One-step Generation in the Post Diffusion Era

Junoh Kang

12th March, 2026

Computer Vision Laboratory
ECE, Seoul National University
junoh.kang@snu.ac.kr

Overview

1. Intro: Limitations of Diffusion Models and One-Step Modeling
   - Consistency Models
   - MeanFlow / CTM
   - Distribution Matching (DMD)
   - Summary & Key Question

2. Main: Drifting Models
   - Method
   - Experiments

3. Discussion

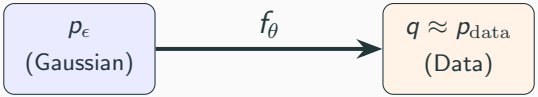
Intro: Limitations of Diffusion Models and One-Step Modeling

The Goal of Generative Modeling

Pushforward Formulation

Learn a network f such that:

$$q = f_{\#}\, p_{\epsilon} \;\approx\; p_{\text{data}}$$



- How do we learn this complex mapping f?
- Learning the full pushforward in one shot is difficult

Diffusion / Flow Models: Inference-time Iteration

Key strategy: Decompose a complex transformation into many small steps

Diffusion Models

Iterative denoising along an SDE:

$$x_{i+1} = x_i + \Delta x_i$$

250+ denoising steps

Flow Matching

Solve an ODE with a numerical solver:

$$\frac{dx}{dt} = v_\theta(x, t)$$

Euler solver: 50–250 NFE

Common Problem

Iteration happens at **inference time** $\Rightarrow$ slow generation

Attempts Toward One-Step Generation

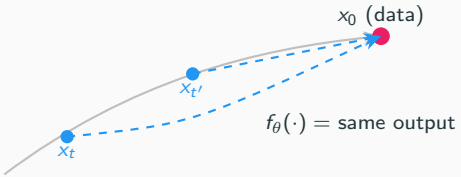
Fundamental question: Must we iterate at inference time?

Consistency Model	CTM / MeanFlow	Distribution Matching (DMD)
The ODE solver always maps to the same endpoint along a trajectory	**Learn average velocity, not just tangential velocity.**	**Match distributions using scores from a pretrained diffusion model**

- Each method attempts to resolve the **multi-step bottleneck** of diffusion/flow

Consistency Models [Song et al., 2023] (ICML 2023)

Key Idea: Points on the same ODE trajectory must map to the same data point



Consistency constraint: $f_\theta(x_t, t) = f_\theta(x_{t'}, t')$
1-step generation: $f_\theta(x_T, T) \to x_0$

Requires an underlying ODE trajectory for training

- CD leverages pretrained diffusion models to obtain two points on the same trajectory.
- CT replaces them with noise-shared pairs that approximate this supervision.

Limitations of Consistency Models

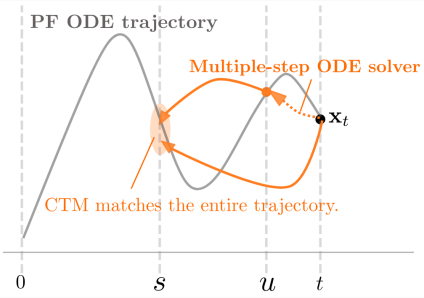
1. **Unstable training**
   - Highly sensitive to curriculum design

2. **Indirect training signal**
   - $f_\theta(x_t, t) = f_\theta(x'_t, t')$ has trivial solution
   - Require many engineering tricks

3. **Dependence on diffusion teacher (CD)**
   - Distillation requires a pretrained diffusion model

CTM [Kim et al., 2024] (ICLR 2024)

Key Idea: Direct and multi-step jumps along the same ODE trajectory are consistent



Consistency constraint: $G_\theta(x_t, t, s) = G_\theta(G_\theta(x_t, t, u), u, s)$

1-step generation: $x_0 = G_\theta(x_T, T, 0)$

MeanFlow [Geng et al., 2025] (NeurIPS 2025 Oral)

Previously covered in seminar — brief recap.

