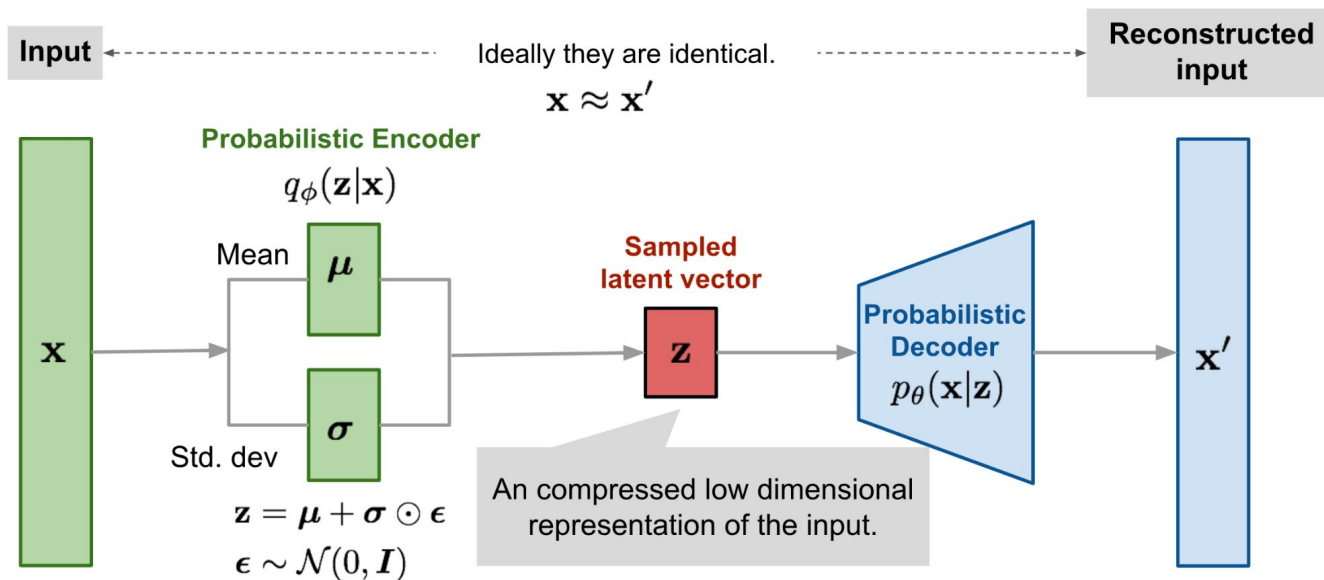


Speeding up sampling from generative models

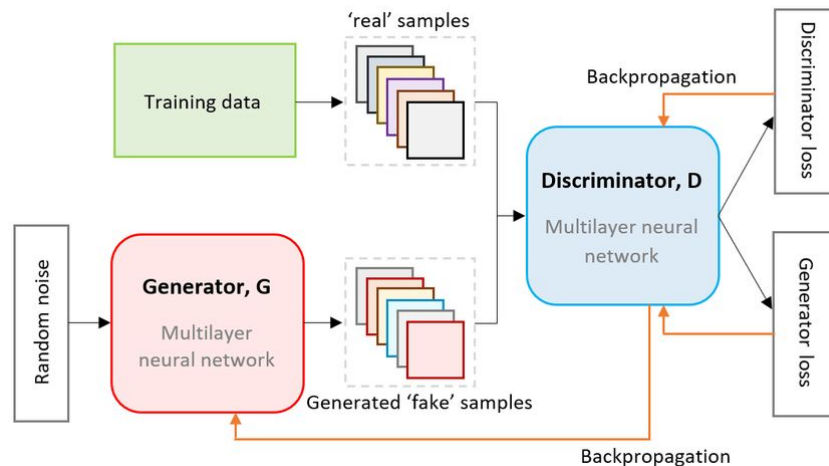
History of generative models: VAE (2013)

Optimizing ELBO $\ln p_{\theta}(x) \geq \mathbb{E}_{z \sim q_{\phi}(\cdot|x)} \left[\ln \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right].$



History of generative models: GAN (2014)

Discriminator tries to distinguish real images from fake while generator tries to generate images indistinguishable from real



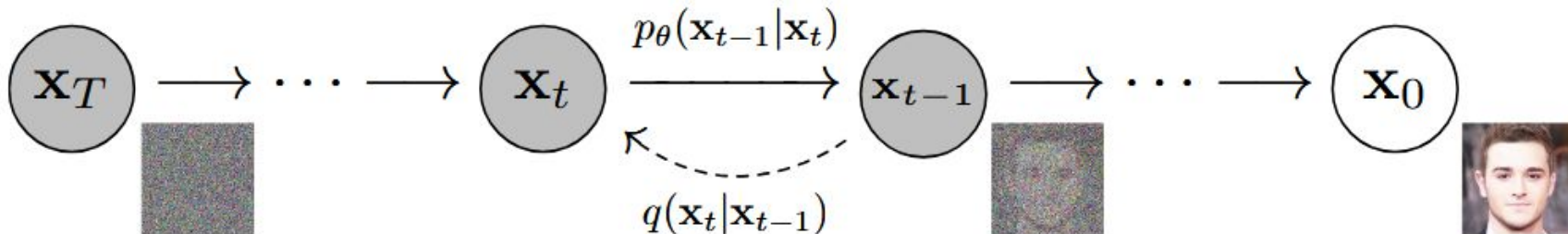
$$\begin{aligned}\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{x \sim p_g(x)} [\log(1 - D(x))]\end{aligned}$$

Diffusion models (2015, 2020)

Forward process progressively adds Gaussian noise to data
Reverse process learns to... reverse the forward process

