$\mathcal{L}(\hat{x}_0, x_0^{\text{ref}}) = ||\hat{x}_0 - x_0^{\text{ref}}||_2^2. \tag{7}$$

In inverse problems, let $\mathcal{F} : \mathcal{R}^d \to \mathcal{R}^n$ represent a degradation operation (e.g., downsampling for super-resolution). We then define:

$$\mathcal{L}(\hat{x}_0, x_0^{\text{ref}}) = ||\mathcal{F}(\hat{x}_0) - x_0^{\text{ref}}||_2^2. \tag{8}$$

Here, x_0^{ref} is a degraded sample, and we guide the model to generate $\hat{x}_0$ such that its degraded version matches x_0^{ref}. For classifier guidance, the cost function can be based on the negative log-likelihood (NLL). Specifically, given a classifier $p_\phi(c|\hat{x}_0)$, the cost function is:

$$\mathcal{L}(\hat{x}_0, c) = -\log p_\phi(c|\hat{x}_0). \tag{9}$$

Remark 1. Although presented in pixel space, this formulation extends to latent space by introducing a Variational Autoencoder (VAE) encoder ($\mathcal{E}$) and decoder ($\mathcal{D}$).

C Key Experiments

In this section, we detail the remaining experiments in details.

C.1 Linear Inversion Problems

We evaluate **`FlowChef`** against several baselines on three common linear tasks: box inpainting, super-resolution, and Gaussian deblurring, under varying difficulty levels. We extend both **`FlowChef`** and the baselines to latent-space models to simulate real-world applications, reporting results on PSNR, SSIM Wang et al. (2004), and LPIPS Zhang et al. (2018) across 200 images from CelebA Liu et al. (2015) and AFHQ-Cat Choi et al. (2020).

C.1.1 Pixel-space models

As **`FlowChef`** requires straightness and no crossovers, we select the Rectified-Flow++ pretrained models Lee et al. (2024). We compare **`FlowChef`** with recent flow-based methods OT-ODE Pokle et al. (2024), D-Flow Ben-Hamu et al. (2024), and PnP-Flow (concurrent work) Martin et al. (2024), implementing the former two baselines manually due to lack of open-source access and tuning them for optimal performance. Additionally, we extend two diffusion-based baselines, DPS Chung et al. (2022) and FreeDoM Yu et al. (2023), for the RFMs. For comparisons, we use the Rectified-Flow++ models that are pretrained on FFHQ (for CelebA) and AFHQ-Cat datasets. Experiments are conducted for 64x64 image resolutions. Hyper-parameters for each method are reported in the Appendix L. Our selected tasks include: (1) Box inpainting with 20x20 and 30x30 centered masks, (2) Super-resolution with 2x and 4x scaling factors, and (3) Gaussian deblurring with an 11x11 kernel at intensities of 1.0 and 10.0, with added Gaussian noise at $\sigma = 0.05$ for robustness.

Results. We present the quantitative and qualitative evaluation results in Table 1 and Appendix M, respectively. It can be observed that **`FlowChef`** significantly improves the performance on both easy and hard settings across the tasks and all metrics consistently. Notably from Table 7, we find that the **`FlowChef`** is also the fastest and most memory efficient. Surprisingly, diffusion-based extended baseline (DPS) significantly outperforms even recent baselines. However, DPS requires backpropagation through billions of parameters of ODESolver. While the concurrent gradient-free work, PnP-Flow, outperforms many other baselines, **`FlowChef`** leads the benchmark.

C.1.2 Latent-space models.

Flow-based baselines are not extended to the latent space models as either they are already very computationally heavy or require extra Jacobian calculations to support the non-linearity introduced by the VAE models. We adapt D-Flow Ben-Hamu et al. (2024) and RectifID Sun et al. (2024) as flow-based baselines, adding diffusion-based baselines PSLD-LDM Rout et al. (2024b) and Resample Song et al. (2024) for comparison. We use InstaFlow Liu et al. (2023b) (Stable Diffusion v1.5 variant) and Flux models as a baseline for flow-based approaches and utilize the original Stable Diffusion v1.5 checkpoint for the diffusion-based baselines. We perform all tasks in 512 x 512 resolution, increasing to 1024 x 1024 for Flux experiments. Our task settings are: (1) Box inpainting with a 128x128 mask, (2) Super-resolution at 4x scaling, and (3) Gaussian deblurring with a 50x50 kernel at intensity 5.0, all without extra Gaussian noise. For consistency, settings are doubled for Flux to a 256x256 mask, 8x super-resolution scaling, and 10.0 deblurring intensity. As VAE encoders add extra unwanted nonlinearity, pixel-level cost functions alone may not be optimal. Hence,

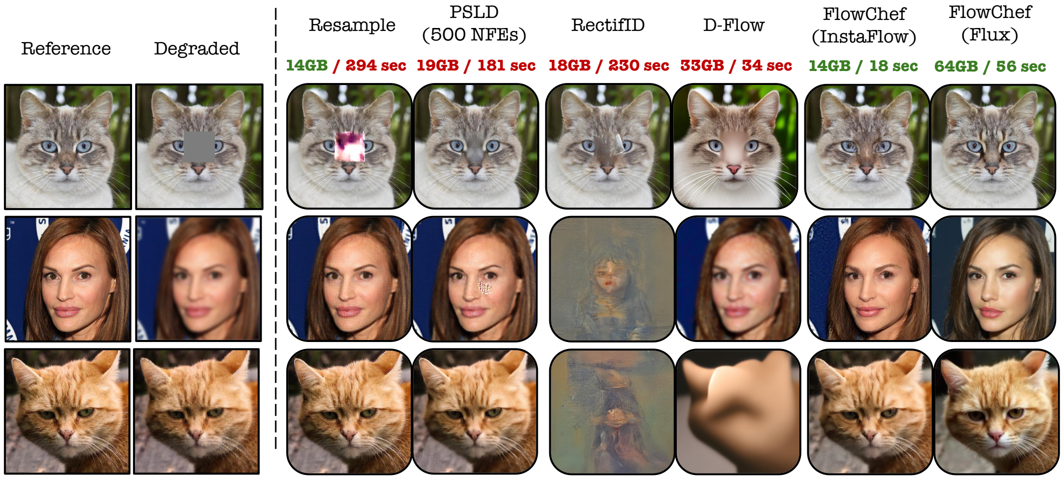


Figure 3: **Qualitative results on linear inverse problems.** All baselines are implemented on stable diffusion v1.5, except **`FlowChef`** Flux variant. Results are reported for VRAM and time on an A100 GPU at 512 x 512 resolution, with Flux experiments at 1024 x 1024. Best viewed when zoomed in.

Method	**BoxInpaint**			**Super Resolution**			**Deblurring**		
	PSNR (↑)	**SSIM (↑)**	**LPIPS (↓)**	**PSNR (↑)**	**SSIM (↑)**	**LPIPS (↓)**	**PSNR (↑)**	**SSIM (↑)**	**LPIPS (↓)**
Diffusion based methods									
Resample	20.12	79.94	19.36	26.91	70.91	30.75	25.27	62.97	41.94
PSLD (500 NFEs)	28.30	93.81	4.49	25.79	65.15	33.27	26.64	65.44	43.10
PSLD (100 NFEs)	26.90	93.13	5.29	21.95	54.67	46.08	21.25	51.62	51.92
Flow based methods									
D-Flow	19.68	65.01	27.79	20.23	60.55	50.30	22.42	64.43	53.04
RectifID	23.81	75.13	10.50	10.36	31.55	67.08	10.40	31.16	66.60
`FlowChef` (InstaFlow)	22.94	73.55	**9.94**	**25.83**	**64.73**	**31.38**	**22.50**	47.42	**42.54**
`FlowChef` (Flux)	**25.74**	**82.99**	**9.40**	**20.25**	**64.34**	**41.88**	18.98	**64.37**	53.43

Table 2: Latent-space model based evaluations for tackling the linear inverse problems. SSIM & LPIPS results are multiplied by 100.

we calculate the loss in the latent space only for the box inpainting task (as the degradation function is known with $\sigma = 0$), allowing us to extend to image editing later. For super-resolution and deblurring, we stick with the pixel-level cost functions. We further detail the task-specific settings and hyperparameters in the Tables 5 & 4.

Results. Quantitative and qualitative results in Figure 3 and Table 2 show that **`FlowChef`** achieves SOTA performance for flow-based methods. However, a huge gap still remains *w.r.t.* the diffusion-based methods like Resample and PSLD. Notably, these baselines take about 5 minutes and 3 minutes, respectively, per image (see Figure 3), while **`FlowChef`** only takes only 18 seconds and less memory (only 14GB). None of the existing flow-based methods can be extended to Flux due to memory constraints. But **`FlowChef`** can seamlessly be applied, which further improves the performance. We find that **`FlowChef`** (Flux) reduces the artifacts in the images completely but observes the slight degradation in color dynamics. We attribute this to the some nonlinearity observed in the trajectory of Flux.

C.2 Image Editing (Human Evals)

We perform extensive human evaluations (A/B testing) across the five models and 100 randomly selected prompts from PIE-Bench. We utilized MTurk for this study and took three assessments per task – resulting in total of 1500 assessments. As noted in Figure 5, we can observe that **`FlowChef`** (InstaFlow) model outperforms the other inversion-free methods DiffEdit and InfEdit, while competing with the inversion-based approach Ledits++. Additionally, **`FlowChef`** (Flux.1 [Dev]) variant outperforms the InstaFlow variant.

D Proof of the Proposition

Proposition 4.1. Let $p_1 \sim \mathcal{N}(0, \mathcal{I})$ be the noise distribution and p_0 be the data distribution. Let x_t denote an intermediate sample obtained from a predefined forward function q as $x_t = q(x_0, x_1, t)$, where $x_0 \sim p_0$ and $x_1 \sim p_1$. Define a ODE sampling process $dx(t) = f(x_t, t)dt$ and quadratic $\mathcal{L} = ||\hat{x}_0 - x_0^{ref}||_2^2$, where $f : \mathcal{R}^d \times [0, T] \to \mathcal{R}^d$ is a nonlinear function parameterized by θ. Then, under Assumption 1, the error dynamics of ODEs for controlled image generation are governed by:

$$\frac{dE(t)}{dt} = -4sE(t) + 2e(t)^T \epsilon(t),$$

where $e(t)$ is $\hat{x}_0 - x_0^{ref}$, $E(t) = e(t)^T e(t)$ is the squared error magnitude, $s > 0$ is the guidance strength, and $\epsilon(t)$ represents the accumulated errors due to non-linearity and trajectory crossovers.

