Generative Modeling via Drifting

Deng et al. 2026

Reviewed & Presented by Joon Hyeok Kim

Key Takeaways

1. (Claimed by the Authors) New paradigm of training

- Pros
  - 1-NFE evaluation with competitive quality
  - Simple Loss Construction
- Cons
  - Relies on batch normalization (Info NCE)
    - which also means that there's a room for improvement?

2. (Suggested) New possibilities for Normalizing Flow?

Concept 1) Push Forward

Def.) Given the distributions p and q, and a model f,

we say q is a pushforward distribution of p under f

if $f(\epsilon) = \mathbf{x} \sim q$ for $\epsilon \sim p$.

We denote $q = f_{\#}p$.

Then, we want to find f_θ s.t. $p_{\text{data}} = f_{\theta\#}p_{\text{prior}}$

Now, denote $\epsilon \sim p_\epsilon$, and assume training the model f_θ.

After some iterative steps, we may be at the i-th step.

Then, we may have a sequence of pushforward distributions $\{f_i\}$

where $\mathbf{x}_i = f_i(\epsilon) \sim q_i$

How can we define a relation between f_is?

Concept 2) Drift & Drifting Field

Now consider $\mathbf{x}_{i+1} = f_{i+1}(\epsilon) \sim q_{i+1}$ and $\mathbf{x}_i = f_i(\epsilon) \sim q_i$.

We may define the drift as $\Delta\mathbf{x}_{i+1} = \mathbf{x}_{i+1} - \mathbf{x}_i$.

Further, we may define a drifting field of $\mathbf{V}_{p,q}$ as

$$\mathbf{V}_{p,q_i} : \mathbb{R}^d \to \mathbb{R}^d \text{ s.t. } \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{V}_{p,q_i}(\mathbf{x}_i)$$

where $p = p_{\text{data}}$ and $\mathbf{x}_i = f_i(\epsilon) \sim q_i$

Then what would be the convergence condition? (i.e., no more drift)

Obviously, $\mathbf{V}_{p,q} = \mathbf{0}$.

Among many candidates for $\mathbf{V}_{p,q}$, authors suggest the anti-symmetric drifting field that satisfies

$$\mathbf{V}_{p,q}(\mathbf{x}) = -\mathbf{V}_{q,p}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^d$$

Why?)

If $p = q$ ($\Leftrightarrow p_{\text{data}} = p_{\text{model}}$: what we want!),

then $\mathbf{V}_{p,q}(\mathbf{x}) = \mathbf{V}_{q,p}(\mathbf{x}) \Rightarrow \mathbf{V}_{p,q}(\mathbf{x}) = 0$

Authors further show that their Softmax method satisfies the converse,

i.e. $\mathbf{V}_{p,q}(\mathbf{x}) = 0 \Rightarrow p = q$ (Refer to Appendix C.1 for more details)

Assuming that there is a method that satisfies "$\mathbf{V}_{p,q}(\mathbf{x}) = 0 \Rightarrow p = q$", optimizing the following objective is equivalent to training f to pushforward p to q.

Training Objective

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

Only if we have $\mathbf{V}$ that satisfies $\mathbf{V}_{p,q}(\mathbf{x}) = 0 \Rightarrow p = q$

Kernel & Info NCE (Softmax)