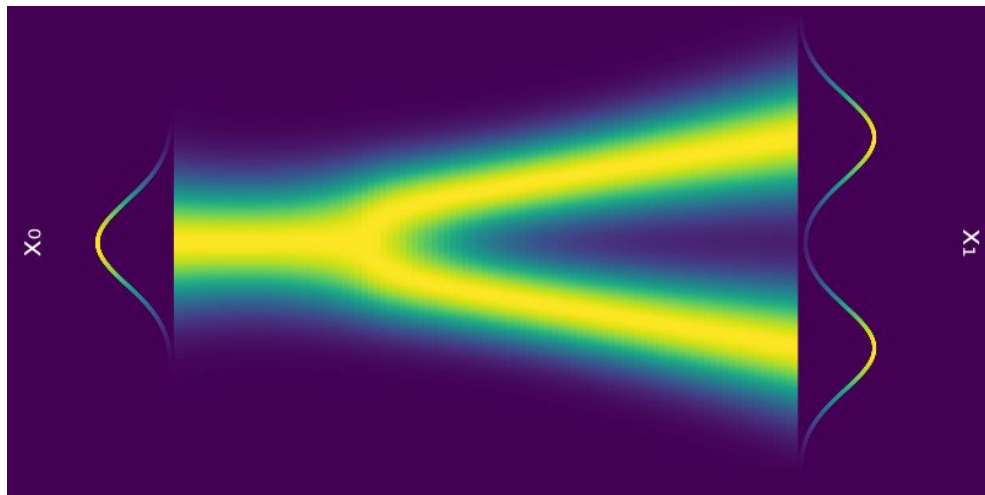
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$



Flow Matching (2022)

Learning a vector field that generates a flow that transforms noise to data



$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x))$$
$$\phi_0(x) = x$$

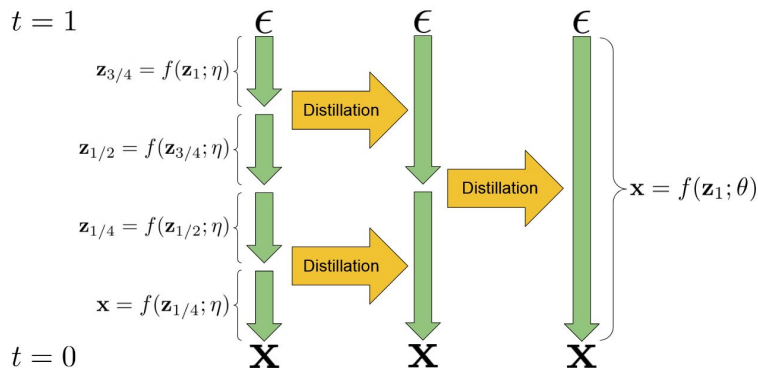
$\phi_t(X_0) = X_t$, where
 X_0 is pure noise
 X_1 is original data

Speed bottleneck

Method	Applicable Frameworks	Typical NFE (High Quality)
DDIM	Diffusion	100 - 250
DPM-Solver	Diffusion	10 - 20
DPM-Solver++	Diffusion / FM	15 - 20
UniPC	Diffusion / FM	10 - 20
RK4	FM	50 - 100

Progressive Distillation (2022)

Iteratively distill teacher into a student with double step size



Algorithm 1 Standard diffusion training

Require: Model $\hat{x}_\theta(z_t)$ to be trained

Require: Data set $\mathcal{D}$

Require: Loss weight function $w()$

while not converged **do**

$\mathbf{x} \sim \mathcal{D}$ $\triangleright$ Sample data

$t \sim U[0, 1]$ $\triangleright$ Sample time

$\epsilon \sim N(0, I)$ $\triangleright$ Sample noise

$\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$ $\triangleright$ Add noise to data

$\tilde{\mathbf{x}} = \mathbf{x}$ $\triangleright$ Clean data is target for $\hat{\mathbf{x}}$

$\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$ $\triangleright$ log-SNR

$L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$ $\triangleright$ Loss

$\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$ $\triangleright$ Optimization

end while

Algorithm 2 Progressive distillation

Require: Trained teacher model $\hat{\mathbf{x}}_\eta(\mathbf{z}_t)$

Require: Data set $\mathcal{D}$

Require: Loss weight function $w()$

Require: Student sampling steps N

for K iterations **do**

$\theta \leftarrow \eta$

$\triangleright$ Init student from teacher

while not converged **do**

$\mathbf{x} \sim \mathcal{D}$

$t = i/N, i \sim \text{Cat}[1, 2, \dots, N]$

$\epsilon \sim N(0, I)$

$\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$

2 steps of DDIM with teacher

$t' = t - 0.5/N, t'' = t - 1/N$

$\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t} (\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$

$\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}} (\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$

$\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t)\mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}$ $\triangleright$ Teacher $\hat{\mathbf{x}}$ target

$\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$

$L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$

$\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$

end while

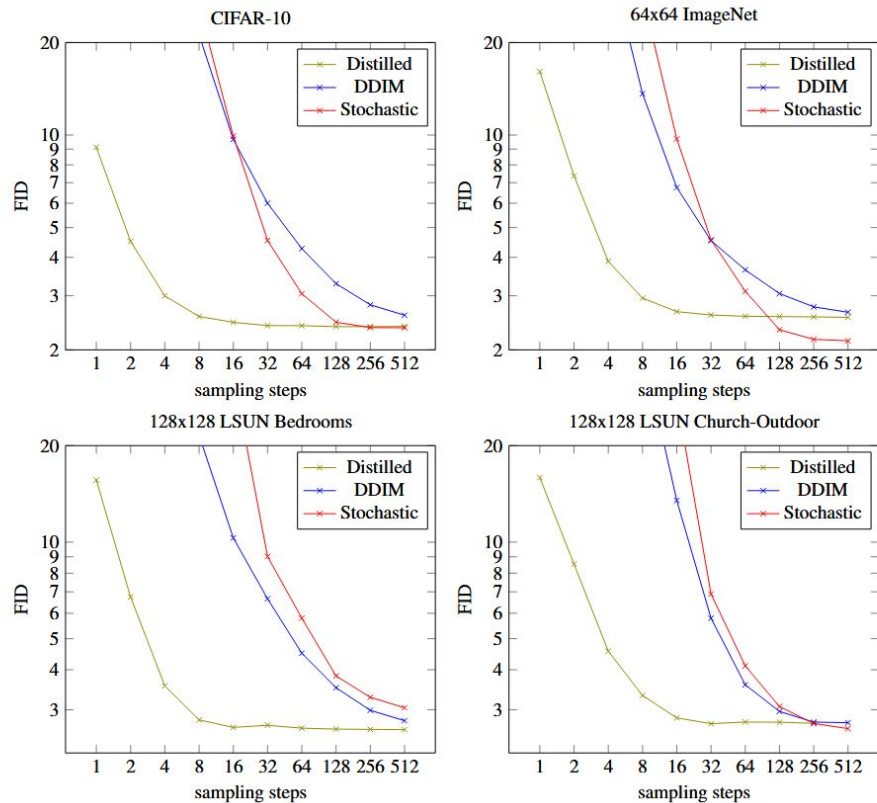
$\eta \leftarrow \theta$

$\triangleright$ Student becomes next teacher

$N \leftarrow N/2$ $\triangleright$ Halve number of sampling steps

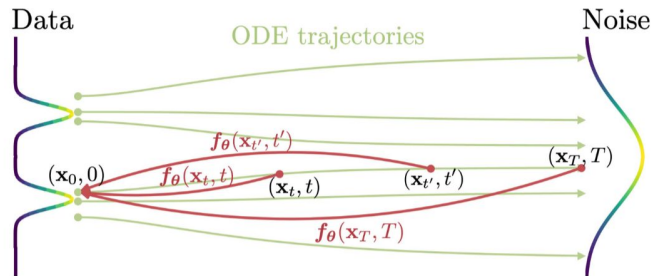
end for

Progressive Distillation (2022)



