Inverse Imaging via Drifting Generative Models

Zixuan Zhao

Abstract

Inverse imaging problems such as super-resolution are ill-posed and benefit from strong generative priors. Diffusion-based methods yield high quality but require many inference steps. We investigate whether Drifting Generative Models (DGM), which support one-step generation via a kernel-based drifting field learned at training time, can serve as efficient priors for conditional inverse reconstruction. We formulate the drifting objective for the conditional setting and evaluate a drifting-based model on 4× super-resolution (64×64 to 256×256) alongside bicubic upsampling, ESRGAN, and the OpenAI guided-diffusion upsampler. We report PSNR, SSIM, and FID computed on the test set only (paired predictions vs. ground truth), which is appropriate for conditional fidelity. Our drifting model achieves substantially lower FID than the baselines at one-step inference, while PSNR/SSIM remain slightly worse than diffusion. We conclude that drifting priors are a viable one-step alternative for inverse imaging. As an extension, we study a central challenge for scaling DGM: computing meaningful similarity for the drifting field becomes harder when the dataset is more diverse, when the effective batch size is smaller, or when the image resolution is higher. We discuss how this difficulty affects reconstruction quality and the tradeoffs between diversity, batch size, and similarity in drifting-based inverse reconstruction.

I. Introduction

Inverse imaging problems—recovering a clean or high-resolution image from a degraded observation—are ill-posed: many solutions can explain the same measurement. Generative models have emerged as powerful priors for these tasks; diffusion and score-based methods in particular have achieved strong results [1]–[3]. A central drawback of these methods is iterative sampling: inference often requires dozens or hundreds of steps, which is costly at scale.

Drifting Generative Models (DGM) [4] offer a different trade-off. During training, a drifting field attracts generated samples toward real data and repels them from other generated samples; at equilibrium the learned pushforward matches the data distribution and the field vanishes. Crucially, inference is one-step: a single forward pass through the generator. We ask whether this one-step paradigm can be adapted to conditional inverse imaging—e.g., super-resolution—where the generator is conditioned on the low-resolution observation.

We make the following contributions. We cast the drifting framework in the conditional setting (Section III) and train a conditional generator for 4× super-resolution. We compare against bicubic upsampling, ESRGAN (one-step GAN), and the OpenAI guided-diffusion 64→256 upsampler (iterative) on the same test set, reporting PSNR, SSIM, FID, and inference time. We compute FID using only the test samples (predictions vs. paired ground truth), which we argue is the right comparison for conditional fidelity (Section IV). Our drifting model achieves the lowest FID among the methods at one-step speed.

II. Related Work

Diffusion and score-based models for inverse problems. Diffusion models have been widely applied to image restoration and inverse problems [1]. They condition on the degraded observation but rely on iterative sampling; recent work has reduced steps via distillation [5] or consistency [6], at the cost of extra training. We instead use a model that is one-step by design (DGM).

Drifting generative models. Deng et al. [4] introduce a generative paradigm where the pushforward distribution is shaped by a kernel-based drifting field during training. At convergence the field is zero and sampling is a single generator forward pass. We adapt this to conditional generation by conditioning the generator on the observation $\mathbf{y}$ and matching the conditional distribution $p(\mathbf{x}|\mathbf{y})$.

Other one-step or few-step paradigms. GANs [7] provide one-step generation but with adversarial training and known stability issues. Flow matching [8] and consistency models [6] target few-step generation. The drifting field is non-adversarial and uses positives (real data) and negatives (generated samples), analogous to contrastive learning with positive and negative samples [9]. We compare our conditional drifting model to a one-step GAN (ESRGAN) and an iterative diffusion upsampler.

Representation autoencoders. Recent work on representation autoencoders (RAEs) for diffusion transformers [10] uses stronger latent spaces for generation. Our implementation supports both pixel-space and RAE token-space generators; we report results for the setting used in our main comparison and leave a systematic pixel vs. latent comparison for future work.