Proof. Consider the sampling process described by the ODE:

$$\frac{dx(t)}{dt} = f(x(t), t), \tag{10}$$

where $f(x(t), t)$ is a nonlinear function often parameterized via neural network θ. To guide the sampling process toward minimizing a loss function $\mathcal{L}(\hat{x}_0, x_0^{ref})$, we can adjust the dynamics by adding the gradient ∇_{x_t} to the vector field (see Eq. 6) as:

$$\frac{dx(t)}{dt} = f(x(t), t) - s \cdot \nabla_{x_t} \mathcal{L}(\hat{x}_0, x_0^{\text{ref}}), \tag{11}$$

where s is the guidance strength. Let $e(t) = \hat{x}_0 - x_0^{ref}$ be the error between the estimated and target samples. Since $\hat{x}_0(t) = x(t) + \int_t^0 f(x(\tau), \tau)d\tau$, differentiating $e(t)$ with respect to t yields:

$$\begin{aligned} \frac{de(t)}{dt} &= \frac{d\hat{x}_0(t)}{dt} && (12)\\ &= \frac{dx(t)}{dt} - f(x(t), t) && (13)\\ &= -s \cdot \nabla_{x_t} \mathcal{L}(\hat{x}_0, x_0^{\text{ref}}). && (14) \end{aligned}$$

However, this requires the compute-intensive backpropagation through ODESolver. Therefore, it is important to find an approximation of ∇_{x_t}. And the most convenient approximation is: $\nabla_{x_t} \approx \nabla_{\hat{x}_0}$. However, this derivation assumes that the integral $\int_t^0 f(x(\tau), \tau)d\tau$ is well-behaved and that $\hat{x}_0(t)$ depends smoothly on $x(t)$. In the presence of nonlinearity and trajectory crossovers, small changes in $x(t)$ can lead to disproportionately large changes in $\hat{x}_0(t)$, due to the sensitivity of the integral to the path taken. Moreover, potential crossovers in the trajectory mean that the mapping from $x(t)$ to $\hat{x}_0(t)$ is not injective; different trajectories $x(t)$ may lead to the same $\hat{x}_0(t)$ or vice versa. This non-unique mapping complicates the error dynamics because $\nabla_{\hat{x}_0}\mathcal{L}$ may not provide a consistent or effective direction for updating $x(t)$. Including the effects of nonlinearity and trajectory crossovers, the error dynamics become:

$$\frac{de(t)}{dt} = -s \cdot \nabla_{\hat{x}_0} \mathcal{L}(\hat{x}_0, x_0^{\text{ref}}) + \epsilon(t), \tag{15}$$

where $\epsilon(t)$ represents the errors introduced by the nonlinearity in $f(x(t), t)$ and the sensitivity of $\hat{x}_0$ to $x(t)$ due to trajectory crossovers. In other words, the approximation error $\epsilon(t)$ can be represented as:

$$\epsilon(t) = s \cdot \left(\nabla_{x_t} \mathcal{L}(\hat{x}_0, x_0^{ref}) - \nabla_{\hat{x}_t} \mathcal{L}(\hat{x}_0, x_0^{ref}) \right). \tag{16}$$

Assuming a quadratic loss function $\mathcal{L} = ||\hat{x}_0 - x_0^{ref}||_2^2$, we have $\nabla_{\hat{x}_0}\mathcal{L} = 2e(t)$, leading to:

$$\frac{de(t)}{dt} = -2se(t) + \epsilon(t). \tag{17}$$

To understand the convergence of the error, we analyze the evolution of the error magnitude $E(t) = e(t)^T e(t)$. Differentiating $E(t)$ with respect to time t, we get:

$$\begin{aligned}
\frac{dE(t)}{dt} &= \frac{d}{dt}\left(e(t)^\top e(t)\right) && (18)\\
&= 2e(t)^\top \frac{de(t)}{dt} && (19)\\
&= 2e(t)^\top\left(-2se(t) + \epsilon(t)\right) && (20)\\
&= -4se(t)^\top e(t) + 2e(t)^\top \epsilon(t) && (21)\\
&= -4sE(t) + 2e(t)^\top \epsilon(t). && (22)
\end{aligned}$$

This completes the proof. □

Notably, we derive this behavior of the ODE processes under the assumption that the error rate cannot be calculated accurately. This can either come from the incorrect estimation of $\hat{x}_0$ or the nonlinearity of ODESolver itself. In the next section, we further concretize this with respect to the RFMs.

E PROOF FOR THEOREM

Lemma 4.2 (Gradient Relationship)**.** Let $u_\theta : \mathcal{R}^d \times [0, T] \to \mathcal{R}^d$ be the velocity function with the parameter θ. Then the gradient of the cost function $\mathcal{L}$ at any timestep t can be approximated as:

$$\nabla_{x_t}\mathcal{L} = (I + t \cdot J_{u_\theta})^T \nabla_{\hat{x}_0}\mathcal{L}. \tag{23}$$

Proof. Leveraging the straight-line trajectories characteristic of rectified flow models, the data sample at $t = 0$ can be estimated directly from an intermediate state x_t:

$$\hat{x}_0 = x_t + t \cdot u_\theta(x_t, t). \tag{24}$$

By differentiating the $\hat{x}_0$ with respect to x_t, we get:

$$\begin{aligned}
\frac{d\hat{x}_0}{dx_t} &= I + t \cdot \frac{du_\theta(x_t, t)}{dx_t} && (25)\\
&= I + t \cdot J_{u_\theta}(x_t, t). && (26)
\end{aligned}$$

Using the chain rule for gradients:

$$\nabla_{x_t}\mathcal{L} = \left(\frac{d\hat{x}_0}{dx_t}\right)^T \nabla_{\hat{x}_0}\mathcal{L}. \tag{27}$$

Substituting the expression for $\frac{d\hat{x}_0}{dx_t}$, we obtain:

$$\nabla_{x_t}\mathcal{L} = (I + t \cdot J_{u_\theta}(x_t, t))^T \nabla_{\hat{x}_0}\mathcal{L}. \tag{28}$$

According to Assumption 3, due to the constancy of Jacobian, J_{u_θ}, for rectified flow models, we can treat it as constant for any time t. Hence, we get our desired approximation:

$$\nabla_{x_t}\mathcal{L} = (I + t \cdot J_{u_\theta}(x_t, t))^T \nabla_{\hat{x}_0}\mathcal{L}. \tag{29}$$

This completes the proof.

□

Hence, as either $t \to 0$ or J_{u_θ} (Assumption 2), both gradients become approximately similar and $\epsilon(t) \to 0$. This guarantees the convergence of the error dynamics as time passes. We further show this behavior of RFMs empirically in Section H and show that this remains true for even large-scale latent models.

Theorem 4.3 (Update Rule for Steering the RFMs)**.** Let $u_\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ be a velocity field with constant Jacobian J_{u_θ}. Define the estimated initial state $\hat{x}_0$ from an intermediate state x_t by

$$\hat{x}_0 = x_t + t \cdot u_\theta(x_t, t).$$

Consider the quadratic loss function $\mathcal{L} = \|\hat{x}_0 - x_0^{\text{ref}}\|^2$, where x_0^{ref} is a reference sample. Then, the update rule for controlled generation is given by

$$x_{t-\Delta t} = x_t + \Delta t u_\theta(x_t, t) - s' \nabla_{\hat{x}_0}\mathcal{L},$$

where:

- $\nabla_{\hat{x}_0}\mathcal{L} = 2(\hat{x}_0 - x_0^{\text{ref}})$,
- $s' \approx (I + \Delta t \cdot J_{u_\theta})\,(I + t \cdot J_{u_\theta})^\top$,
- I is the identity matrix.

Proof. By lemma 2.2 and Assumption 2, we can further approximate the Eq. 23:

$$\nabla_{x_t}\mathcal{L} = (I + t \cdot J_{u_\theta})^T \nabla_{\hat{x}_0}\mathcal{L} \approx K^T \nabla_{\hat{x}_0}\mathcal{L}, \tag{30}$$

where K is the constant matrics as $\Delta t \to 0$ and $t \to 0$. Under this formulation, we can perform controlled image generation in three steps:

$$\begin{aligned} &\text{Step 1:} & \hat{x}_0 &= x_t + t \cdot u_\theta(x_t, t) \\ &\text{Step 2:} & \hat{x}_t &= x_t - K^T \nabla_{\hat{x}_0}\mathcal{L} \\ &\text{Step 3:} & x_{t-\Delta t} &= \hat{x}_t + \Delta t \cdot u_\theta(\hat{x}_t, t). \end{aligned} \tag{31}$$

However, this will require additional forward passes. But according to Assumption 2 if Δt is sufficiently small, then by Taylor series approximation, we get:

$$\begin{aligned} x_{t-\Delta t} &= x_t - K^T \nabla_{\hat{x}_0}\mathcal{L} + \Delta t \cdot u_\theta\left(x_t - K^T \nabla_{\hat{x}_0}\mathcal{L}, t\right) && (32) \\ &= x_t - K^T \nabla_{\hat{x}_0}\mathcal{L} \\ &\quad + \Delta t \left[u_\theta(x_t, t) - J_{u_\theta} \cdot K^T \cdot \nabla_{\hat{x}_0}\mathcal{L}\right] && (33) \end{aligned}$$

Now, as J_{u_θ} is constant *w.r.t.* Δt. Hence, we get:

$$\begin{aligned} x_{t-\Delta t} &= x_t - (I + \Delta t \cdot J_{u_\theta}) K^T \nabla_{\hat{x}_0}\mathcal{L} + \Delta t \cdot u_\theta(x_t, t) && (34) \\ &= x_t + \Delta t \cdot u_\theta(x_t, t) - s' \nabla_{\hat{x}_0}\mathcal{L}, && (35) \end{aligned}$$

where $s' = (I + \Delta t \cdot J_{u_\theta}) K^T$ is constant and it can predetermined.