Consistency Models (2023)

Learn a function that predicts data from a point on a probability flow ODE using self-consistency loss



Algorithm 2 Consistency Distillation (CD)

Input: dataset $\mathcal{D}$, initial model parameter θ , learning rate η , ODE solver $\Phi(\cdot, \cdot; \phi)$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ
 $\theta^- \leftarrow \theta$

repeat

Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[[1, N - 1]]$

Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \mathbf{I})$

$\hat{\mathbf{x}}_{t_n}^{\phi} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \phi)$

$\mathcal{L}(\theta, \theta^-; \phi) \leftarrow$

$\lambda(t_n)d(\mathbf{f}_{\theta}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \mathbf{f}_{\theta^-}(\hat{\mathbf{x}}_{t_n}^{\phi}, t_n))$

$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(\theta, \theta^-; \phi)$

$\theta^- \leftarrow \text{stopgrad}(\mu \theta^- + (1 - \mu)\theta)$

until convergence

Algorithm 3 Consistency Training (CT)

Input: dataset $\mathcal{D}$, initial model parameter θ , learning rate η , step schedule $N(\cdot)$, EMA decay rate schedule $\mu(\cdot)$, $d(\cdot, \cdot)$, and $\lambda(\cdot)$

$\theta^- \leftarrow \theta$ and $k \leftarrow 0$

repeat

Sample $\mathbf{x} \sim \mathcal{D}$, and $n \sim \mathcal{U}[[1, N(k) - 1]]$

Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathcal{L}(\theta, \theta^-) \leftarrow$

$\lambda(t_n)d(\mathbf{f}_{\theta}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \mathbf{f}_{\theta^-}(\mathbf{x} + t_n\mathbf{z}, t_n))$

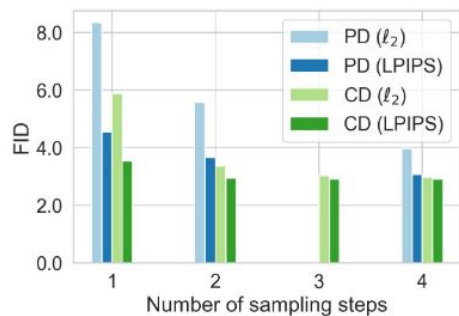
$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(\theta, \theta^-)$

$\theta^- \leftarrow \text{stopgrad}(\mu(k)\theta^- + (1 - \mu(k))\theta)$

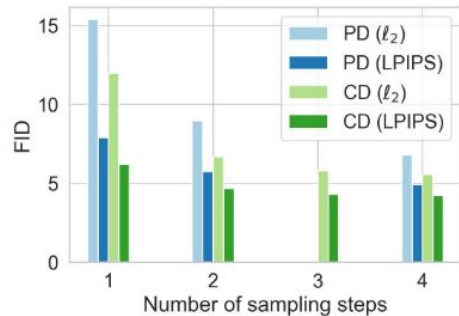
$k \leftarrow k + 1$

until convergence

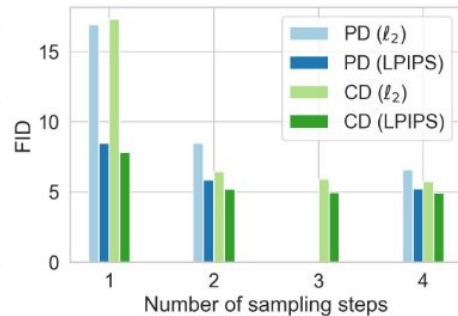
Consistency Models (2023)



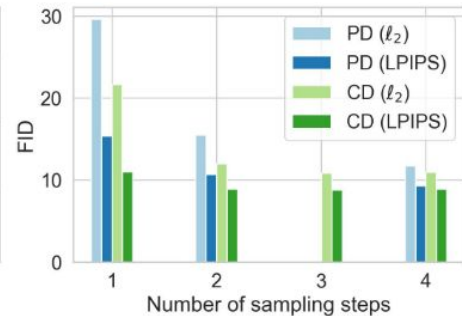
(a) CIFAR-10



(b) ImageNet 64 $\times$ 64



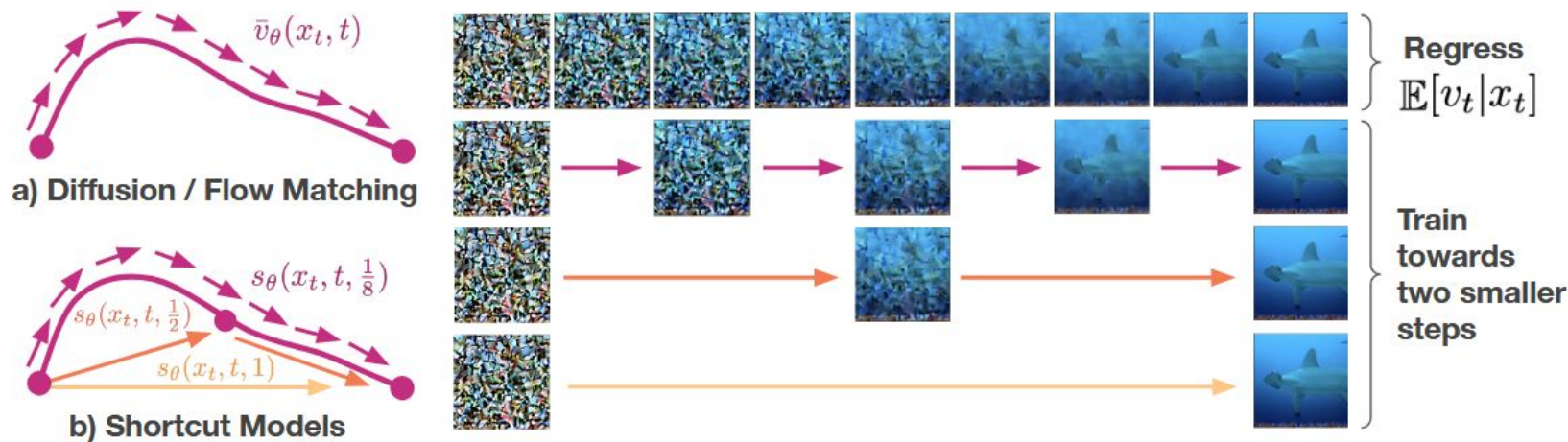
(c) Bedroom 256 $\times$ 256



(d) Cat 256 $\times$ 256

Shortcut Models (2024, K. Frans, S. Levine, UC Berkeley)