Key Idea: Instantaneous → Average Velocity

$$\text{Flow Matching:} \quad v(x_t, t) \quad \text{(instantaneous)} \tag{1}$$

$$\text{MeanFlow:} \quad u(z_t, r, t) = \frac{1}{t-r} \int_r^t v(z_s, s)\, ds \quad \text{(average)} \tag{2}$$

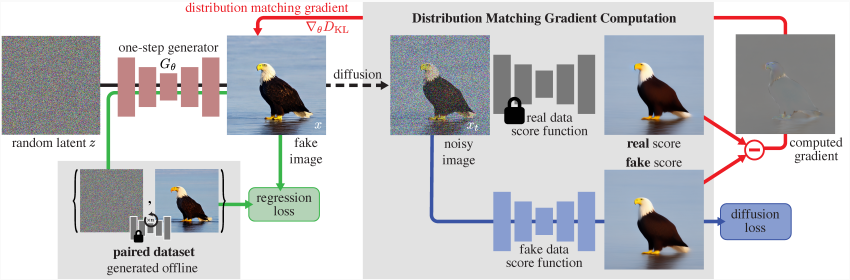
1-step generation: $x_1 = \epsilon + u_\theta(\epsilon, 0, 1)$
MeanFlow Identity: $v(z_t, t) = \partial_z u \cdot v + \partial_t u \quad \rightarrow$ trained via JVP

Limitations of CTM / MeanFlow

1. **Common: Additional constraints for special cases**
   - CTM: $G_\theta(x_t, t, t) = x_t$
   - MeanFlow: $u_\theta(x_t, t, t) = v(x_t, t)$
2. **CTM**
   - Training Stability: CTM has indirect training signal unlike MeanFlow.
   - Not self-contained: Require GAN component for performance.

Distribution Matching Distillation (DMD) [Yin et al., 2024] (CVPR 2024)

Key Idea: Directly minimize a KL-divergence between two distributions



The gradient of KL-divergence can be explicitly expressed with score functions.

$$\nabla_\theta \int_0^T w(t)\ D_{\mathrm{KL}}(q_t^{fake} \| p_t^{\mathrm{real}}) dt \approx \int_0^T w'(t)\ \mathbb{E}_{\hat{\mathbf{x}}_t}\left[\epsilon_{real}(\hat{\mathbf{x}}_t; t) - \epsilon_{fake}(\hat{\mathbf{x}}_t; t)\right] \frac{\partial \hat{\mathbf{x}}_t}{\partial \theta} dt$$

Limitations of DMD

1. **Not self-contained**
   - Require pretrained diffusion model.
   - One-step generator should be initialized reasonably.
2. **Heavy overhead before training**
   - Required to generate paired dataset (noise, image)

Summary of Existing One-Step Methods

Method	Objective type	ODE Dep.	Self-contained
Diffusion Models	Regression	–	✓
Consistency Models	Consistency constraint	✓	△
CTM	Consistency constraint	✓	△
MeanFlow	Regression (velocity)	✓	✓
DMD	Regression (KL-divergence)	✗	✗

Main: Drifting Models

Drifting Models [Deng et al., 2026]: Concept

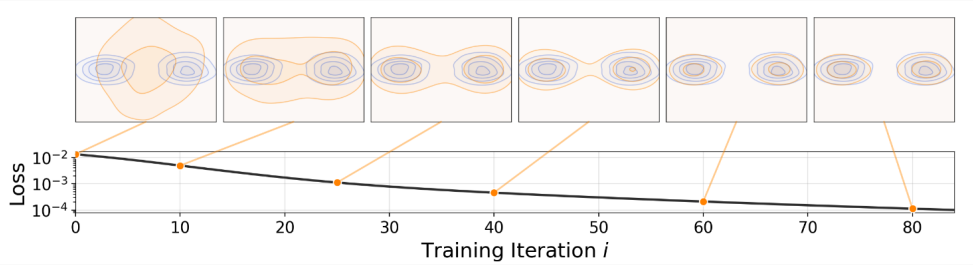
Diffusion and Flow Models

Distribution evolves during **inference** along pseudo-time t.

Drifting Models

Distribution evolves during **training** across iterations i.