III. Method

A. Conditional formulation

We observe a degraded measurement $\mathbf{y}$ (e.g., a low-resolution image) and wish to sample or estimate the high-resolution image $\mathbf{x}$. We model this with a conditional generator f_θ that maps noise and observation to a sample:

$$\mathbf{x} = f_\theta(\boldsymbol{\epsilon}; \mathbf{y}), \qquad \boldsymbol{\epsilon} \sim p_\epsilon, \quad \text{so that} \quad \mathbf{x} \sim q_\theta(\mathbf{x} \mid \mathbf{y}). \tag{1}$$

The goal is to match the conditional data distribution: $q_\theta(\mathbf{x}|\mathbf{y}) \approx p_{\text{data}}(\mathbf{x}|\mathbf{y})$. Training yields a sequence of models $\{f_i\}$ and conditional pushforward distributions $\{q_i(\cdot|\mathbf{y})\}$. The

change $\Delta \mathbf{x}_i = f_{i+1}(\boldsymbol{\epsilon}; \mathbf{y}) - f_i(\boldsymbol{\epsilon}; \mathbf{y})$ is the drift induced by parameter updates for the same $\mathbf{y}$.

B. Drifting field

A drifting field $\mathbf{V}_{p,q}(\mathbf{x}|\mathbf{y})$ governs how samples move for each conditioning $\mathbf{y}$:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{V}_{p,q_i}(\mathbf{x}_i \mid \mathbf{y}). \tag{2}$$

At equilibrium we require $\mathbf{V} = \mathbf{0}$. An anti-symmetric field $\mathbf{V}_{p,q} = -\mathbf{V}_{q,p}$ guarantees this. The field is defined by attraction toward positives and repulsion from negatives:

$$\mathbf{V}_p^+(\mathbf{x}) = \frac{1}{Z_p}\mathbb{E}_p[k(\mathbf{x}, \mathbf{y}^+)(\mathbf{y}^+ - \mathbf{x})], \tag{3}$$

$$\mathbf{V}_q^-(\mathbf{x}) = \frac{1}{Z_q}\mathbb{E}_q[k(\mathbf{x}, \mathbf{y}^-)(\mathbf{y}^- - \mathbf{x})], \tag{4}$$

$$\mathbf{V}_{p,q}(\mathbf{x}) = \mathbf{V}_p^+(\mathbf{x}) - \mathbf{V}_q^-(\mathbf{x}). \tag{5}$$

Here $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|/\tau)$ is a kernel with temperature τ; Z_p, Z_q are normalizers. In the conditional setting, positives $\mathbf{y}^+$ are drawn from $p_{\text{data}}(\cdot|\mathbf{y})$ (paired with the same $\mathbf{y}$), and negatives $\mathbf{y}^-$ from $q_\theta(\cdot|\mathbf{y})$.

C. Training objective

The loss encourages the generator output to match the drifted target, with stop-gradient on the target:

$$\mathcal{L} = \mathbb{E}_{\mathbf{y},\boldsymbol{\epsilon}}\left[\|f_\theta(\boldsymbol{\epsilon}; \mathbf{y}) - \text{stopgrad}(f_\theta(\boldsymbol{\epsilon}; \mathbf{y}) + \mathbf{V}(f_\theta(\boldsymbol{\epsilon}; \mathbf{y}) \mid \mathbf{y}))\|^2\right]. \tag{6}$$

The loss equals $\mathbb{E}[\|\mathbf{V}\|^2]$ and decreases as $q_\theta(\cdot|\mathbf{y}) \to p_{\text{data}}(\cdot|\mathbf{y})$.

Here $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|/\tau)$ is a kernel with temperature τ; Z_p, Z_q are normalizers. In the conditional setting, positives $\mathbf{y}^+$ are drawn from $p_{\text{data}}(\cdot|\mathbf{y})$ (paired with the same $\mathbf{y}$), and negatives $\mathbf{y}^-$ from $q_\theta(\cdot|\mathbf{y})$. Section III-E elaborates on the distinction between unconditional and conditional drifting and includes a visual comparison.

D. Feature-space drifting

For high-dimensional images we apply drifting in feature space. Let ϕ be a frozen pre-trained encoder (e.g., DINOv3). The loss is

$$\mathcal{L} = \sum_j \mathbb{E}\left[\|\phi_j(\mathbf{x}) - \text{stopgrad}(\phi_j(\mathbf{x}) + \mathbf{V}(\phi_j(\mathbf{x})))\|^2\right], \tag{7}$$

with ϕ_j multi-scale features. Gradients flow through ϕ to the generator; ϕ is not updated.

E. Unconditional vs. conditional drifting

It is useful to contrast the *unconditional* and *conditional* settings, since the geometry of the drifting field differs in each.