Condition model on the step size and learn self consistency



Loss and Algorithm

$$\mathcal{L}^S(\theta) = E_{x_0 \sim \mathcal{N}, x_1 \sim D, (t,d) \sim p(t,d)} \left[\underbrace{\|s_\theta(x_t, t, 0) - (x_1 - x_0)\|^2}_{\text{Flow-Matching}} + \underbrace{\|s_\theta(x_t, t, 2d) - s_{\text{target}}\|^2}_{\text{Self-Consistency}} \right],$$

where $s_{\text{target}} = s_\theta(x_t, t, d)/2 + s_\theta(x'_{t+d}, t, d)/2$ and $x'_{t+d} = x_t + s_\theta(x_t, t, d)d$.

Algorithm 1 Shortcut Model Training

while not converged **do**

$x_0 \sim \mathcal{N}(0, I)$, $x_1 \sim D$, $(d, t) \sim p(d, t)$

$x_t \leftarrow (1 - t)x_0 + tx_1$ Noise data point

for first k batch elements **do**

$s_{\text{target}} \leftarrow x_1 - x_0$

Flow-matching target

$d \leftarrow 0$

for other batch elements **do**

$s_t \leftarrow s_\theta(x_t, t, d)$

First small step

$x_{t+d} \leftarrow x_t + s_t d$

Follow ODE

$s_{t+d} \leftarrow s_\theta(x_{t+d}, t + d, d)$

Second small step

$s_{\text{target}} \leftarrow \text{stopgrad}(s_t + s_{t+d})/2$

Self-consistency target

$\theta \leftarrow \nabla_\theta \|s_\theta(x_t, t, 2d) - s_{\text{target}}\|^2$

Algorithm 2 Sampling

$x \sim \mathcal{N}(0, I)$

$d \leftarrow 1/M$

$t \leftarrow 0$

for $n \in [0, \dots, M - 1]$ **do**

$x \leftarrow x + s_\theta(x, t, d) d$

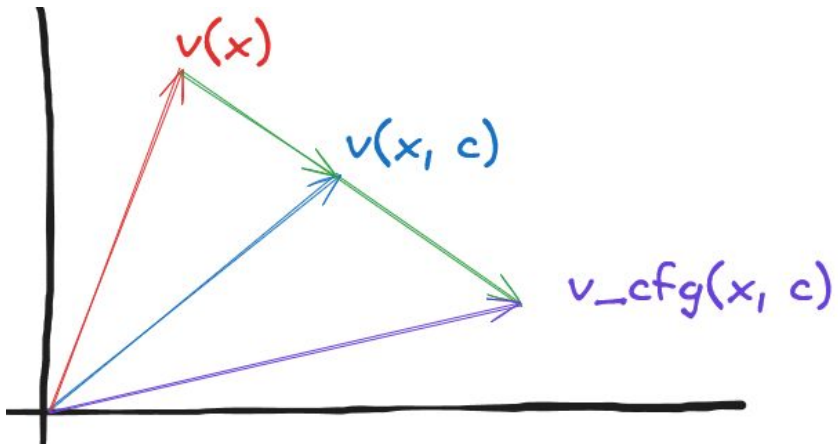
$t \leftarrow t + d$

return x

Classifier free guidance

$$v_{cfg}(x|c) = v(x|\emptyset) + \omega(v(x|c) - v(x|\emptyset))$$

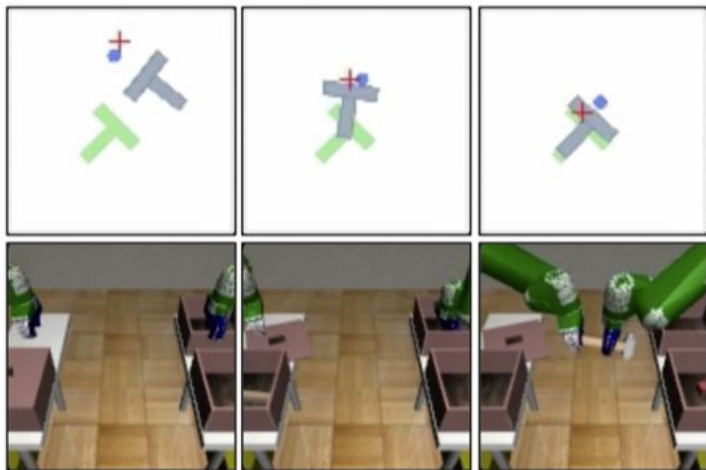
Authors only use CFG during training at $d = 0$



Evaluation

	CelebAHQ-256 (unconditioned)			Imagenet-256 (class conditioned)		
	128-Step	4-Step	1-Step	128-Step	4-Step	1-Step
Two phase training						
Progressive Distillation	(302.9)	(251.3)	14.8	(201.9)	(142.5)	35.6
Consistency Distillation	59.5	39.6	38.2	132.8	98.01	136.5
Reflow	16.1	18.4	23.2	16.9	32.8	44.8
End-to-end (single training run)						
Diffusion	23.0	(123.4)	(132.2)	39.7	(464.5)	(467.2)
Flow Matching	7.3	(63.3)	(280.5)	17.3	(108.2)	(324.8)
Consistency Training	53.7	19.0	33.2	42.8	43.0	69.7
Live Reflow (ours)	6.3	27.2	43.3	46.3	95.8	58.1
Shortcut Models (ours)	6.9	13.8	20.5	15.5	28.3	40.3

Evaluation (Robotics edition)