Distribution evolution in drifting models



Drifting Field for Training

Drift

During training, SGD induces pushforward distributions $\{q_i\}$ with $q_i = [f_i]_{\#} p_\epsilon$. Authors define the drift as

$$\Delta x_i = f_{i+1}(\epsilon) - f_i(\epsilon),$$

which gives

$$x_{i+1} = x_i + \underbrace{\big(f_{i+1}(\epsilon) - f_i(\epsilon)\big)}_{\Delta x_i = \text{“drift”}}.$$

- Parameter update $\rightarrow$ output shifts slightly $\rightarrow$ this is the “drift”
- Key idea: govern this drift with an **explicit field** V

$\rightarrow$ *Origin of the name “Drifting”*

Drifting Field for Training

Drifting Field

$V_{p,q} : \mathbb{R}^d \to \mathbb{R}^d$ determines the direction and magnitude of sample movement:

$$x_{i+1} = x_i + V_{p,\,q_i}(x_i).$$

Anti-symmetry condition:

$$V_{p,q}(x) = -V_{q,p}(x), \quad \forall x.$$

Proposition 3.1

If V is anti-symmetric: $q = p \;\Rightarrow\; V_{p,q}(x) = 0,\ \forall x$

Proof:

$$q = p \;\Rightarrow\; V_{p,q} = V_{q,p} = -V_{p,q} \;\Rightarrow\; V_{p,q} = 0.$$

Intuition: When the two distributions match, samples stop drifting (equilibrium).
Converse: If $V_{p,q} = 0 \Rightarrow p = q$, train model to make $V_{p,q} = 0$.

Drifting Field for Training

Identifiability of the Zero-Drift (Appendix C.1.)

$V_{p,q} = 0 \Rightarrow p = q$ does not hold for general cases.

However, with the following assumptions:

- Kernel $k(x, y)$ is expressive enough.
- $V_{p,q}(x) = 0$ holds for all x

$\Rightarrow$ With some heuristics, $V_{p,q} = 0$ implies $p = q$.

Training Objective

Fixed-point relation at equilibrium:

$$f_{\hat{\theta}}(\epsilon) = f_{\hat{\theta}}(\epsilon) + V_{p,\, q_{\hat{\theta}}}\big(f_{\hat{\theta}}(\epsilon)\big)$$

Convert to a fixed-point iteration $\rightarrow$ implemented with **stop-gradient**:

Loss Function

$$\mathcal{L} = \mathbb{E}_{\epsilon}\left[\left\| \underbrace{f_\theta(\epsilon)}_{\text{prediction}} - \underbrace{\text{sg}\big(f_\theta(\epsilon) + V(f_\theta(\epsilon))\big)}_{\text{frozen target}} \right\|^2\right]$$

- Loss value $= \mathbb{E}\big[\|V\|^2\big] \;\rightarrow$ smaller V means closer to equilibrium
- V depends on $q_\theta \rightarrow$ stop-gradient bypasses backprop through the distribution

Distinction in Objective

Previous methods

Objective is fixed during training

$$L(\theta) = \mathbb{E}[\|\epsilon_\theta(x_t, t) - \epsilon\|^2]$$

Drifting Models

Objective depends on current model distribution

$$L(\theta; q_\theta) = \mathbb{E}\left[\|f_\theta(\epsilon) - \mathrm{sg}(f_\theta(\epsilon) + V_{p,q_\theta}(f_\theta(\epsilon)))\|^2\right]$$

Designing the Drifting Field

Drifting Field driven by Attraction and Repulsion

$$V_{p,q}(x) = \mathbb{E}_{y^+\sim p, y^-\sim q}\left[\mathcal{K}(x, y^+, y^-)\right] = \underbrace{V_p^+(x)}_{\text{attraction}} - \underbrace{V_q^-(x)}_{\text{repulsion}},$$

for

$$V_p^+(x) = \frac{1}{Z_p}\mathbb{E}_p\big[k(x, y^+)(y^+ - x)\big], \quad V_q^-(x) = \frac{1}{Z_q}\mathbb{E}_q\big[k(x, y^-)(y^- - x)\big],$$

and

$$Z_p(x) = \mathbb{E}_p[k(x, y^+)], \quad Z_q(x) = \mathbb{E}_q[k(x, y^-)].$$

Then, the drifting field is expressed as

$$V_{p,q} := \frac{1}{Z_p Z_q}\mathbb{E}_{p,q}[k(x, y^+)k(x, y^-)(y^+ - y^-)].$$

Designing the Drifting Field

Drifting Field driven by Attraction and Repulsion

$$V_{p,q}(x) := \frac{1}{Z_p Z_q} \mathbb{E}_{p,q}[k(x, y^+)k(x, y^-)(y^+ - y^-)].$$