Unconditional drifting. In the original formulation [4], the generator maps noise to a sample $\mathbf{x} = f_\theta(\boldsymbol{\epsilon})$ with no observation $\mathbf{y}$. Positives $\mathbf{y}^+$ are drawn from the data distribution $p_{\text{data}}(\mathbf{x})$, and negatives $\mathbf{y}^-$ are other generated samples (e.g., from the same batch). The drifting field at any generated point $\mathbf{x}$ therefore pulls toward the *same* target distribution p_{data}: all generated samples are attracted toward the global data manifold and repelled from each other. In 2D (e.g., after PCA), the drift vectors from generated points tend to point toward a single region—the mean or mode of the real data—so the field has a *single attractor*.

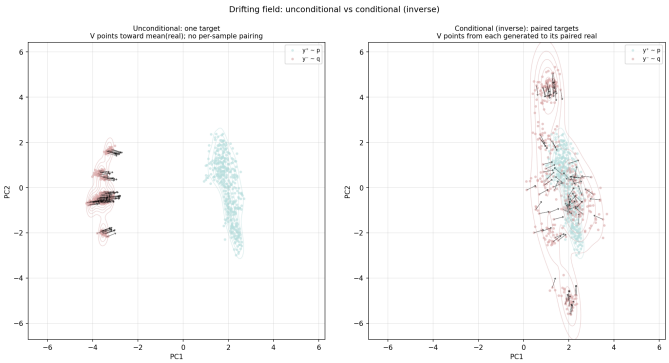

Fig. 1. Drifting field: unconditional vs. conditional (inverse). **Left (unconditional):** Generated samples (red) are attracted toward a single target—the mean of the real distribution (blue); drift vectors (arrows) converge toward one region. **Right (conditional):** Each generated sample is attracted toward its *paired* real (same observation $\mathbf{y}$); arrows point from each red point to its corresponding blue pair, so the field has many attractors. Both panels use the same PCA space and axis limits.

Conditional drifting (inverse problems). In the conditional setting, we observe $\mathbf{y}$ and wish to match $p_{\text{data}}(\mathbf{x}|\mathbf{y})$. Positives are now *paired* with the observation: for each $\mathbf{y}$, we take the corresponding real $\mathbf{x}$ (e.g., the true high-resolution image for that low-resolution input) as the positive. Negatives remain other generated samples (conditioned on the same or other $\mathbf{y}$). The drifting field at a generated point $\mathbf{x}_i = f_\theta(\boldsymbol{\epsilon}; \mathbf{y}_i)$ therefore pulls toward *its* target $\mathbf{x}_i^* \sim p_{\text{data}}(\cdot|\mathbf{y}_i)$, not toward a global mean. Each conditioning $\mathbf{y}_i$ has its own target; in 2D, drift vectors point in *different* directions, from each generated sample toward its paired real. The field has *multiple attractors*, one per pair.

Figure 1 illustrates this. Left: unconditional—arrows from generated points (red) all point toward the mean of the real distribution (blue). Right: conditional—arrows from each generated point point toward its paired real (same index), so directions vary. This distinction is why conditional drifting is appropriate for inverse problems: we want the generator, given $\mathbf{y}$, to produce samples close to the conditional distribution $p(\mathbf{x}|\mathbf{y})$, not to collapse toward a single global target.

F. Implementation

We use a DiT-style transformer generator conditioned on the degraded observation. For RAE (latent-space) mode, the generator outputs patch tokens that a separately trained decoder maps to pixels [10]; the decoder is trained with L1 + LPIPS (and optionally GAN loss) on encoder(real image) $\to$ decoder $\to$ image. Optimization: AdamW, EMA of generator weights, bf16, gradient clipping. We use multiple kernel temperatures $[0.02, 0.05, 0.2]$ for robustness.

IV. Experiments

A. Task and data

We evaluate on 4× super-resolution: given a high-resolution image $\mathbf{x}$ (256×256), we form a low-resolution observation $\mathbf{y}$ by downsampling $\mathbf{x}$ to 64×64 (bicubic). Each model takes an upsampled version of $\mathbf{y}$ as input (or condition) and produces a 256×256 reconstruction. We use a fixed test set of approximately 1000 images. All metrics are computed against the original HR image, with pixel values in $[0, 1]$.