	# Steps	Push-T	Transport
IBC	100	0.90	0.00
Diffusion Policy	100	0.95	1.00
LSTM-GMM	1	0.67	0.76
BET	1	0.79	0.38
Diffusion Policy	1	0.12	0.00
Shortcut Policy (ours)	1	0.87	0.80

Still a gap between one-step and multi-step generation

Improved Shortcut Models (2025, Qualcomm AI Research)

Problems in shortcut models:

- flaw of compounding guidance
- inflexible fixed guidance
- frequency bias
- divergent self-consistency from EMA
- curvy flow trajectories



Flaw of compounding guidance

$$g_{\theta}^w(x_t, t, c, d) = s_{\theta}(x_t, t, \emptyset, d) + w(s_{\theta}(x_t, t, c, d) - s_{\theta}(x_t, t, \emptyset, d))$$

Proposition 1. *The model's prediction for a single large shortcut step of size $Nd = 1$ approximately equals the average of the guided displacements corresponding to the N smallest steps, but with an exponentially compounded guidance scale:*

$$s_{\theta}(x_0, 0, c, Nd) \approx \frac{1}{N} \sum_{i=0}^{N-1} g_{\theta}^{w^{\log_2(N)}} \left(x'_{\frac{i}{N}}, \frac{i}{N}, c, d \right). \quad (6)$$

Proof. See Appendix A. □

Intrinsic Guidance Training

Make guidance scale a model's input

$$s_{\theta}(\cdot, c, d = 0, w) \approx s_{\theta}(\cdot, c, d = 0, w = 0) + w \cdot \underbrace{(s_{\theta}(\cdot, c, d = 0, w = 0) - s_{\theta}(\cdot, \emptyset, d = 0, w = 0))}_{\mathbf{s}_{\text{guidance}}},$$

$$\mathcal{L}_{\text{velocity}}(\theta) := \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D, t \sim p(t)} \left[\|s_{\theta}(\mathbf{x}_t, t, c, d = 0, w = 0) - \mathbf{s}_{\text{velocity}}\|^2 \right],$$

$$\text{where } \mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1 \quad \text{and} \quad \mathbf{s}_{\text{velocity}} = \mathbf{x}_1 - \mathbf{x}_0.$$

$$\mathcal{L}_{\text{guidance}}(\theta) := \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D, (t, w) \sim p(t, w)} \left[\|s_{\theta}(\mathbf{x}_t, t, c, d = 0, w) - (\mathbf{s}_{\text{velocity}} + w \cdot \text{sg}(\mathbf{s}_{\text{guidance}}))\|^2 \right]$$

$$\mathcal{L}_{\text{consistency}}(\theta) := \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D, (t, w, d) \sim p(t, w, d)} \left[\|s_{\theta}(x_t, t, c, 2d, w) - s_{\text{consistency}}\|^2 \right],$$

$$\text{where } s_{\text{consistency}} := s_{\theta-}(x_t, t, c, d, w)/2 + s_{\theta-}(x'_{t+d}, t, c, d, w)/2$$

$$\text{and } x'_{t+d} = x_t + s_{\theta}(x_t, t, c, d, w)d,$$

Interval guidance

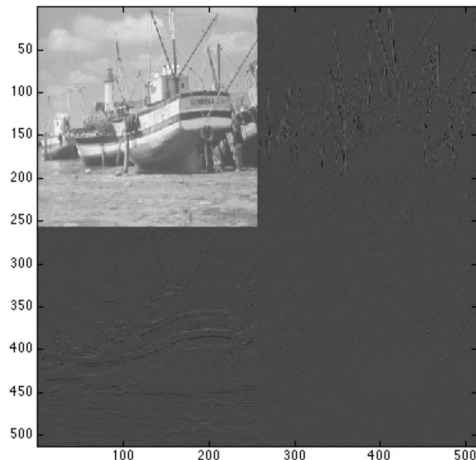
CFG at small timesteps can lead to premature collapse towards a few dominant modes

$$w(t) = \begin{cases} w, & \text{if } t \in [t_{\text{interval}}, 1) \\ 0, & \text{otherwise,} \end{cases}$$

Multi-Level Wavelet Function Against Frequency Bias

Deep neural networks often prioritize learning low-frequency information over high-frequency details, a phenomenon known as frequency bias

Solution: DWT



Algorithm 1 Multi-level Wavelet Function

```
class MultiLevelWaveletLoss:
    def __init__(self):
        self.dwt = DWT_2D("haar") # Discrete Wavelet Transform (DWT)
        self.diff_func = MSELoss() # Distance function

    def concatenated_dwt(self, x):
        xll, xlh, xhl, xhh = self.dwt(x) # Decompose into 4 wavelet sub-bands using DWT
        details = torch.cat([xll, xlh, xhl, xhh], dim=1) # Concatenate the sub-bands
        return details

    def __call__(self, pred, target, num_levels):
        total_loss = diff_func(pred, target).mean() # Calculate low-level loss in original output space

        # Recursively calculate loss on wavelet sub-bands for each level
        pred_curr, target_curr = pred, target
        for current_level in range(num_levels):
            # Derive predicted and target sub-bands using outputs from previous level
            pred_bands = self.concatenated_dwt(pred_curr)
            target_bands = self.concatenated_dwt(target_curr)

            # Calculate loss on current level
            total_loss += diff_func(pred_bands, target_bands).mean()
            pred_curr, target_curr = pred_bands, target_bands

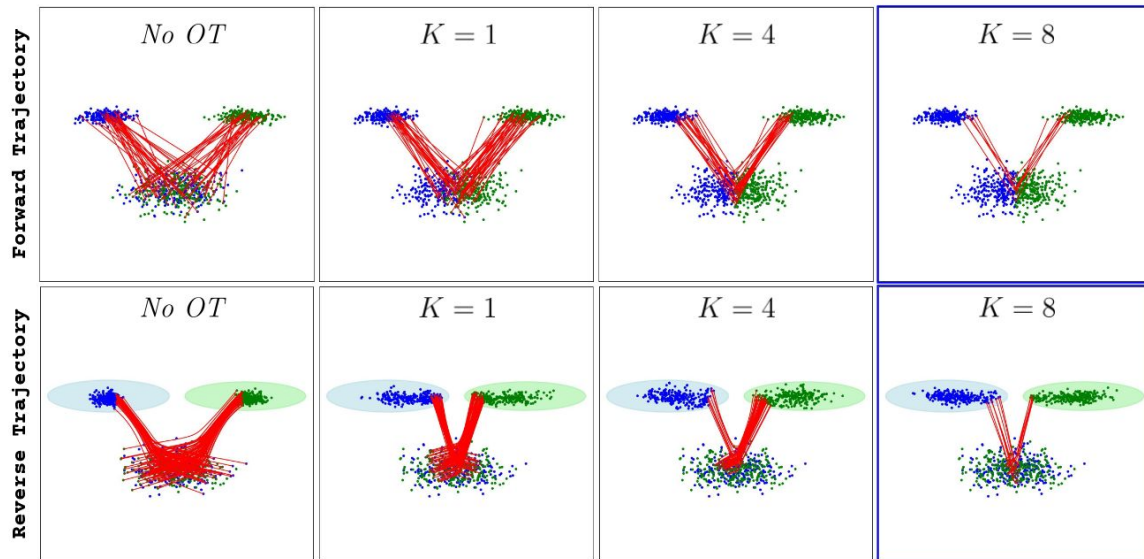
        # Taking average from all levels
        loss = total_loss / (num_levels + 1)
        return loss
```