Anti-symmetric: $V_{p,q} = -V_{q,p}$

Kernel

Authors define the kernel $k(\cdot, \cdot)$, a function that measures the similarity, as

$$k(x, y) = \exp\left(-\frac{1}{\tau}\|x - y\|\right).$$

Implementation: $\tilde{k}(x, y) = \text{softmax}_y\left(-\frac{1}{\tau}\|x - y\|\right)$ absorbs normalizing factor. Then, further apply extra softmax over x.

Remark: similar to InfoNCE

Drifting in Feature Space

Kernel fails to capture meaningful similarity on high-dimensional raw space.

$\Rightarrow$ Introduce a self-supervised pretrained feature extractor ϕ

Multi-scale Feature Space Loss

$$\mathcal{L} = \sum_j \mathbb{E}\left[\left\|\phi_j(x) - \mathrm{sg}\big(\phi_j(x) + V(\phi_j(x))\big)\right\|^2\right]$$

Implementation: Multi-scale features from ResNet-style image encoder, MoCo, SimCLR

ϕ is used **only at training time**, not needed at inference

Key limitation

Without a feature encoder, the method does not work on ImageNet.

Classifier-Free Guidance & Training Algorithm

CFG at training time

Positive samples: $y^+ \sim p_{\text{data}}(\cdot|c)$

Negative samples: $y_- \sim \tilde{q}(\cdot|c) = (1-\gamma)q_\theta(\cdot|c) + \gamma p_{\text{data}}(\cdot|\phi)$

Then, the training goal is

$$\tilde{q}(\cdot|c) = p_{\text{data}}(\cdot|c) \iff q_\theta(\cdot|c) = \alpha p_{\text{data}}(\cdot|c) - (\alpha-1)p_{\text{data}}(\cdot|\phi),$$

where CFG scale $\alpha = \frac{1}{1-\gamma}$ conditioned on the network $\rightarrow$ **freely adjustable at inference**.

Training Algorithm

Training Algorithm

Algorithm 1 Training Loss. Note: for brevity, here the negative samples `y_neg` are from the same batch of generated data, though they can include other source of negatives.

```
# f: generator
# y_pos: [N_pos, D], data samples

e = randn([N, C]) # noise
x = f(e) # [N, D], generated samples
y_neg = x # reuse x as negatives

V = compute_V(x, y_pos, y_neg)
x_drifted = stopgrad(x + V)

loss = mse_loss(x - x_drifted)
```

Implementation details

- Main experiment is performed on latent space
  "In our case, when using ϕ, the generator f can still produce outputs in the pixel space or the latent space of a tokenizer."
- Feature extractor: MoCo, SimCLR, MAE in latent space

Experiments: ImageNet (256x256) (SD-VAE latent space)

Necessity of Anti-symmetry

Table 1. **Importance of anti-symmetry:** breaking the anti-symmetry leads to failure. Here, the anti-symmetric case is defined in Eq. (10) and Eq. (11); other destructive cases are defined in similar ways. (Setting: B/2 model, 100 epochs)

case	drifting field $\mathbf{V}$	FID
anti-symmetry (default)	$\mathbf{V}^+ - \mathbf{V}^-$	**8.46**
1.5× attraction	$1.5\mathbf{V}^+ - \mathbf{V}^-$	41.05
1.5× repulsion	$\mathbf{V}^+ - 1.5\mathbf{V}^-$	46.28
2.0× attraction	$2\mathbf{V}^+ - \mathbf{V}^-$	86.16
2.0× repulsion	$\mathbf{V}^+ - 2\mathbf{V}^-$	112.84
attraction-only	$\mathbf{V}^+$	177.14