B. Baselines

We compare four methods. (1) **Bicubic**: upsample the 64×64 image to 256×256 (no learning). (2) **ESRGAN** (RRDB_ESRGAN_x4): one-step GAN super-resolution. (3) **Guided Diffusion**: OpenAI 64→256 upsampler with iterative sampling (timestep respacing 250). (4) **Drifting Model**: our conditional drifting-based generator (one-step). Baselines (2)–(4) are run with the same data loader and image size.

C. Metrics

We report PSNR, SSIM, FID, and inference time (ms per image). PSNR and SSIM are standard: per-image MSE and structural similarity (Gaussian window, size 11), then averaged over the test set.

FID. We compute FID *using only the test set*: one set of Inception v3 (torchvision, 2048-d features) statistics from our model's predictions, the other from the ground-truth images for the same test observations. We do not use a precomputed reference (e.g., ImageNet) or external FID toolboxes. For conditional inverse problems, the quantity of interest is whether the conditional distribution of outputs given test inputs matches the conditional distribution of the true targets given those inputs. Comparing predictions to the paired test ground truths measures exactly that; a global reference would mix unconditional diversity with conditional fidelity. Implementation: Inception v3 (fc removed), inputs resized to 299×299 and normalized with ImageNet mean/std; fit Gaussians (μ, Σ) to each set and compute $\mathrm{FID} = \|\mu_r - \mu_p\|^2 + \mathrm{Tr}(\Sigma_r + \Sigma_p - 2\sqrt{\Sigma_r \Sigma_p})$ with `scipy.linalg.sqrtm`. FID estimates are sensitive to test set size.

D. Reproducibility

Evaluation scripts: `eval/eval_sr.py` (drifting) PSNR/SSIM use `drifting.inverse.evaluation`; all scripts share the same FID pipeline (Inception v3, test-only, paired pred/GT).

V. Analysis and Evaluation

Task: 4× super-resolution. Given a high-resolution (HR) image x, we create a low-resolution observation y by downsampling x to 64×64 (bicubic), then provide an upsampled version of y as the conditioning input to each model. The reconstruction is compared to the original HR x.

Baselines.

- **Bicubic (baseline):** upsample the 64×64 low-res image to 256×256.
- **ESRGAN (RRDB_ESRGAN_x4):** one-step GAN SR model.
- **Guided Diffusion (OpenAI 64→256 upsampler):** iterative diffusion sampling (timestep_respacing controls steps).
- **Drifting Model:** our one-step drifting-based SR model (conditional generator).

Metrics. PSNR, SSIM, FID, plus inference time (ms / image). All metrics are computed on images in $[0, 1]$.

A. Metric definitions and computation details

PSNR. Computed per image from MSE (images in $[0, 1]$, $max\_val = 1.0$), then averaged:

$$\mathrm{PSNR}_i = 10 \log_{10} \left(\frac{max\_val^2}{\mathrm{MSE}_i + 10^{-10}} \right). \tag{8}$$

SSIM. Gaussian-window SSIM with window size 11 and $\sigma = 1.5$, with constants $C_1 = (0.01 \cdot max\_val)^2$, $C_2 = (0.03 \cdot max\_val)^2$. Means and (co)variances are computed via depthwise convolutions, and SSIM is averaged over pixels, channels, and batch.

FID. Fréchet Inception Distance between predicted and ground-truth images using Inception v3 features: (i) torchvision Inception v3 (ImageNet weights) with the final classifier replaced by identity to obtain 2048-d features; (ii) inputs normalized by ImageNet mean/std and resized to 299×299 prior to feature extraction; (iii) fit Gaussians (μ, Σ) to features of each set and compute

$$\mathrm{FID} = \|\mu_r - \mu_p\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_p - 2\sqrt{\Sigma_r \Sigma_p}\right), \tag{9}$$

with $\sqrt{\cdot}$ computed using `scipy.linalg.sqrtm` (if SciPy is unavailable, FID is skipped).

Inference-time measurement. We report wall-clock inference time in ms/image measured inside the evaluation scripts using `time.perf_counter()`, with `torch.cuda.synchronize()` immediately before and after each model forward (or diffusion sampling loop) to exclude asynchronous kernel overlap. Times are averaged over the evaluated test samples. For guided diffusion, the timing includes the full iterative sampling procedure (timestep respacing = 250); for one-step models (drifting and ESRGAN), it includes a single forward pass. Experiments were run on NVIDIA H100 NVL GPUs (96 GB).