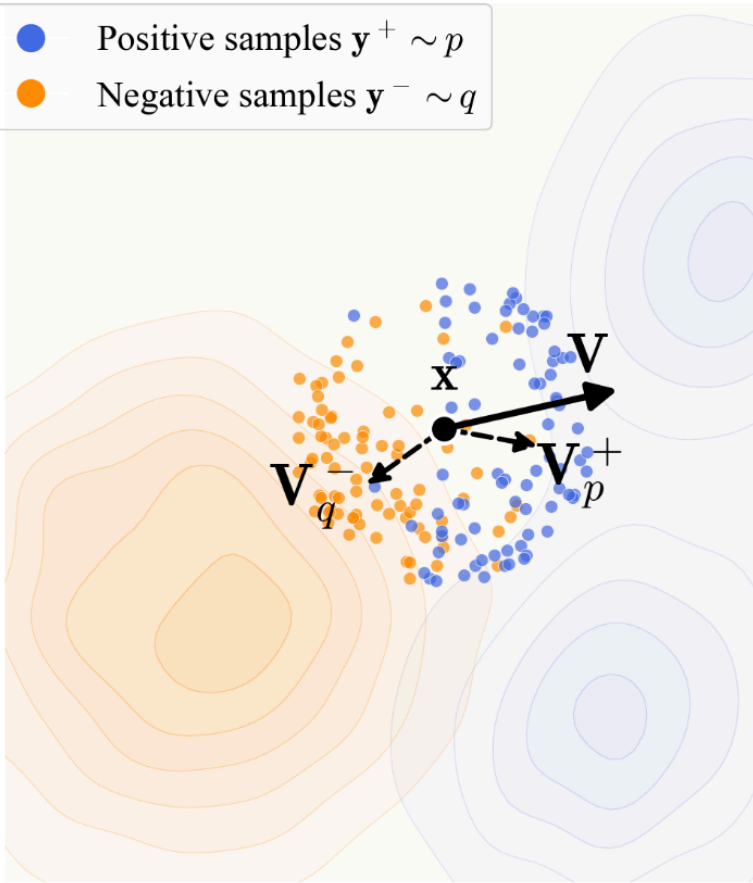
We want to find $\mathbf{V}$ s.t. $\mathbf{V}_{p,q}(\mathbf{x}) = 0 \Rightarrow p = q$

Use a kernel function $\mathbf{k}(\cdot, \cdot)$ to get the positive/negative drift field of

$$\begin{cases} \mathbf{V}_p^+(\mathbf{x}) := \dfrac{\mathbb{E}_p\left[\mathbf{k}(\mathbf{x}, \mathbf{y}^+)(\mathbf{y}^+ - \mathbf{x})\right]}{Z_p} \\ \mathbf{V}_q^-(\mathbf{x}) := \dfrac{\mathbb{E}_q\left[\mathbf{k}(\mathbf{x}, \mathbf{y}^-)(\mathbf{y}^- - \mathbf{x})\right]}{Z_q} \end{cases} \quad \text{for} \quad \begin{cases} Z_p(\mathbf{x}) := \mathbb{E}_p[\mathbf{k}(\mathbf{x}, \mathbf{y}^+)] \\ Z_q(\mathbf{x}) := \mathbb{E}_q[\mathbf{k}(\mathbf{x}, \mathbf{y}^-)] \end{cases}$$

Then adding two drift fields, we may get

$$\begin{aligned}\mathbf{V}_{p,q}(\mathbf{x}) &= \mathbf{V}_p^+(\mathbf{x}) - \mathbf{V}_q^-(\mathbf{x}) \\ &= \frac{1}{Z_p Z_q}\mathbb{E}_{p,q}\left[\mathbf{k}(\mathbf{x},\mathbf{y}^+)\mathbf{k}(\mathbf{x},\mathbf{y}^-)(\mathbf{y}^+ - \mathbf{y}^-)\right]\end{aligned}$$



Following the Info NCE, the above $\mathbf{k}(\mathbf{x},\mathbf{y})$ is implemented using the Softmax function where the logit is given by $-\frac{1}{\tau}\left\|\mathbf{x}-\mathbf{y}\right\|$

cf.) $\mathcal{L} = -\mathbb{E}\left[\log \frac{\sum_{\mathbf{y}^+\in\mathcal{Y}^+} f(\mathbf{y}^+,\mathbf{x})}{\sum_{\mathbf{y}\in\mathcal{Y}^+\cup\mathcal{Y}^-} f(\mathbf{y},\mathbf{x})}\right]$

Refer to Oord et al. 2019, "*Representation Learning with Contrastive Predictive Coding*" for more details.

Implementation Details

- Backbone : $f = \mathrm{DiT}$

- Masked Autoencoder (MAE) is used for the Imagenet dataset

  (Successful on the Pixel-space generation as well.)