Larger N_{pos} and N_{neg} Improve

Table 2. **Allocation of positive and negative samples.** In both sub-tables, we control the total compute by fixing the epochs (100) and the batch size $B = N_c \times N_{\text{pos}}$ (4096). Here, N_c is for class labels. Under the same budget, increasing positive samples (**left**) and negative samples (**right**) improves generation quality. (Setting: B/2 model, 100 epochs)

N_c	N_{pos}	N_{neg}	B	FID
64	1	64	4096	20.43
64	16	64	4096	10.39
64	32	64	4096	8.97
64	**64**	64	4096	**8.46**

N_c	N_{pos}	N_{neg}	B	FID
512	8	8	4096	11.82
256	16	16	4096	10.16
128	32	32	4096	9.32
64	64	**64**	4096	**8.46**

Experiments: ImageNet (256x256) (SD-VAE latent space)

Importance of the Feature Extractor

Table 3. **Feature space for drifting.** We compare self-supervised learning (SSL) encoders. Standard SimCLR and MoCo encoders achieve competitive results, whereas our customized latent-MAE performs best and benefits from increased width and longer training. (Generator setting: B/2 model, 100 epochs)

	feature encoder (ϕ)				
SSL method	arch	block	width	SSL ep.	FID
SimCLR	ResNet	bottleneck	256	800	11.05
MoCo-v2	ResNet	bottleneck	256	800	8.41
latent-MAE (default)	ResNet	basic	256	192	8.46
latent-MAE	ResNet	basic	384	192	7.26
latent-MAE	ResNet	basic	512	192	6.49
latent-MAE	ResNet	basic	640	192	6.30
latent-MAE	ResNet	basic	640	1280	4.28
latent-MAE + cls ft	ResNet	basic	640	1280	**3.36**

Further Engineering

Table 4. **From ablation to final setting.** We train our model for more epochs, adjust hyper-parameters for this regime, and use a larger model size.

case	arch	ep	FID
(a) baseline (from Table 3)	B/2	100	3.36
(b) longer	B/2	320	2.51
(c) longer + hyper-param.	B/2	1280	1.75
(d) larger model	L/2	1280	**1.54**

Experiments: ImageNet (256x256) and Robotics Control

Latent space

method	space	params	NFE	FID↓	IS↑
Multi-step Diffusion/Flows					
DiT-XL/2 (Peebles & Xie, 2023)	SD-VAE	675M+49M	250×2	2.27	278.2
SiT-XL/2 (Ma et al., 2024)	SD-VAE	675M+49M	250×2	2.06	270.3
SiT-XL/2+REPA (Yu et al., 2024)	SD-VAE	675M+49M	250×2	1.42	305.7
LightningDiT-XL/2 (Yao et al., 2025)	VA-VAE	675M+70M	250×2	1.35	295.3
RAE+DiT^{DH}-XL/2 (Zheng et al., 2025)	RAE	839M+415M	50×2	**1.13**	262.6
Single-step Diffusion/Flows					
iCT-XL/2 (Song & Dhariwal, 2023)	SD-VAE	675M+49M	1	34.24	–
Shortcut-XL/2 (Frans et al., 2024)	SD-VAE	675M+49M	1	10.60	–
MeanFlow-XL/2 (Geng et al., 2025a)	SD-VAE	676M+49M	1	3.43	–
AdvFlow-XL/2 (Lin et al., 2025)	SD-VAE	673M+49M	1	2.38	284.2
iMeanFlow-XL/2 (Geng et al., 2025b)	SD-VAE	610M+49M	1	1.72	282.0
Drifting Models					
Drifting Model, B/2	SD-VAE	133M+49M	1	1.75	263.2
Drifting Model, L/2	SD-VAE	463M+49M	1	**1.54**	258.9