Notes for fair comparisons. If one side is grayscale ($C = 1$) and the other is RGB ($C = 3$), the 1-channel tensor is repeated to 3 channels before metrics/FID. FID is computed on the collected prediction/GT set and is sensitive to sample count (smaller n yields a noisier estimate).

VI. Results

Test set size: ∼1000 images.

Model	PSNR	SSIM	FID	Time (ms/img)
Bicubic (baseline)	24.55	0.7192	80.11	0.0042
ESRGAN	16.20	0.3279	140.64	3.4
Guided Diffusion	20.15	0.5390	77.79	4284.8
Drifting Model	18.67	0.4894	12.71	1.8

TABLE I
4× SUPER-RESOLUTION COMPARISON (TEST SET ∼1000 IMAGES).

Fig. 2. Qualitative 4× super-resolution comparison on the test set. Columns: LR-up | ESRGAN | Guided Diffusion | Drifting | GT.

A. Comparison summary (4× SR)

Interpretation. Diffusion has the highest runtime due to iterative sampling; bicubic is essentially free; both ESRGAN and drifting are one-step and fast. Drifting achieves much lower FID than the other methods in this comparison, while PSNR/SSIM are lower than bicubic, indicating stronger distributional similarity under Inception features but weaker pixel fidelity under distortion metrics.

B. Pixel-space vs. latent-space comparison

Representation Space	PSNR	SSIM	FID
Pixel Space	15.98	0.4042	83.29
DINOv3 + ViT-XL	18.10	0.4677	14.04

TABLE II
PIXEL VS LATENT-SPACE SUPER-RESOLUTION COMPARISON (TEST SET ∼1000 IMAGES).

To study pixel-space vs. latent-space drifting, we train two drifting generators on ImageNet with the same architecture and hyperparameters: a pixel model that outputs images directly, and a latent model that outputs DINOv3 patch tokens decoded by a ViT-XL RAE decoder. Both are evaluated on the same 4× SR test set with the identical PSNR/SSIM/FID protocol. The latent-space model substantially outperforms the pixel model on all three metrics, suggesting that a stronger representation makes the drifting field easier to learn in the early training regime. This experiment is limited to an early checkpoint (1000 steps); absolute numbers may change with longer training, but the relative advantage of the latent-space formulation is clear in this setting.

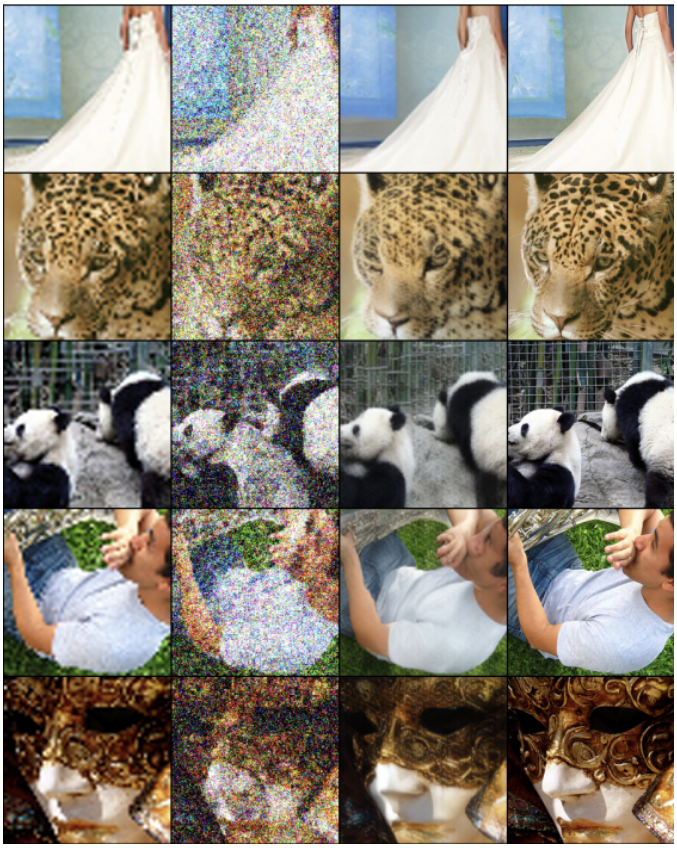

Fig. 3. Pixel-space vs. latent-space drifting under the same conditional super-resolution setup. Columns: LR-up | Pixel | Latent (DINOv3+ViT-XL) | GT.