Scaling Optimal Transport

Assign noise-data pairing deterministically.

You want bigger batch size for that to work better, solve OT for every K batches in advance

$$\min_{\pi \in S(N)} \sum_{i=0}^{N-1} \|x_0^i - x_1^{\pi(i)}\|$$



Twin EMA

Remember this? ->

$$\mathcal{L}_{\text{consistency}}(\theta) := \mathbb{E}_{\substack{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D \\ (t, w, d) \sim p(t, w, d)}} \left[\|\mathbf{s}_{\theta}(x_t, t, c, 2d, w) - s_{\text{consistency}}\|^2 \right],$$

$$\text{where } s_{\text{consistency}} := s_{\theta-}(x_t, t, c, d, w)/2 + s_{\theta-}(x'_{t+d}, t, c, d, w)/2$$

$$\text{and } x'_{t+d} = x_t + s_{\theta}(x_t, t, c, d, w)d,$$

Dilemma:

- No EMA -> oscillating loss

- Fast decay -> worse samples during inference

- Slow decay -> outdated network on backward

Solution:

- 2 EMAs!

Evaluation

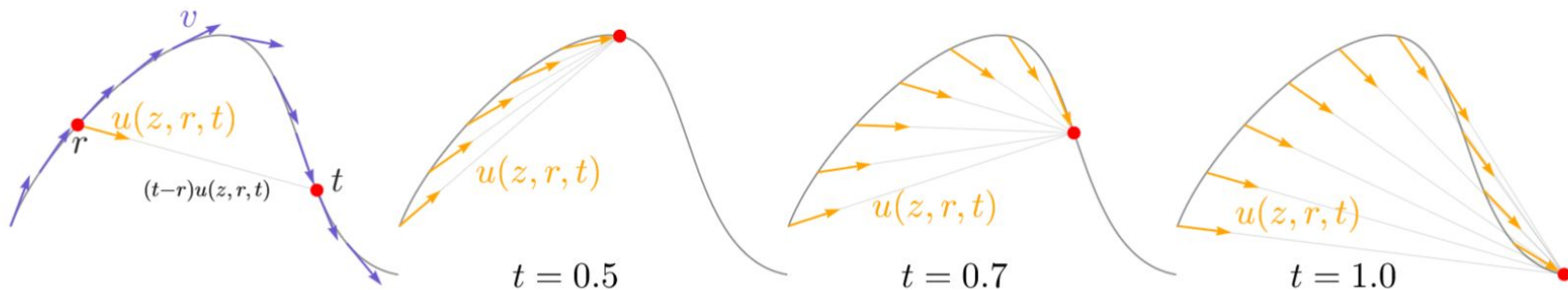
Model	FID-50k ($\downarrow$)	NFE ($\downarrow$)	#Params
GAN			
BigGAN-deep [5]	4.06	1	112M
GigaGAN [28]	3.45	1	569M
StyleGAN-XL [30]	2.30	1	166M
Masked & AR			
VQGAN [18]	26.52	1024	227M
MaskGIT [6]	6.18	8	227M
VAR- d_{20} [60]	2.57	10	600M
VAR- d_{30} [60]	1.92	10	2B
MAR [37]	1.98	100	400M
Diffusion & Flow Models			
ADM [14]	10.94	250	554M
CDM [25]	4.88	8100	-
SimDiff [27]	2.77	512	2B
LDM-4-G [52]	3.60	250	400M
U-DiT-L [61]	3.37	250	916M
U-ViT-H [2]	2.29	50	501M
DiT-XL/2 [48]	2.27	250	675M
SiT-XL/2 [43]	2.15	250	675M
REPA-XL/2 [71]	1.42	250	675M
FlowDCN [64]	2.00	250	618M

One-to-Many Step Models			
iCT [58]	34.24	1	675M
	20.3	2	675M
SM [20]	10.60	1	675M
	7.80	4	675M
IMM [73]	3.80	128	675M
	7.77	1	675M
	3.99	2	675M
	2.51	4	675M
	1.99	8	675M
iSM (ours)	5.27	1	675M
	2.44	2	675M
	2.05	4	675M
	1.93	8	675M
	1.88	128	675M

Method	FID _{N=1} ↓	FID _{N=4} ↓
Shortcut Models [20]	21.38	13.46
<i>Improved Shortcut Models (iSM)</i>		
+ Intrinsic Guidance	9.62	3.17
+ Interval Guidance in Training	8.49	2.81
+ Multi-level Wavelet Function	8.12	2.64
+ Scaling Optimal Transport	7.97	2.23
+ Twin EMA	6.56	2.16

Mean Flows (2025, Kaiming He, CMU/MIT)

Learn average velocity instead of the instantaneous



Derivation of the loss function

Average velocity $u(z_t, r, t) \triangleq \frac{1}{t-r} \int_r^t v(z_\tau, \tau) d\tau.$

$$\frac{d}{dt}(t-r)u(z_t, r, t) = \frac{d}{dt} \int_r^t v(z_\tau, \tau) d\tau \implies u(z_t, r, t) + (t-r) \frac{d}{dt} u(z_t, r, t) = v(z_t, t)$$

Mean Flow Identity
$$\underbrace{u(z_t, r, t)}_{\text{average vel.}} = \underbrace{v(z_t, t)}_{\text{instant. vel.}} - \underbrace{(t-r) \frac{d}{dt} u(z_t, r, t)}_{\text{time derivative}}$$

Partial derivative expansion
$$\frac{d}{dt} u(z_t, r, t) = v(z_t, t) \partial_z u + \partial_t u$$

$$\mathcal{L}(\theta) = \mathbb{E} \|u_\theta(z_t, r, t) - \text{sg}(u_{\text{tgt}})\|_2^2,$$

where
$$u_{\text{tgt}} = v(z_t, t) - (t-r) (v(z_t, t) \partial_z u_\theta + \partial_t u_\theta)$$

Algorithm

Utilizes Jacobian Vector Product function to compute time derivative of u

How to sample r, t for training:

75% of batch size $r = t$

25% of batch size $r, t \sim \text{lognorm}(-0.4, 1.0)$

Algorithm 1 MeanFlow: Training.