Hence, this concludes the proof that for appropriate guidance scale s', we can perform the controlled generation as derived above. □

F NUMERICAL ACCURACY FOR MODEL STEERING

In our controlled generation framework, we aim to steer the generation process towards a reference sample x_0^{ref} by solving the modified ODE:

$$\frac{dx(t)}{dt} = f(x(t), t) = u_\theta(x(t), t) - s' \nabla_{\hat{x}_0} \mathcal{L}. \tag{36}$$

The accuracy of this numerical integration is crucial, as errors can accumulate over time, leading to deviations from the desired trajectory. The smoothness of the modified velocity field $f(x(t), t)$ significantly impacts this accuracy. Specifically, a smaller magnitude of $\left|\frac{d}{dt} f(x(t), t)\right|$ reduces local truncation errors. The following Proposition formalizes this relationship, stating that the numerical accuracy improves as $\left|\frac{d}{dt} f(x(t), t)\right|$ decreases.

Proposition F.1. *(Informal). Given the prior notations, Assumptions, and Theorem, for any p-th order numerical method solving Eq. equation 36, the accuracy of the numerical solution increases as the magnitude of $\left|\frac{d}{dt} f(x(t), t)\right|$ decreases.*

Proof. To analyze the local truncation error, consider the Taylor series expansion of the exact solution around time t when integrating backward in time from t to $t - \Delta t$:

$$\begin{aligned} x(t - \Delta t) =& x(t) - \Delta t\, f(x(t), t) + \frac{(\Delta t)^2}{2} \frac{d}{dt} f(x(t), t) \\ & - \frac{(\Delta t)^3}{6} \frac{d^2}{dt^2} f(x(t), t) + O\left((\Delta t)^4\right). \end{aligned}$$

The numerical method updates the solution using:

$$x_{t-\Delta t} = x_t + \Delta t\, \phi(x_t, t), \tag{37}$$

where $\phi(x_t, t)$ is the increment function. The local truncation error τ is the difference between the exact solution and the numerical approximation:

$$\begin{aligned} \tau &= x(t - \Delta t) - x_{t-\Delta t} \\ &= \Big[x(t) - \Delta t\, f(x(t), t) + \frac{(\Delta t)^2}{2} \frac{d}{dt} f(x(t), t) \\ &\quad - \frac{(\Delta t)^3}{6} \frac{d^2}{dt^2} f(x(t), t) + O\left((\Delta t)^4\right) \Big] \\ &\quad - \left[x_t + \Delta t\, \phi(x_t, t) \right]. \end{aligned}$$

The first p-order terms cancel out, and we have:

$$||\tau|| \leq \left\| \frac{(\Delta t)^{p+1}}{(p+2)!} \frac{d^{p+1}}{dt^{p+1}} f(x(t), t) \right\| \tag{38}$$

According to the Mean Value Theorem, we have

$$||\tau|| \leq C (\Delta t)^{p+1} \max_{t \in [t_n, t_{n+1}]} \left\| \frac{d}{dt} f(x(t), t) \right\| \tag{39}$$

where C is a constant depending on the method. The global error $e(t) = x(t) - x_t$ accumulates these local errors over the integration interval. Under standard assumptions (e.g., Lipschitz continuity of f), the global error is bounded by:

$$\|e(t)\| \leq K(\Delta t)^p \left(e^{L(T-t)} - 1\right) \max_{t\in[0,T]} \left\| \frac{d}{dt} f(x(t), t) \right\|, \tag{40}$$

where K is a constant depending on the Lipschitz constant L of f and the total integration time T. $\square$

As the magnitude of $\left\| \frac{d}{dt} f(x_t, t) \right\|$ decreases, both the local truncation error and the global error decrease, enhancing the accuracy of the numerical solution. In the context of controlled generation, ensuring that $f(x_t, t)$ changes smoothly over time leads to more accurate integration and better alignment with the reference point x_0^{ref}. This insight and prior assumptions require that the guidance scale s' and δt be sufficiently smaller, where higher NFEs lead to the lower Δt. Hence, we increase the NFEs significantly to stabilize the steering (see Section L). By carefully selecting s', we ensure that the additional term $s' \cdot \nabla_{\hat{x}_0} \mathcal{L}$ does not introduce excessive variability into $f(x(t), t)$, maintaining the smoothness necessary for accurate numerical integration.

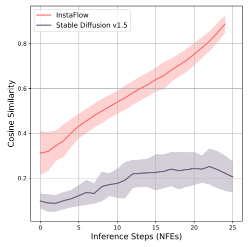
G Extended Related Works

Generative Models. Recent advances in generative models, especially diffusion models like Latent Diffusion Model (LDM) Rombach et al. (2022), GLIDE Nichol et al. (2021), and DALL-E2 Ramesh et al. (2022), have significantly improved photorealism compared to GAN-based methods such as StackGAN Zhang et al. (2017) and BigGAN Brock (2018). Pretrained diffusion models have been successfully applied to diverse tasks, including image editing Hertz et al. (2023), personalization Patel et al. (2024a), and style transfer Wang et al. (2024), but their inference flexibility remains limited, and they demand substantial resources Gu et al. (2024); Patel et al. (2024b). Distillation-based strategies like Latent Consistency Models Luo et al. (2023) and Distribution Matching Distillation Yin et al. (2024) address some limitations but lack control and broader applicability. Rectified Flow Models (RFMs) Liu et al. (2023a); Lipman et al. (2023), exemplified by Flux[1], SD3 Esser et al. (2024), and InstaFlow Liu et al. (2023b), show promise but face challenges in downstream tasks due to inversion inaccuracies and other limitations. This work addresses these gaps, extending RFMs to downstream tasks in a training-, gradient-, and inversion-free manner.

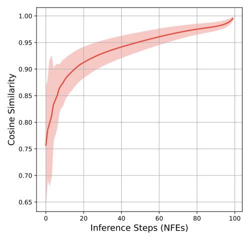
Conditional Sampling. Song *et al.* introduced noise-aware classifiers for controlling sampling in diffusion models Dhariwal & Nichol (2021), but these require task-specific training. Classifier-free guidance (CFG) Ho & Salimans (2022) avoids this but necessitates an additional pretraining stage. FreeDoM Yu et al. (2023) and MPGD He et al. (2024) improve sampling control but remain computationally intensive. Initial extensions of conditional sampling to flow models face similar challenges, such as compute-heavy gradient backpropagation and limited applicability to latent space models. Our method, **`FlowChef`**, eliminates these issues, seamlessly enabling gradient- and inversion-free conditional sampling in latent-space models.

Inverse Problems. This task addresses training-free approaches for solving inverse problems such as in-painting, super resolution, Gaussian de-blurring etc Daras et al. (2024). Inverse problems, dominated by diffusion-based methods Daras et al. (2024), include pixel-space solutions such as DPS Chung et al. (2022), Π-GDM Peng et al. (2024), and BlindDPS Chung et al. (2023). PSLD Rout et al. (2024b) extends support to latent-space models, while manifold-based methods Song et al. (2024); He et al. (2024) further enhance performance. Since Dhariwal et. al. demonstrated that guiding models with classifiers improves image generation quality Dhariwal & Nichol (2021), much of the current literature focuses on diffusion models, particularly pixel-space models Daras et al. (2024); Song et al. (2023); Chung et al. (2022); Wu et al. (2024a). However, these models face challenges when scaled to latent-space models, as they are incompatible with off-the-shelf pretrained models and require backpropagation through ODESolvers, which can take at least three minutes per image for satisfactory results Daras et al. (2024); Rout et al. (2024b); Song et al. (2024); Rout et al. (2024a). Methods such as MPGD He et al. (2024) attempt to mitigate these issues via manifold correction, but limitations persist, especially with large-scale models. Recent work has extended these approaches to ODEs (e.g., OT-ODE) and flow models Pokle et al. (2024). D-Flow Ben-Hamu et al.

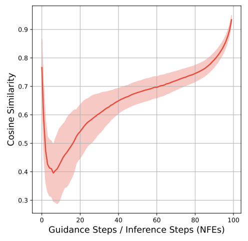
[1]`https://huggingface.co/black-forest-labs/FLUX.1-dev`



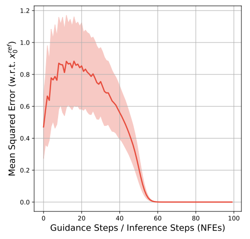
(a) Gradient Similarity in InstaFlow *vs.* Stable Diffusion v1.5.



(b) Gradient Similarity in Rectified Flow ++ model.



(c) Gradient Similarity in Rectified Flow ++ during model steering.



(d) Convergence in Rectified Flow ++ during model steering.

Figure 4: Empirical analysis of gradient similarity (a, b, and c) and convergence rate. (a) and (b) analyzes the gradients without model steering. (c) contains the gradient similarity during the active model steering. And (d) shows the trajectory similarity at each timestep t *w.r.t.* the inversion based trajectory.

Method	**NFEs**	**CG Scale**	**FID (↓)**	**VRAM (↓)**	**Time (↓)**
DDIM	50	-	5.39	3.67	14.22
MPGD	50	1	4.24	6.56	25.01
MPGD	50	10	5.46	6.56	25.01
Ours$_{\text{w/ skip grad}}$	50	1	**19.28**	6.56	24.95
RFPP (2-flow)	2	-	4.56	3.29	0.28
RFPP (2-flow)	15	-	4.29	3.36	2.75
Ours$_{\text{w/ backpropagation}}$	15	5	**2.77**	17.98	12.79
Ours$_{\text{w/ skip grad}}$	15	50	3.13	**6.64**	**5.85**

Table 3: Performance of Various guided sampling methods on ImageNet64x64 with 32 batch size inference on A6000 GPU.