- CFG is used : $p_{\mathrm{data}}(\cdot \mid \emptyset)$ is added to the negative sample pool

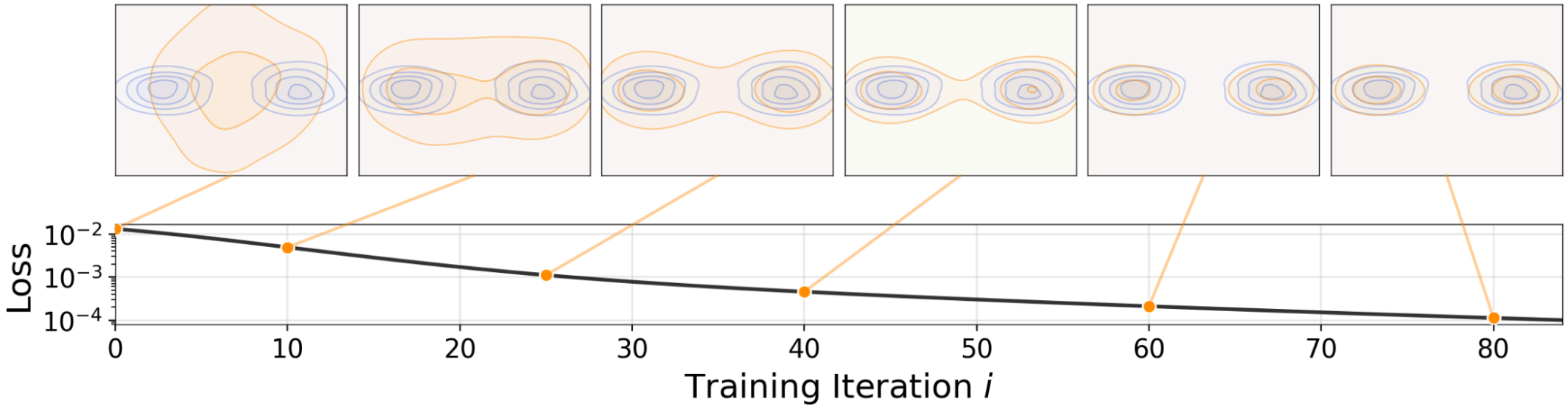
[Training Pipeline]

$$\epsilon \sim \mathcal{N} \rightarrow \mathrm{MAE}(f(\epsilon)) = \mathbf{y}^{-}$$
$$\mathbf{x} \sim p_{\mathrm{data}} \rightarrow \mathrm{MAE}(\mathbf{x}) = \mathbf{y}^{+}$$
$$\rightarrow \ \mathrm{Softmax}\left(\frac{-\|\mathbf{x}-\mathbf{y}\|}{\tau}\right)_{\mathbf{y}=\mathbf{y}^{-},\mathbf{y}^{+}} \approx \mathcal{K}(\mathbf{x}, \mathbf{y}^{-}, \mathbf{y}^{+})$$
$$\rightarrow \mathbf{V}_{p,q} \rightarrow \mathcal{L} = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\left\| \underbrace{f_\theta(\boldsymbol{\epsilon})}_{\text{prediction}} - \underbrace{\mathrm{stop\_grad}(f_\theta(\boldsymbol{\epsilon}) + \mathbf{V}_{p,q}(f_\theta(\boldsymbol{\epsilon})))}_{\text{frozen target}} \right\|^2\right]$$

Experiments

2D Toy Example : Bimodal Distribution

- Result
  - Successful and robust to mode collapse
    - Why?) Each mode attracts and push samples to each other