Note: in PyTorch and JAX, `jvp` returns the function output and JVP.

```
# fn(z, r, t): function to predict u
# x: training batch

t, r = sample_t_r()
e = randn_like(x)

z = (1 - t) * x + t * e
v = e - x

u, dudt = jvp(fn, (z, r, t), (v, 0, 1))

u_tgt = v - (t - r) * dudt
error = u - stopgrad(u_tgt)

loss = metric(error)
```

Algorithm 2 MeanFlow: 1-step Sampling

```
e = randn(x_shape)
x = e - fn(e, r=0, t=1)
```

Classifier Free Guidance

No CFG = bad

Default CFG = x2 NFE

Solution: add CFG directly to the loss function

$$v^{\text{cfg}}(z_t, t \mid \mathbf{c}) \triangleq \omega v(z_t, t \mid \mathbf{c}) + (1 - \omega) v(z_t, t)$$

$$u^{\text{cfg}}(z_t, r, t \mid \mathbf{c}) = v^{\text{cfg}}(z_t, t \mid \mathbf{c}) - (t - r) \frac{d}{dt} u^{\text{cfg}}(z_t, r, t \mid \mathbf{c})$$

$$v^{\text{cfg}}(z_t, t \mid \mathbf{c}) = \omega v(z_t, t \mid \mathbf{c}) + (1 - \omega) u^{\text{cfg}}(z_t, t, t)$$

$$\mathcal{L}(\theta) = \mathbb{E} \left\| u_{\theta}^{\text{cfg}}(z_t, r, t \mid \mathbf{c}) - \text{sg}(u_{\text{tgt}}) \right\|_2^2,$$

$$\text{where } u_{\text{tgt}} = \tilde{v}_t - (t - r) (\tilde{v}_t \partial_z u_{\theta}^{\text{cfg}} + \partial_t u_{\theta}^{\text{cfg}}).$$

Evaluation

method	params	NFE	FID
<i>1-NFE diffusion/flow from scratch</i>			
iCT-XL/2 [43] [†]	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	3.43
<i>2-NFE diffusion/flow from scratch</i>			
iCT-XL/2 [43] [†]	675M	2	20.30
iMM-XL/2 [52]	675M	1×2	7.77
MeanFlow-XL/2	676M	2	2.93
MeanFlow-XL/2+	676M	2	2.20

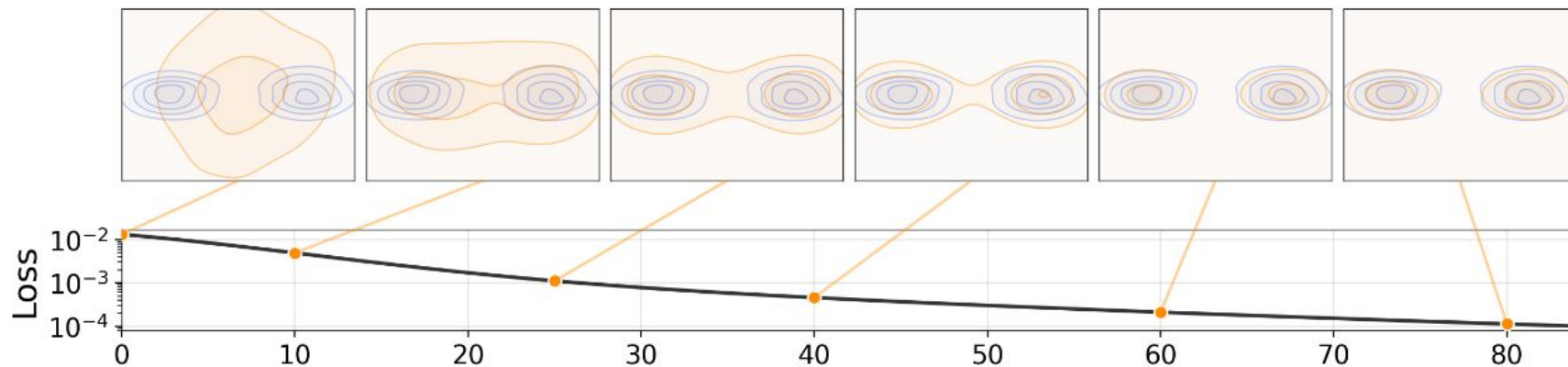
method	params	NFE	FID
<i>GANs</i>			
BigGAN [5]	112M	1	6.95
GigaGAN [21]	569M	1	3.45
StyleGAN-XL [40]	166M	1	2.30
<i>autoregressive/masking</i>			
AR w/ VQGAN [10]	227M	1024	26.52
MaskGIT [6]	227M	8	6.18
VAR- <i>d</i> 30 [47]	2B	10×2	1.92
MAR-H [27]	943M	256×2	1.55
<i>diffusion/flow</i>			
ADM [8]	554M	250×2	10.94
LDM-4-G [37]	400M	250×2	3.60
SimDiff [20]	2B	512×2	2.77
DiT-XL/2 [34]	675M	250×2	2.27
SiT-XL/2 [33]	675M	250×2	2.06
SiT-XL/2+REPA [51]	675M	250×2	1.42

There are a lot of papers!