(2024), for instance, optimizes initial noise by differentiating through the full trajectory chain; however, this comes with significant resource demands and is not adaptable to state-of-the-art (SOTA) models like Flux or SD3 Esser et al. (2024). In this work, we propose **`FlowChef`**, which addresses linear inverse problems in a gradient- and inversion-free manner.

Image Editing. Diffusion-based approaches dominate image editing Huang et al. (2024), but they rely heavily on accurate inversion Ju et al. (2023); Brack et al. (2024); Mokady et al. (2023); Huberman-Spiegelglas et al. (2024). Although inversion-free diffusion methods are faster, they often lack in edit quality Xu et al. (2023); Couairon et al. (2023); Meng et al. (2022); Wu et al. (2024b). Despite RFMs being SOTA in text-to-image (T2I) generation, they still lack robust editing capabilities. iRDS Yang et al. (2024) presents an inversion strategy for RFMs, especially InstaFlow Liu et al. (2023b), but it lacks quality and control. Similarly, RectifID Sun et al. (2024) offers an optimization-based approach to modify the whole trajectory for personalized T2I generation but performs poorly with InstaFlow like straight models. To the best of our knowledge, we present the first comprehensive solution that enhances RFMs for image editing and extends beyond it that too without significant computational or time overhead.

H Empirical Findings

In Section 2, we provided theoretical insights into **`FlowChef`** along with an intuitive algorithm. To complement the theory, we conducted an empirical analysis on large-scale RFMs to validate the Assumptions, Propositions, Lemmas, and Theorems presented. The results are summarized in Figure 4.

In Figure 4a, we compare the gradient cosine similarity with and without backpropagation through the ODESolver for InstaFlow and Stable Diffusion v1.5. For all denoising steps, the gradients of

Algorithm 1: **`FlowChef`** *vs.* Baseline FreeDoM.

```
1  Input: Pretrained Rectified-flow model $u_\theta$, input noise sample $x_T \sim N(0, I)$, target data
     sample $x_0^{ref}$, and $\mathcal{L}$ cost function.
2  for $t \in \{T...0\}$ do
3      $v \leftarrow u_\theta(x_t, t)$   $dt \leftarrow 1/T$
4      $x_t \leftarrow x_t.require\_grad\_(True)$
5      for $N$ steps do
6          $v \leftarrow u_\theta(x_t, t)$
7          $\hat{x}_0 \leftarrow x_t + t \cdot v$
8          $loss \leftarrow \mathcal{L}(\hat{x}_0, x_0^{ref})$
9          $\nabla_{x_t} \leftarrow$ grad(loss, $x_t$)   // Compute heavy BP
10         $x_t \leftarrow$ Optimize($x_t$, loss)   // Lemma 2.2
11     $v \leftarrow u_\theta(x_t, t)$
12     $x_{t-1} \leftarrow x_t + dt \cdot v$   // Theorem 2.3
13 RETURN $x_0$
```

SDv1.5 behave nearly randomly, indicating that the stochasticity of the base model significantly impacts gradients, even when using the ODE sampling process during inference. In contrast, for InstaFlow, as denoising progresses ($t \to 0$), gradient alignment improves, supporting our derivation in Lemma 2.2, which states that as $t \to 0$, we achieve $\nabla_{x_t} \approx \nabla_{\hat{x}_0}$.

Further analysis was performed on the Rectified Flow++ model, which is designed for straight trajectories with zero crossovers. As shown in Figure 4b, well-trained models exhibit high gradient similarity even at the initial stages of denoising. However, as illustrated in Figure 4c, during active steering, the gradient direction initially diverges before improving. This behavior is also reflected in the convergence plot in Figure 4d.

We hypothesize that this phenomenon arises due to the proximity to the Gaussian noise space ($p_1 \sim N(0, I)$), where model steering is more error-prone since minor adjustments can disproportionately affect future trajectories. As denoising progresses and the distribution moves further from the noise (p_1), these errors diminish, and convergence is achieved. These observations align well with our theoretical predictions, further reinforcing the validity of **`FlowChef`**.

I ALGORITHMS

This section provides an overview of the algorithms underpinning **`FlowChef`** for image editing and its comparison to baseline methods for a comprehensive understanding.

Image Editing. As described in Section 2.2, **`FlowChef`** can be easily extended to image editing. Revisiting the core concept, **`FlowChef`** modifies random trajectories to align with a target sample. Image editing involves balancing similarity with the target sample while introducing deviations to achieve desired edits.

Figure 1 and Section L illustrate how **`FlowChef`** progressively transfers characteristics from high-level structure to finer details like color composition. However, editing requirements vary by task. For example, adding an object benefits from trajectory adjustments earlier in the denoising process, while color changes require gradual learning at later stages. We can optimize parameters for diverse tasks using the generalized **`FlowChef`**, as detailed in Algorithm **??**.

To simplify the process, we extend **`FlowChef`** to support off-the-shelf editing tasks, such as those in the PIE-Benchmark, as detailed in Algorithm 1. Assume a non-edit region mask, M_{edit}, derived from cross-attention or human annotation. To steer the trajectory towards the desired edits, we modify the velocity (v) using a classifier-free guidance strategy:

Algorithm 2: : **`FlowChef`** optimized for a wide range of image editing tasks.

```
1  Input: Pretrained Rectified-flow model $u_\theta$, input noise sample $x_T \sim N(0, I)$, target data
     sample $x_0^{ref}$, $c^{edit}$ is edit prompt, $c^{base}$ is base prompt, $M$ is user-provided input mask, and $\mathcal{L}$
     cost function.
2  for $t \in \{T...0\}$ do
3      $dt \leftarrow 1/T$
4      $c \leftarrow [c^{edit}, c^{base}]$
5      $v \leftarrow u_\theta(x_t, t, c)$
6      $v_{edit}, v_{base} = v.chunk(2)$
7      $v = v_{edit} + \neg mask \cdot (v_{edit} - v_{base}) \cdot s$
8      $M_{edit} \leftarrow M$
9      $x_t \leftarrow x_t.require\_grad\_(true)$
10     if $t < min_T$ then
11         for $N$ steps do
12             $\hat{x}_0 \leftarrow x_t + t \cdot v$
13             if $t < max\_full\_steps_T$ then
14                 $M_{edit} \leftarrow I$
15             $loss \leftarrow \mathcal{L}(\hat{x}_0, x_0^{ref}) \cdot M_{edit}$
16             $x_t \leftarrow$ Optimize($x_t$, loss)    // Lemma 2.2
17     $x_{t-1} \leftarrow x_t + dt \cdot v$                // Theorem 2.3
18 RETURN $x_0$
```

$$v = v_{edit} + \neg mask \cdot (v_{edit} - v_{base}) \cdot s, \tag{41}$$

where v_{edit} corresponds to the edit prompt and v_{base} to the base (negative) prompt. This adjustment ensures the trajectory reflects the desired edits.

To maintain alignment of non-edited regions with the target sample, we modify the cost function as follows:

$$\mathcal{L}(\hat{x}_0, x_0^{ref}) = ||(\hat{x}_0 - x_0^{ref}) \cdot M_{edit}||_2^2. \tag{42}$$

Preserving the original image structure is crucial for edits such as color or material changes. To achieve this, we introduce the parameter $max\_full\_steps\_T$, which determines the number of steps that apply full **`FlowChef`** guidance with an identity mask. This ensures structural preservation while facilitating edits. Section L details a comprehensive reference for hyperparameters.

`FlowChef` vs. Baseline FreeDoM. Algorithm 1 compares **`FlowChef`** to the baseline FreeDoM, a diffusion model method that modifies the score function using a classifier guidance-like approach. FreeDoM requires estimating velocity and calculating gradients (∇_{x_t}) through backpropagation via the ODESolver u_θ, as marked in red. In contrast, as highlighted in green, **`FlowChef`** eliminates the need for backpropagation while still achieving convergence. This simplification makes **`FlowChef`** a more efficient and practical solution without sacrificing performance.

J Experimental Setup

This section outlines the hyperparameters used for **`FlowChef`** and baseline methods in solving inverse problems.

Pixel-Space Models. All evaluations were conducted using the Rectified Flow++ checkpoint. Since public implementations of OT-ODE and D-Flow are unavailable, we implemented these methods manually based on the provided pseudocode and performed hyperparameter tuning to ensure op-

Hyperparameter	OT-ODE	D-Flow	PnP-Flow	FlowChef
Iterations / NFEs	200	20	50	200
Optimization per iteration	1	-	-	1
Optimization per denoising	-	50	-	-
Avg. sampling steps	-	-	5	-
Guidance scale	1	1	1	500
Cost function	L1	L1**2	L1	MSE
initial time (1 means noise)	0.8	-	-	-
blending strength	-	0.05	-	-
inversion	×	✓	×	×
learning rate	1	1	1	1

Table 4: Hyperparameters for solving inverse problems using pixel-space models.

Hyperparameter	D-Flow	RectifID	FlowChef
Iterations / NFEs	10	4	100
Optimization per iteration	-	-	1
Optimization per denoising	20	400	-
Blending strength	0.1	-	-
Guidance scale	0.5	0.5	0.5
Cost function	MSE	MSE	MSE
Learning rate	0.5	1	0.02
Optimizer	Adam	SGD	Adam
loss multiplier (latent/pixel)	0.000001	0.0001 / 100000	0.001/1000
inversion	✓	×	×

Table 5: Hyperparameters for solving inverse problems using latent-space models (InstaFlow).

Model	Hyperparameters	Chage Object	Add Object	Remove Object	Change Attrbiute	Chage Pose	Change Color	Change Material	Change Background	Change Style
FlowChef (InstaFlow)	**Learning rate**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1.0	0.5
	Max setps	50	50	50	50	20	30	50	50	30
	Optimization steps	1	1	3	2	2	2	2	4	1
	Inference steps	50	50	50	50	50	50	50	50	50
	Full source steps	30	30	0	10	10	20	20	0	30
	Edit guidance scale	2.0	2.0	2.0	4.5	8.0	8.0	4.0	3.0	6.0
FlowChef (Flux)	**Learning rate**	0.4	0.5	0.5	0.5	0.5	0.4	0.5	0.5	0.4
	Optimization steps	1	1	1	1	1	1	1	1	1
	Inference steps	30	30	30	30	30	30	30	30	30
	Full source steps	5	5	0	2	5	3	5	0	5
	Edit guidance scale	4.5	4.5	4.5	4.5	7.5	10.0	4.5	0.0	10.0

Table 6: Hyperparameter examples for which various editing tasks can be performed (following Algorithm 2). Notably, the **`FlowChef`** (Flux) variant can be further optimized for task-specific settings that will follow Algorithm 1 with a careful selection of hyperparameters.