Pixel space

method	space	params	NFE	FID↓	IS↑
Multi-step Diffusion/Flows					
ADM-G (Dhariwal & Nichol, 2021)	pix	554M	250×2	4.59	186.7
SiD, UViT/2 (Hoogeboom et al., 2023)	pix	2.5B	1000×2	2.44	256.3
VDM++, UViT/2 (Kingma & Gao, 2023)	pix	2.5B	256×2	2.12	267.7
SiD2, UViT/2 (Hoogeboom et al., 2024)	pix	–	512×2	1.73	–
SiD2, UViT/1 (Hoogeboom et al., 2024)	pix	–	512×2	**1.38**	–
JiT-G/16 (Li & He, 2025)	pix	2B	100×2	1.82	292.6
PixelDiT/16 (Yu et al., 2025)	pix	797M	200×2	1.61	292.7
Single-step Diffusion/Flows					
EPG-L/16 (Lei et al., 2025)	pix	540M	1	8.82	–
GANs					
BigGAN (Brock et al., 2018)	pix	112M	1	6.95	152.8
GigaGAN (Kang et al., 2023)	pix	569M	1	3.45	225.5
StyleGAN-XL (Sauer et al., 2022)	pix	166M	1	2.30	265.1
Drifting Models					
Drifting Model, B/16	pix	134M	1	1.76	299.7
Drifting Model, L/16	pix	464M	1	**1.61**	307.5

Robotics Control

Task	Setting	Diffusion Policy NFE: 100	**Drifting Policy NFE: 1**
Single-Stage Tasks *(State & Visual Observation)*			
Lift	State	0.98	**1.00**
	Visual	**1.00**	**1.00**
Can	State	0.96	**0.98**
	Visual	0.97	**0.99**
ToolHang	State	0.30	**0.38**
	Visual	**0.73**	0.67
PushT	State	**0.91**	0.86
	Visual	0.84	**0.86**
Multi-Stage Tasks *(State Observation)*			
BlockPush	Phase 1	0.36	**0.56**
	Phase 2	0.11	**0.16**
Kitchen	Phase 1	**1.00**	**1.00**
	Phase 2	**1.00**	**1.00**
	Phase 3	**1.00**	0.99
	Phase 4	**0.99**	0.96

Discussion

Summary

1. **Training-time distribution evolution**
   Diffusion simulates dynamics at inference.
   Drifting instead evolves the generator distribution during training.

2. **Drift-based generative learning**
   Training updates induce a drift in generated samples.
   The model explicitly learns a drift field to align distributions.

3. **Kernel interaction between distributions**
   Attraction to data + repulsion from generated samples
   $\rightarrow$ distribution alignment through kernel mean-shift dynamics

Positioning of Drifting Among Existing Methods

Method	Objective type	ODE Dep.	Self-contained
Diffusion Models	Regression	–	✓
Consistency Models	Consistency constraint	✓	△
CTM	Consistency constraint	✓	△
MeanFlow	Regression (velocity)	✓	✓
DMD	Regression (KL-divergence)	✗	✗
Drifting Models	Regression (Drift)	✗	△

- While the objective type of drifting models is regression, the drifting field itself has a contrastive-loss-like form.

Limitations

1. **Theoretical gap**
   - $V \to 0 \Rightarrow p = q$ is **not guaranteed**
   - Only an identifiability heuristic is provided (Appendix C.1)

2. **Kernel and representation dependency**: $k(\phi(x), \phi(y))$
   - Difficult to design expressive kernel on image or video domain.
   - Performance depends on the feature representation used to compute the kernel similarity.

3. **Not guaranteed design optimality**
   - Kernel, drifting field, and architecture all have room for improvement.

4. **Scalability unverified**
   - Not tested beyond 256×256, or with text conditioning.
   - Requires a large batch size due to the contrastive-like objective design.

Reference

Deng, M., Li, H., Li, T., Du, Y., and He, K. (2026).
Generative modeling via drifting.
arXiv preprint arXiv:2602.04770.

Geng, Z., Deng, M., Bai, X., Kolter, J. Z., and He, K. (2025).
Mean flows for one-step generative modeling.
NeurIPS.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. (2024).
Consistency trajectory models: Learning probability flow ODE trajectory of diffusion.

In *The Twelfth International Conference on Learning Representations*.

Reference

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023).
Consistency models.
In *ICML*.

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. (2024).
One-step diffusion with distribution matching distillation.
In *CVPR*.