- Reflow (2022)
- Improved Consistency Training (2023)
- Inductive Moment Matching (2025)
- AlphaFlow (2025)
- Adversarial Flow (2025)
- Improved Mean Flow (2025)

Drifting models (2026, Kaiming He, MIT/Harvard)

Design a process of transforming one distribution into another using a drifting field that evolves over time



Drifting field and training objective

Update rule: $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{V}_{p,q_i}(\mathbf{x}_i)$ where V should drift x from p to q

Idea: If V is antisymmetric then the fixed point of this equation is $p = q$

$$\mathbf{V}_{p,q}(\mathbf{x}) = -\mathbf{V}_{q,p}(\mathbf{x})$$

Loss based on this update rule

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

Designing the drifting field

$$\mathbf{V}_{p,q}(\mathbf{x}) = \mathbb{E}_{\mathbf{y}^+ \sim p} \mathbb{E}_{\mathbf{y}^- \sim q} [\mathcal{K}(x, \mathbf{y}^+, \mathbf{y}^-)]$$

$$\mathbf{V}_p^+(\mathbf{x}) := \frac{1}{Z_p} \mathbb{E}_p [k(\mathbf{x}, \mathbf{y}^+) (\mathbf{y}^+ - \mathbf{x})] \quad k(\mathbf{x}, \mathbf{y}) = \exp \left(-\frac{1}{\tau} \|\mathbf{x} - \mathbf{y}\| \right)$$

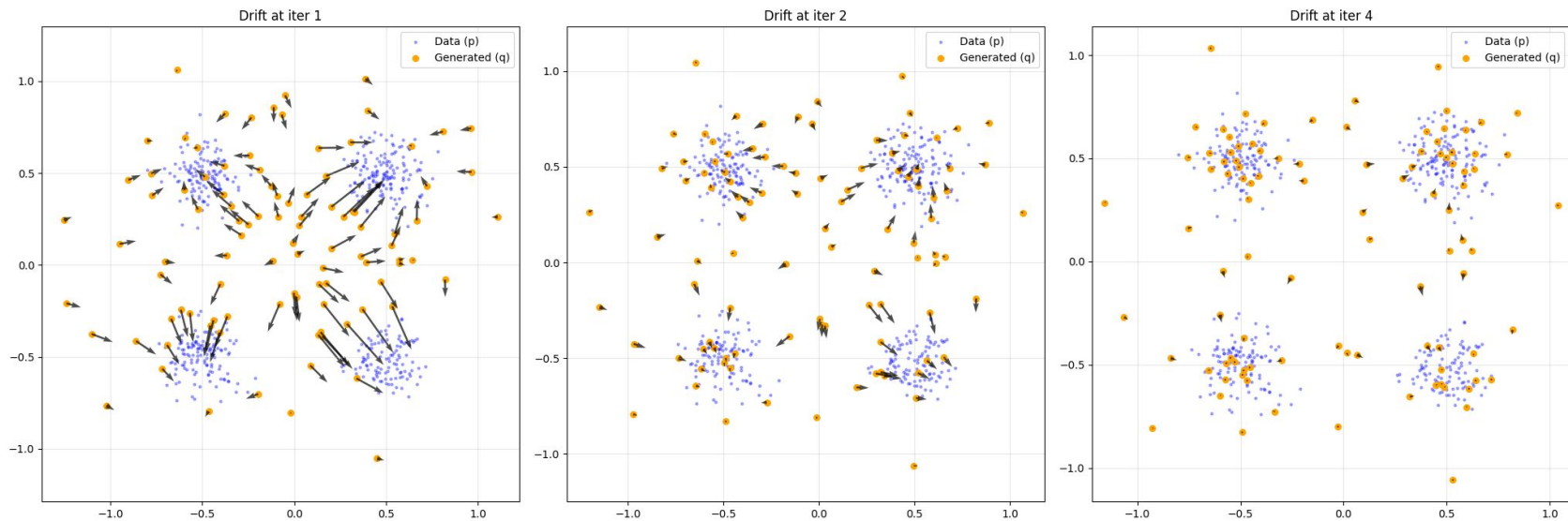
$$\mathbf{V}_q^-(\mathbf{x}) := \frac{1}{Z_q} \mathbb{E}_q [k(\mathbf{x}, \mathbf{y}^-) (\mathbf{y}^- - \mathbf{x})]$$

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

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

Visualization

https://colab.research.google.com/github/lambertae/lambertae.github.io/blob/main/projects/drifting/notebooks/drifting_model_demo.ipynb



Drifting in feature space

Since we want the kernel function to measure similarity and keep it simple, we have to measure that similarity in a feature space of an encoder model

$$\mathbb{E} \left[\left\| \phi(\mathbf{x}) - \text{stopgrad} \left(\phi(\mathbf{x}) + \mathbf{V}(\phi(\mathbf{x})) \right) \right\|^2 \right]$$

“...we report that we were unable to make our method work on ImageNet without a feature encoder. In this case, the kernel may fail to effectively describe similarity, even in the presence of a latent VAE.”

Classifier Free Guidance

Alternative approach to CFG: sample negative samples from a mixture of generated data and real data with different class label

$$\tilde{q}(\cdot|c) \triangleq (1 - \gamma) q_{\theta}(\cdot|c) + \gamma p_{\text{data}}(\cdot|\emptyset)$$

Training objective: $\tilde{q}(\cdot|c) \approx p_{\text{data}}(\cdot|c)$

$$q_{\theta}(\cdot|c) = \alpha p_{\text{data}}(\cdot|c) - (\alpha - 1) p_{\text{data}}(\cdot|\emptyset)$$

$$\alpha = \frac{1}{1-\gamma} \geq 1$$

Evaluation

method	space	params	NFE	FID↓	IS↑
<i>Multi-step Diffusion/Flows</i>					
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
<i>Single-step Diffusion/Flows</i>					
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
<i>Drifting Models</i>					
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

method	space	params	NFE	FID↓	IS↑
<i>Multi-step Diffusion/Flows</i>					
ADM-G (Dhariwal & Nichol, 2021)	pix	554M	250×2	4.59	186.7
SiD, UViT/2 (Hoogetboom 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 (Hoogetboom et al., 2024)	pix	—	512×2	1.73	—
SiD2, UViT/1 (Hoogetboom 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
<i>Single-step Diffusion/Flows</i>					
EPG-L/16 (Lei et al., 2025)	pix	540M	1	8.82	—
<i>GANs</i>					
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
<i>Drifting Models</i>					
Drifting Model, B/16	pix	134M	1	1.76	299.7
Drifting Model, L/16	pix	464M	1	1.61	307.5

Evaluation

SSL method	feature encoder (ϕ)				FID
	arch	block	width	SSL ep.	
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

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

		Diffusion Policy	Drifting Policy
Task	Setting	NFE: 100	NFE: 1
<i>Single-Stage Tasks (State & Visual Observation)</i>			
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
<i>Multi-Stage Tasks (State Observation)</i>			
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

Thank you for your attention!