Metric	OT-ODE	PnP-Flow	D-Flow	**`FlowChef`**
VRAM (GB)	0.70	0.40	6.44	**0.43**
Time (sec)	10.39	5.23	80.42	**4.31**

Table 7: Compute requirement comparisons on a A6000 GPU.

timal performance. Notably, DPS and FreeDoM hyperparameters are the same as the **`FlowChef`**. Table 4 provides a detailed overview of the hyperparameters used for each baseline.

Latent-Space Models. For latent-space models, we extended D-Flow to the InstaFlow pretrained model, repurposed RectifID for inverse problems, and fine-tuned the hyperparameters for optimal results. The best-performing hyperparameters for each baseline are listed in Table 5. We utilized their baseline implementations for diffusion model-based approaches such as Resample and PSLD-LDM, modifying only the number of inference steps. Specifically, we used 100 NFEs for Resample and 100/500 NFEs for PSLD.

K EXTENDED RESULTS

Classifier Guidance: Style Transfer. We conducted classifier-guided style transfer experiments using 100 randomly selected style reference images paired with 100 random prompts. The objective was to generate stylistic images that align visually with the reference style while adhering to the prompt. A pretrained CLIP model was used for evaluation, and we report both CLIP-T and CLIP-

Method	CLIP-I (↑)	CLIP-T (↑)	VRAM	Time
FreeDoM	**0.5343**	0.2541	17GB	80 sec
MPGD	0.5285	**0.2616**	**16GB**	20 sec
RetifID	0.4583	0.1702	18GB	30 sec
D-Flow	0.4851	0.2591	23GB	5 sec
FlowChef(10 NFEs)	0.5044	**0.2655**	**14GB**	**2 sec**
FlowChef(30 NFEs)	0.5301	0.2600	**14GB**	**7 sec**
FlowChef(30 NFEs × 2)	**0.5531**	0.2478	**14GB**	12 sec

Table 8: Comparison of Various Classifier Guided Style Transfer.

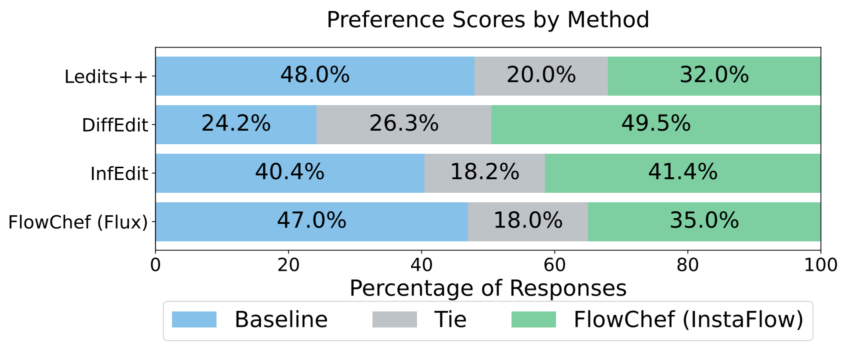


Figure 5: Human preference analysis for image editing.

Methods	CLIP-I (↑)	CLIP-T (↑)	Time (↓)
RF-Inversion	**0.8573**	0.2790	∼31 sec
FlowChef (ours)	0.8269	**0.2828**	**∼15 sec**

Table 9: Comparison of **FlowChef** with concurrent work RF-Inversion on top of Flux for editing task "wearing glasses".

S scores Radford et al. (2021). For baseline comparisons, we included diffusion-based methods FreeDoM and MPGD and flow-based methods D-Flow and RectifID, which were extended for this task. The backbone was fixed to Stable Diffusion v1.5 (SDv1.5), with **FlowChef** evaluated in its InstaFlow variant to ensure a consistent comparison. Both quantitative and qualitative results are presented in Table 8, demonstrating the effectiveness of **FlowChef** in this setup.

Multiobject editing & 3D generations. To highlight the versatility and effectiveness of **FlowChef**, we extended our method to tackle multi-object image editing and 3D multiview generation. Figure 7 demonstrates **FlowChef** (Flux) performing complex multi-object edits, such as simultaneously modifying two pots and hats. Notably, this capability relies on the base model's ability to understand textual instructions effectively. **FlowChef** leverages this strength of Flux, achieving edits without requiring inversion, a significant advantage over traditional methods. In Figure 6, we explore **FlowChef**'s multiview synthesis capability, inspired by Score Distillation Sampling (SDS) Poole et al. (2023). By incorporating the core idea of **FlowChef** for model steering into recent work on RFDS Yang et al. (2024), we evaluate its effectiveness for 3D view generation. While **FlowChef** does not improve inference efficiency or reduce cost compared to RFDS-Rev Yang et al. (2024), it demonstrates competitive performance in generating high-quality multiview outputs. These results underline the adaptability of **FlowChef**, showcasing its potential for advanced generative tasks such as multi-object editing and 3D synthesis, while maintaining the state-of-the-art quality expected from RFMs.

RF-Inversion *vs.* FlowChef. In this section, we briefly compare **FlowChef** with the concurrent work, RF-Inversion, which introduces an inversion strategy for rectified flow models us-

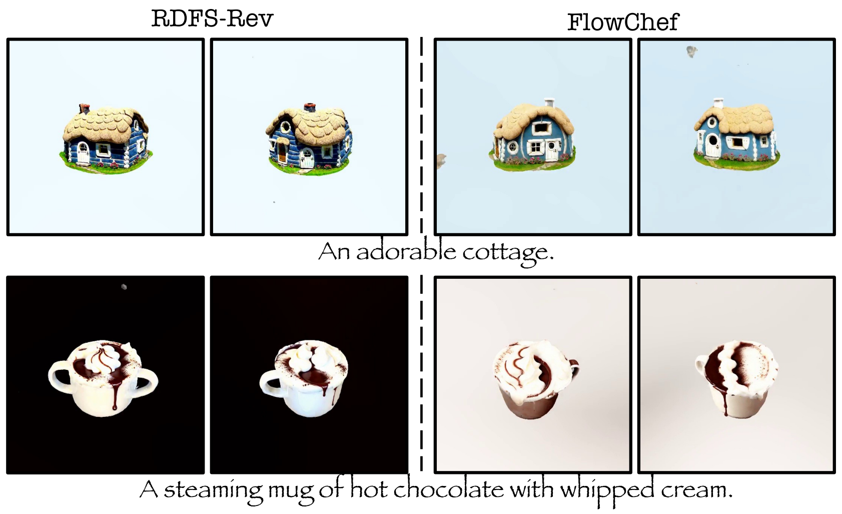


Figure 6: Extending **FlowChef** to 3D multiview synthesis.

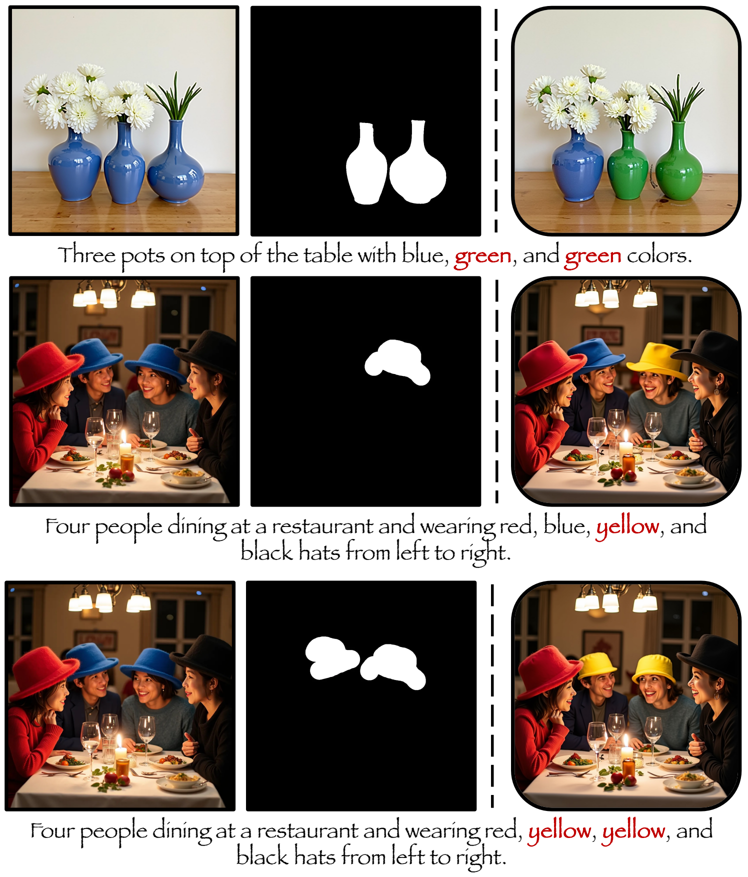

Three pots on top of the table with blue, green, and green colors.
Four people dining at a restaurant and wearing red, blue, yellow, and black hats from left to right.
Four people dining at a restaurant and wearing red, yellow, yellow, and black hats from left to right.