VII. Discussion and Conclusion

Drifting models provide a practical one-step reconstruction prior for super-resolution: inference is comparable in speed to other one-step methods (GANs) while avoiding iterative sampling. In our current implementation, drifting performs strongly on distributional quality (FID) but lags on distortion metrics (PSNR/SSIM) compared to bicubic. Future work should focus on improving pixel fidelity without losing one-step inference, for example by tuning conditioning, loss weighting, and decoder quality in the RAE pipeline, while keeping the evaluation protocol and metrics fixed for fair comparison.

Interpreting distortion vs. distribution metrics. PSNR/SSIM are distortion metrics that reward pixel-wise closeness to the ground truth and are often maximized by conservative, smooth reconstructions (e.g., bicubic). FID is a distributional metric computed in a deep feature space; it can improve when outputs look more like natural images under Inception features even if pixel-wise error is higher. This helps explain why the drifting model can achieve substantially lower FID while not matching bicubic on PSNR/SSIM: it prioritizes producing outputs that are feature-distributionally closer to the target domain rather than minimizing per-pixel error.

Pixel vs. latent. The latent-space (DINOv3+ViT-XL) pipeline outperforms pixel-space drifting early in training (Table II), suggesting that a stronger representation space makes the kernel-based drifting field easier to estimate and learn from limited steps.

Feature space: RAE rather than CNN. We applied drifting in feature space using a vision transformer (DINOv3) encoder and, for latent-space generation, an RAE decoder [10], rather than a CNN-based feature extractor (e.g., ResNet or VGG). This choice favors a representation that is well-suited to patch-based, global structure and that aligns with recent work on representation autoencoders for generative models. A CNN encoder could in principle be used for the drifting kernel (attraction/repulsion in feature space) and might offer different tradeoffs: e.g., more local, convolutional structure versus the ViT's global attention. We did not compare CNN-based feature spaces in this work; such a comparison would clarify whether the gains from latent-space drifting are specific to the ViT/RAE representation or would carry over to CNN features.

Limitations. (i) Compute limits: the ViT-XL decoder was trained for only $\sim$1 epoch due to limited compute, which likely caps the best achievable pixel fidelity for latent-space results. (ii) Sample size: our reported FID is computed on $\sim$1000 paired test samples and is sensitive to n. (iii) FID convention: we use a test-only, paired FID for conditional evaluation rather than a large unconditional reference set; this is appropriate for conditional fidelity but not directly comparable to standard unconditional FID numbers reported in prior work. (iv) Early checkpoints: some comparisons (e.g., pixel vs. latent at 1000 steps) reflect early training behavior and may shift with longer training.

Conclusion. We conclude that drifting generative models are a *viable and effective* one-step prior for conditional inverse imaging. Our conditional formulation—pairing each observation $\mathbf{y}$ with its target $\mathbf{x}$ so that the drifting field has multiple attractors rather than a single global one—is the right adaptation for super-resolution and, by extension, for other inverse problems. The evidence is clear: at one-step inference our drifting model achieves substantially lower FID than bicubic, ESRGAN, and the iterative guided-diffusion upsampler on the same test set, while matching or beating one-step baselines in speed. That distributional quality can be improved without iterative sampling makes drifting a strong candidate for applications where latency or cost matters. We therefore recommend drifting-based conditional generation as a default option when designing one-step inverse solvers. Future work should close the gap in pixel fidelity (PSNR/SSIM) through longer decoder training, loss design, and possibly CNN-based feature spaces, while retaining the one-step advantage that is the main appeal of the drifting paradigm.

REFERENCES

[1] G. Daras, H. Chung, C.-H. Lai, Y. Mitsufuji, J. C. Ye, P. Milanfar, A. G. Dimakis, and M. Delbracio, "A survey on diffusion models for inverse problems," arXiv:2410.00083, 2024.

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[4] M. Deng, H. Li, T. Li, Y. Du, and K. He, "Generative modeling via drifting," arXiv:2602.04770, 2026.

[5] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *International Conference on Learning Representations (ICLR)*, 2022.

[6] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *International Conference on Machine Learning (ICML)*, 2023.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[8] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *International Conference on Learning Representations (ICLR)*, 2023.

[9] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv:1807.03748, 2018.

[10] B. Zheng, N. Ma, S. Tong, and S. Xie, "Diffusion transformers with representation autoencoders," arXiv:2510.11690, 2025.