Figure 7: **FlowChef** (Flux) multi object editing examples.

ing a linear quadratic regulator perspective from optimal transport, particularly for image editing tasks. RF-Inversion relies on image inversion, significantly increasing compute time—nearly doubling it compared to **FlowChef**. To evaluate, we conducted a "wearing glasses" editing task using 256 randomly selected SFHQ face images on the Flux.1[dev] model. As shown in Table 9,

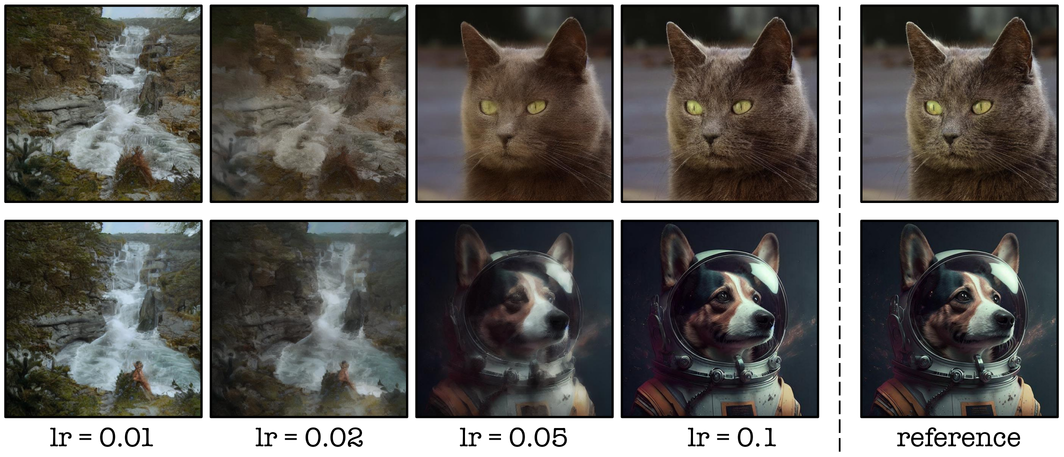


Figure 8: Effect of **`FlowChef`** learning rate with fixed 20 max steps and one optimization step on InstaFlow.

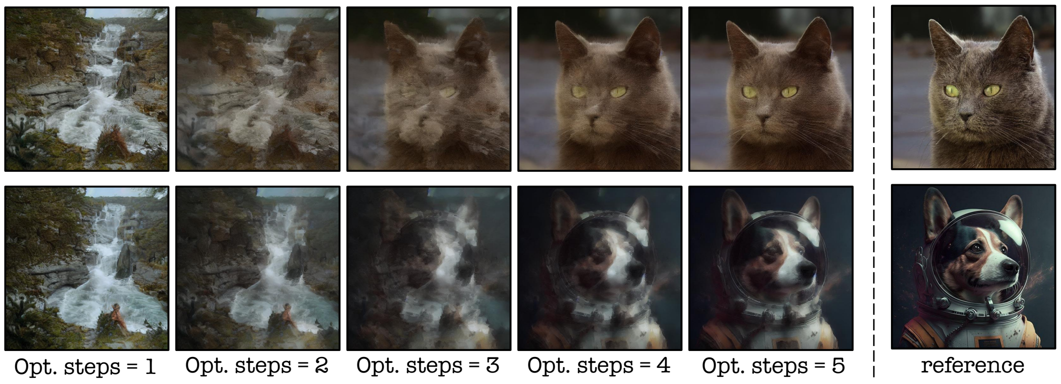


Figure 9: Effect of **`FlowChef`** optimization steps with fixed 20 max steps and 0.02 learning rate on InstaFlow.

`FlowChef` achieves competitive performance in half the time. At a high level, RF-Inversion can be viewed as a special case of **`FlowChef`**, where the starting point is an inverted image rather than random noise. We applied a similar editing strategy to both methods for a fair comparison, as outlined in Algorithm **??**, using a learning rate of 0.07, 20 optimization steps, and 30 total inference steps. On an A100 GPU, this configuration required approximately 15 seconds per inference. This comparison highlights the efficiency and versatility of **`FlowChef`** in handling image editing tasks.

L Hyper-parameter Study

Figures 8, 9, and 10 present an analysis of the impact of various hyperparameters on steering the InstaFlow model using **`FlowChef`**. Figure 8 demonstrates that a lower learning rate combined with a single optimization step is insufficient to effectively steer the model. Optimal performance is achieved with a learning rate of 0.1. Additionally, Figure 9 shows that lower learning rates necessitate more optimization steps to achieve convergence. Finally, Figure 10 illustrates how the denoising trajectory can be controlled by adjusting the learning rate and optimization steps, enabling recovery of the target sample with the desired accuracy. This control is particularly critical for image editing tasks, where striking the right balance between preserving the reference sample and applying the editing prompt is essential. Table 6 further highlights optimal hyperparameter settings for image editing tasks, providing valuable guidance for achieving high-quality edits. This study underscores the flexibility of **`FlowChef`** in adapting to diverse use cases by tuning these parameters effectively.

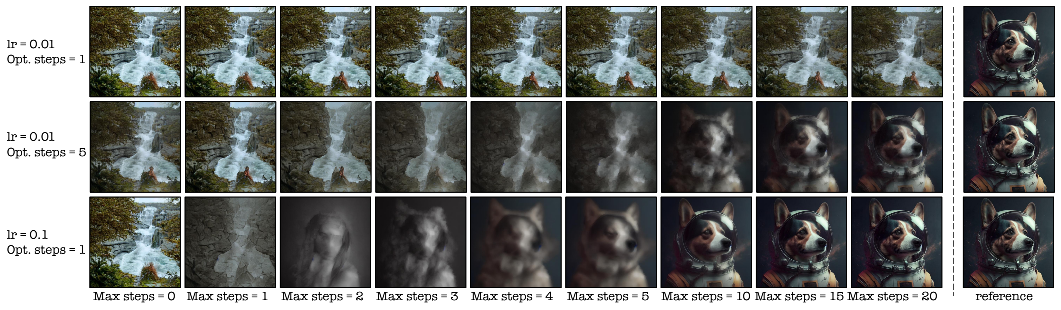


Figure 10: Effect of various **FlowChef**'s steering parameters with increasing maximum optimization steps on InstaFlow.

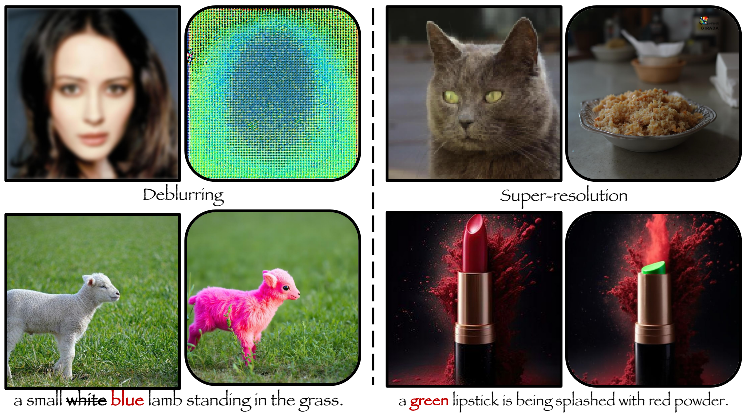

Deblurring
Super-resolution
a small ~~white~~ blue lamb standing in the grass.
a green lipstick is being splashed with red powder.

Figure 11: **FlowChef** (Flux) model failures on inverse problems and image editing.

M Qualitative Results

Figure 12 showcases additional qualitative examples of image editing tasks. For tasks such as changing materials or removing objects, **FlowChef** outperforms the baselines significantly. However, some limitations are noted: while **FlowChef** (InstaFlow) struggles to replace a cat with a tiger, InfEdit handles this task effectively, and Ledits++ exhibits difficulties. On the other hand, **FlowChef** (Flux) achieves superior results, though it replaces a dog with a tiger instead of a lion in one instance. In the final example, both Ledits++ and **FlowChef** successfully edit long hair into short hair. Importantly, the results in Figure 12 are presented without cherry-picking, using consistent hyperparameters for both baselines and **FlowChef**. Variability in outcomes may still arise due to random seeds and fine-tuned hyperparameter selection.

Figures 13, 14, 15, 16, 17, and 18 provide pixel-level qualitative results for various inverse problems, spanning inpainting, deblurring, and super-resolution tasks under both easy and hard scenarios. Readers are encouraged to zoom in to inspect these comparisons more closely. For each task, we randomly selected 10 CelebA examples and evaluated various baselines. Across all difficulty levels, FreeDoM, DPS, and PnPFlow demonstrate better performance than D-Flow and OT-ODE. However, **FlowChef consistently outperforms all baselines**, producing sharp and visually appealing results where other methods either fail outright or introduce excessive smoothness. Hard scenarios pose challenges for all methods, but **FlowChef** notably improves performance even under these conditions. While **FlowChef** shows promise, future work is needed to address potential adversarial effects and further enhance robustness.

N LIMITATIONS & FUTURE WORK

Limitations. While **`FlowChef`** represents a significant leap in steering RFMs for controlled generation, it shares some limitations with its baseline counterparts. Hyperparameter tuning remains a challenge, particularly due to differences in trajectory behavior. For instance, while InstaFlow trajectories are relatively linear, Flux.1[Dev] trajectories exhibit non-linearity, necessitating careful tuning. As shown in Figure 7, **`FlowChef`** (Flux) faces difficulties in deblurring and super-resolution tasks, which we attribute to the pixel-space loss and non-linear behavior of the VAE model. Importantly, these limitations occur in less than 10% of cases and can often be resolved by simply adjusting the random seed. Furthermore, due to Flux's lack of true classifier-free guidance (CFG), Algorithm 2 occasionally fails to perfectly execute color changes, sometimes producing the unaltered target image without reflecting the edit (see Figure 7). Despite these minor limitations, **`FlowChef`** still delivers state-of-the-art performance, making these challenges opportunities for further refinement rather than fundamental drawbacks.

Future Work. **`FlowChef`** opens a promising avenue for steering RFMs effortlessly with guaranteed convergence for controlled image generation. While this work extensively evaluates **`FlowChef`** on image generative models, future research should focus on expanding its capabilities to video and 3D generative models, areas that remain largely unexplored. Additionally, the current implementation assumes the availability of human-annotated masks for image editing. Automating this step with advanced attention mechanisms could make **`FlowChef`** a fully automated image editing framework. We encourage the research community to build upon this foundation to enhance its accessibility and functionality.

Ethical Concerns. As with all generative models, ethical concerns such as safety, misuse, and copyright issues apply to **`FlowChef`** Kim et al. (2024a;b). By enabling controlled generation with state-of-the-art RFMs, **`FlowChef`** can be leveraged for beneficial and harmful purposes. To mitigate these risks, future efforts should focus on solutions such as image watermarking, content moderation, and unlearning harmful behaviors. While these issues are not unique to **`FlowChef`**, addressing them will be key to ensuring its responsible use.

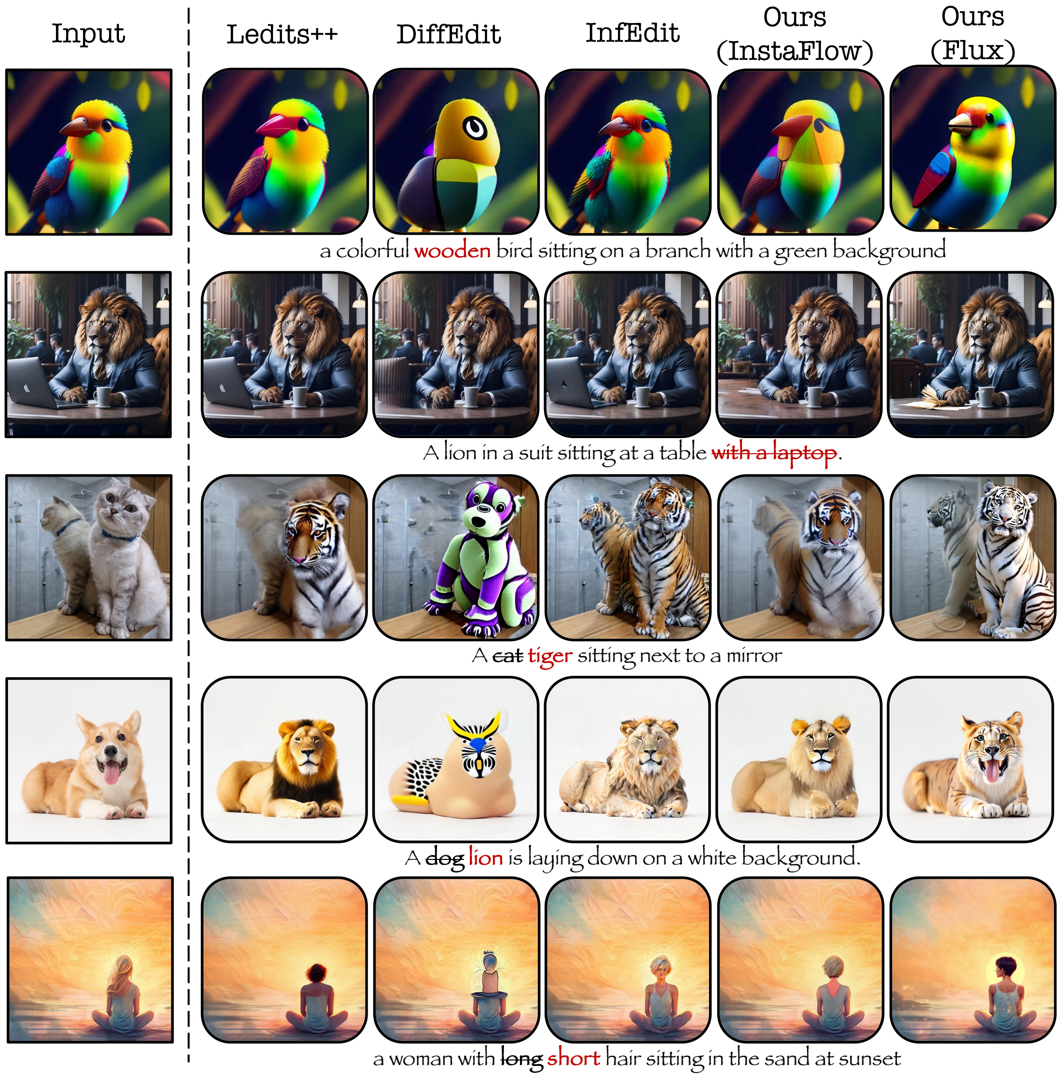

Input
Ledits++
DiffEdit
InfEdit
Ours (InstaFlow)
Ours (Flux)
a colorful wooden bird sitting on a branch with a green background
A lion in a suit sitting at a table ~~with a laptop~~.
A ~~cat~~ tiger sitting next to a mirror
A ~~dog~~ lion is laying down on a white background.
a woman with ~~long~~ short hair sitting in the sand at sunset

Figure 12: **Qualitative results on image editing.** Additional qualitative comparisons of **`FlowChef`** with the baselines.

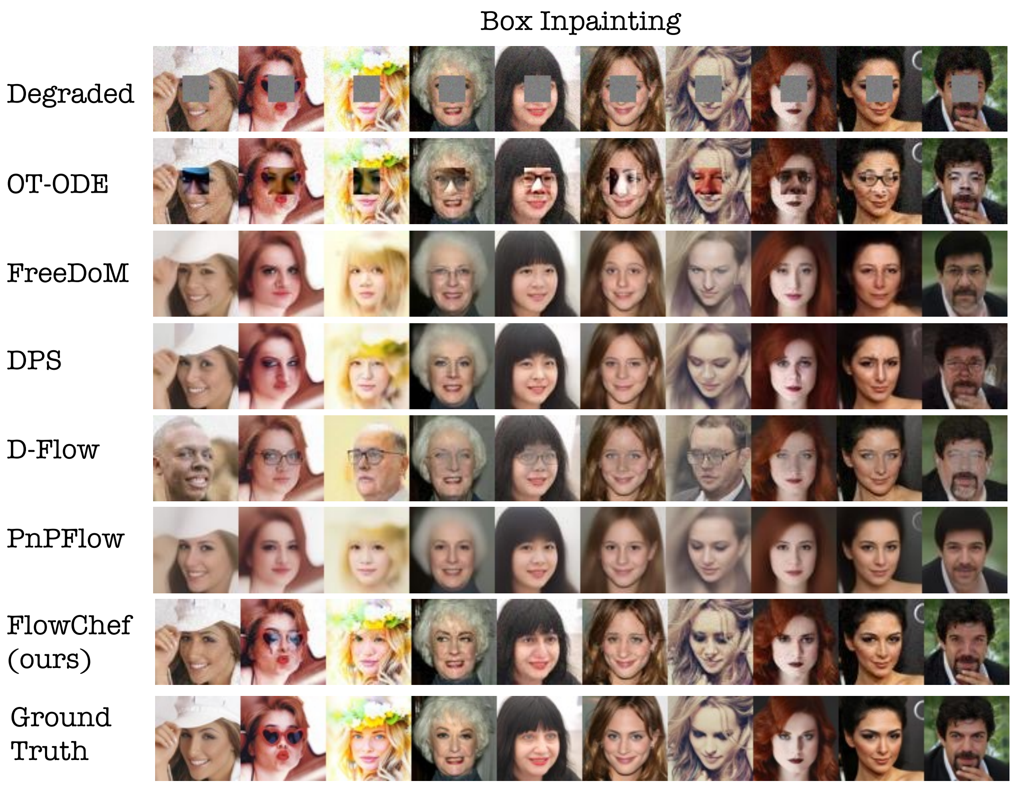


Figure 13: Qualitative examples of various methods for easy box inpainting task on RF++.

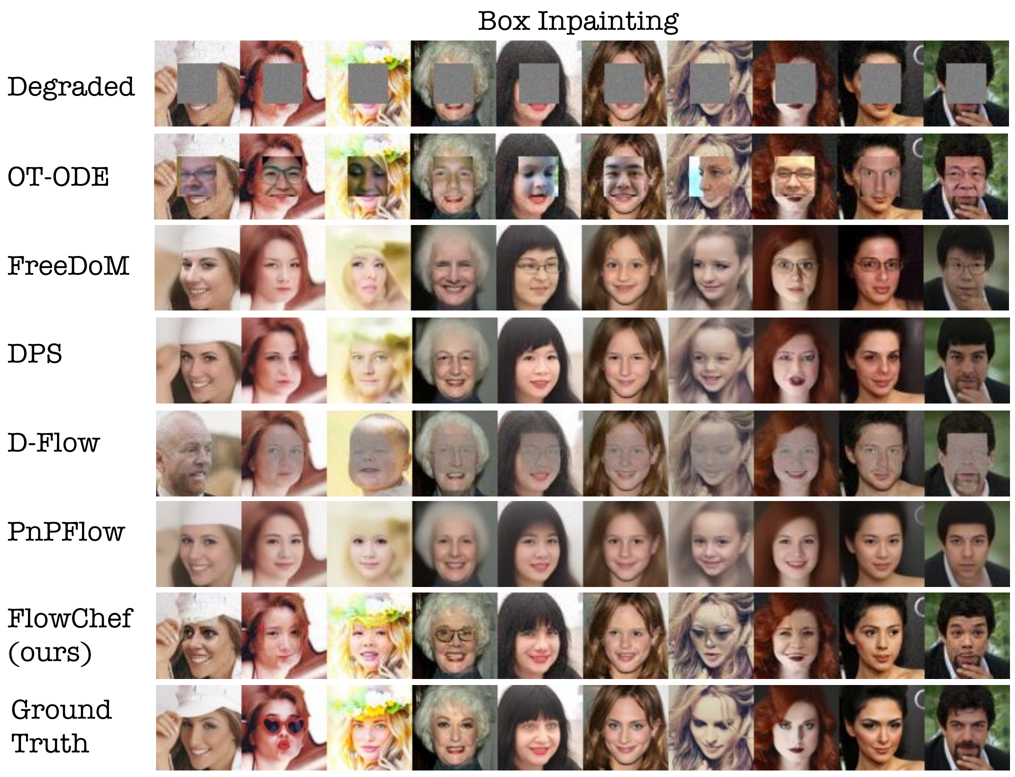


Figure 14: Qualitative examples of various methods for hard box inpainting task on RF++.

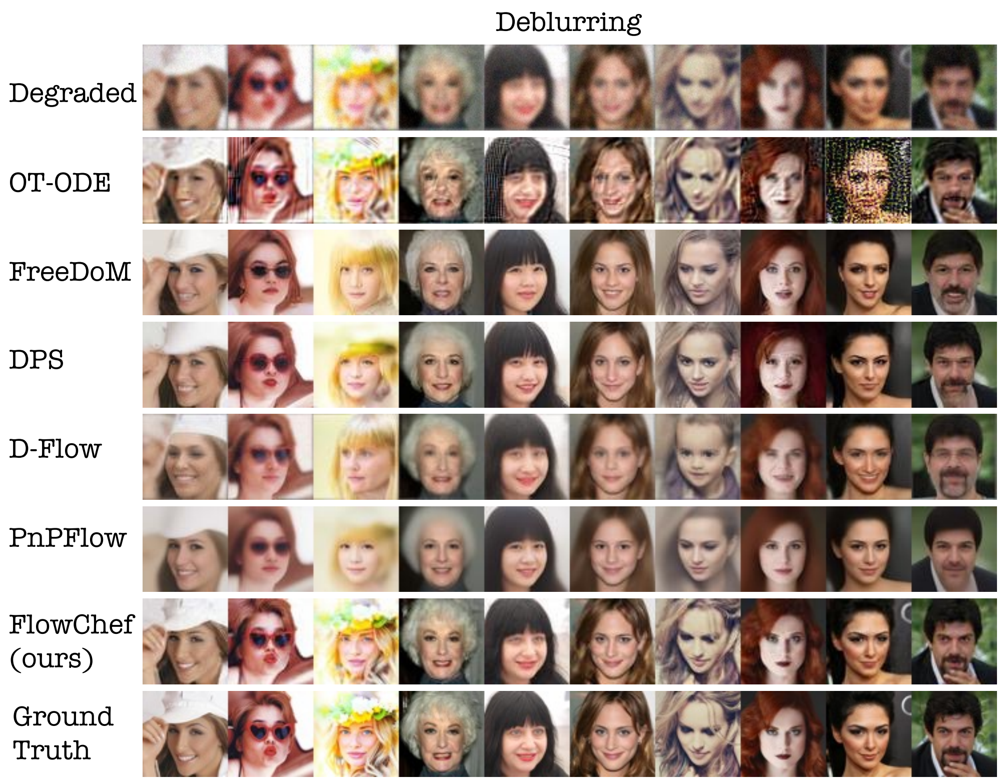


Figure 15: Qualitative examples of various methods for an easy deblurring task on RF++.

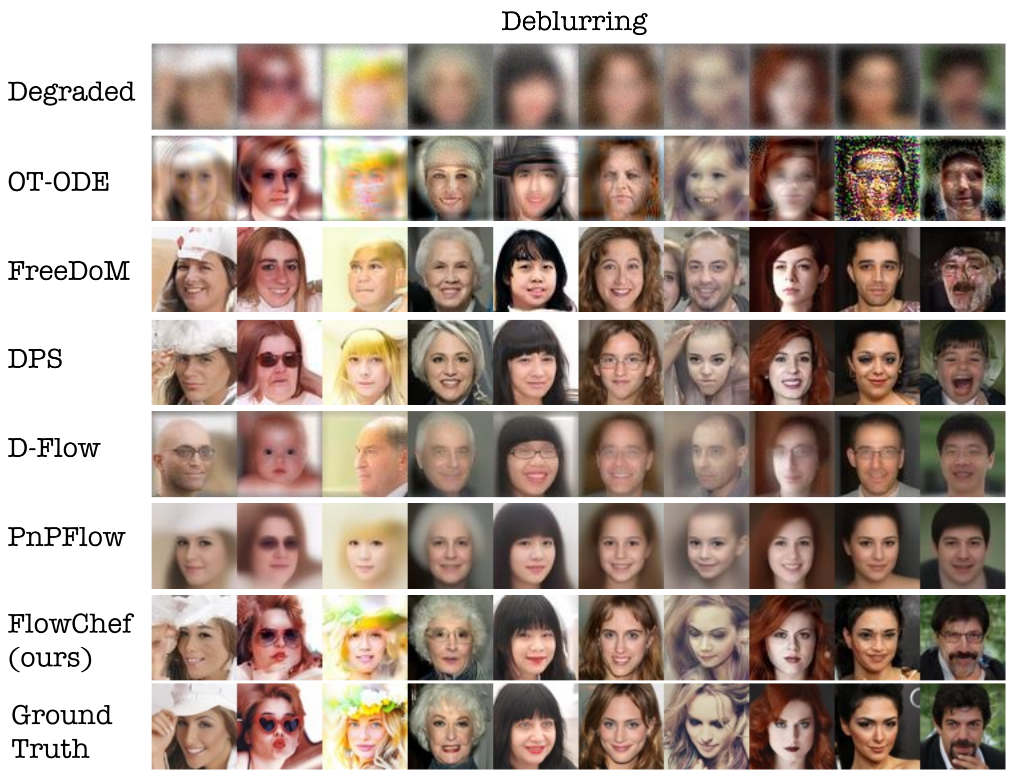


Figure 16: Qualitative examples of various methods for the hard deblurring task on RF++.

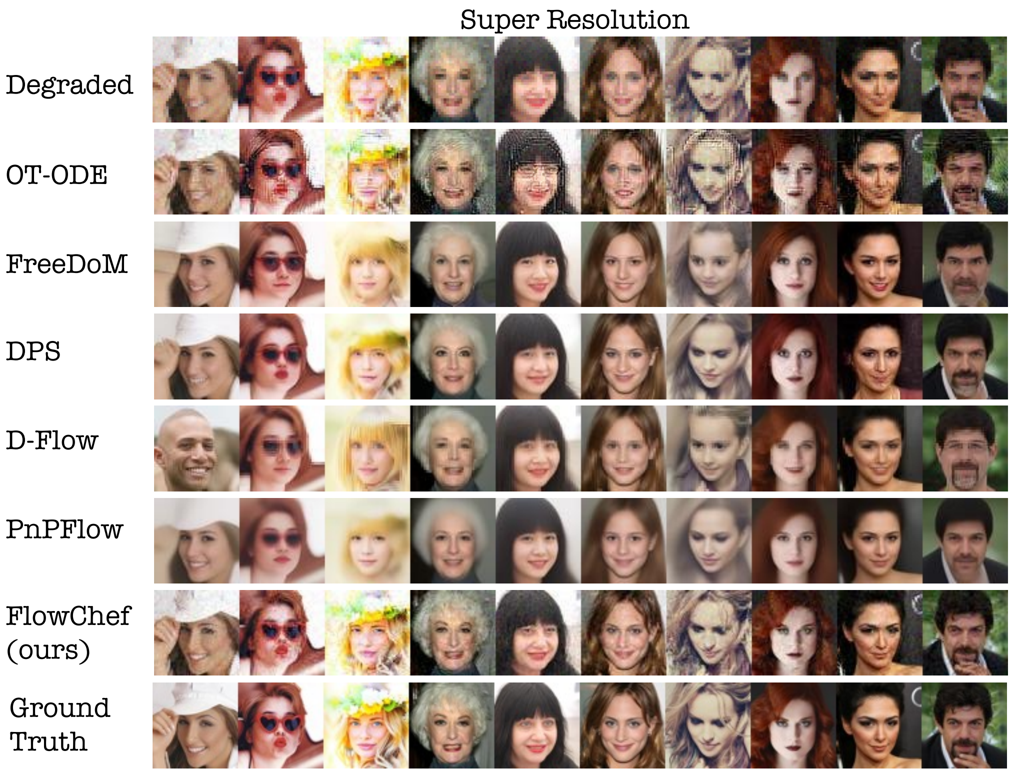


Figure 17: Qualitative examples of various methods for an easy super-resolution task on RF++.

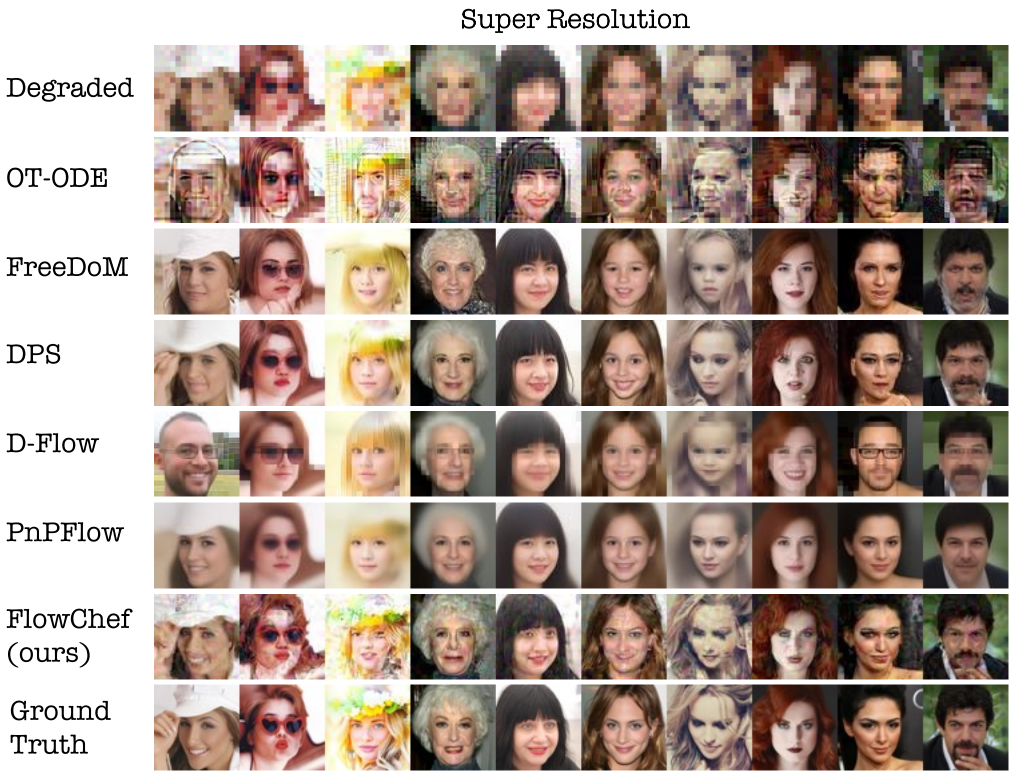


Figure 18: Qualitative examples of various methods for the hard super-resolution task on RF++.