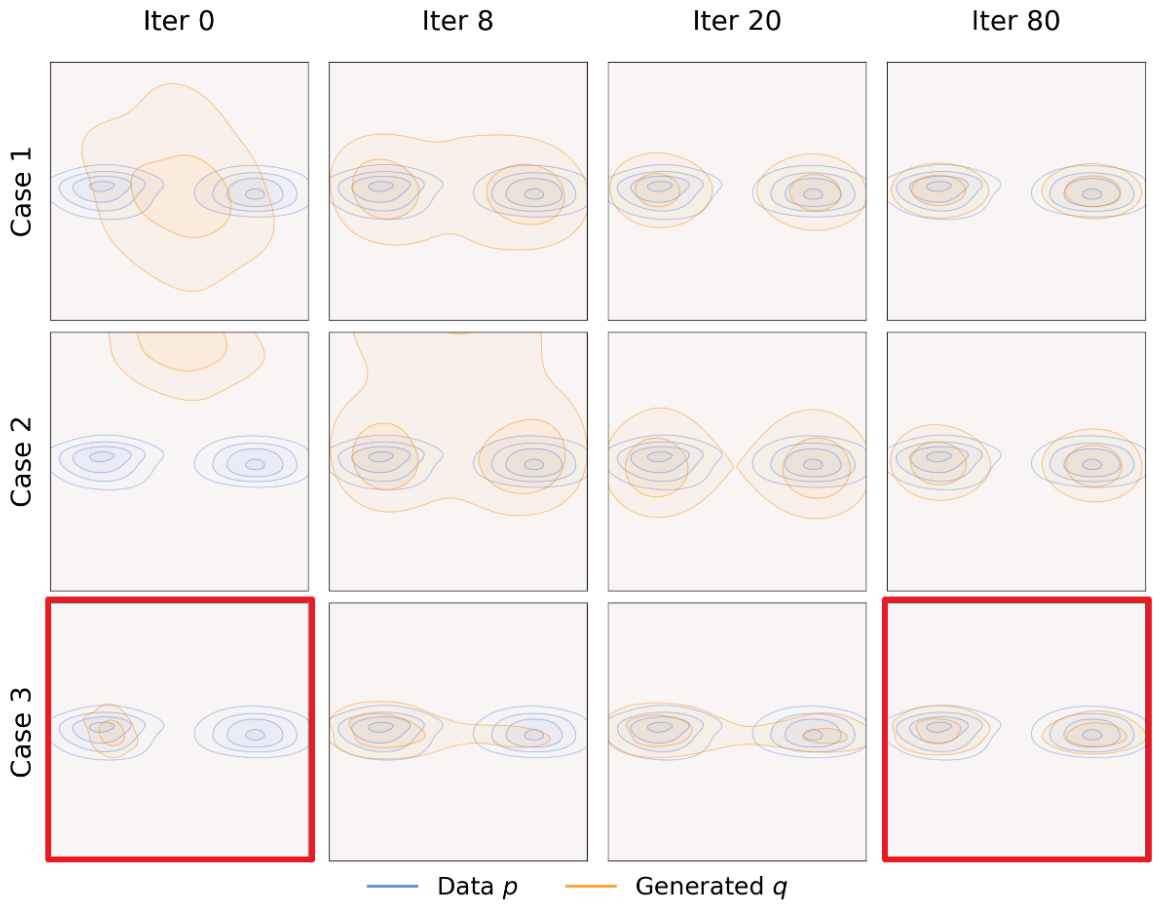


Robustness to Mode Collapse

Case 3

Model distribution q initialized as the unimodal and mode collapsed form.

In iter 80, q successfully generalized the binomal data distribution.



Experiment) ImageNet 256×256 in Latent Space

Authors used Masked Autoencoder (MAE, pre-trained & ResNet-based) to compute loss on the feature (latent) space

$$\mathrm{NFE} = 1 \quad (\because f(\epsilon) \sim p_{\mathrm{data}})$$

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$^{\mathrm{DH}}$-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

Experiment) ImageNet 256×256 in 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

Did anti-symmetric $\mathbf{V}_{p,q}$ work?

If you recall, authors used $\mathbf{V}_{p,q}(\mathbf{x}) = \mathbf{V}_p^+(\mathbf{x}) - \mathbf{V}_q^-(\mathbf{x})$

to satisfy $\mathbf{V}_{p,q}(\mathbf{x}) = 0 \Rightarrow p = q$

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

Experiment) Robotics in $\mathrm{NFE} = 1$?

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

Possible Direction : NF x Drifting Loss

What if we replace DiT with TarFlow (NF), get the exact log-likelihood $\log p_\theta$ from NF, and use the exact score $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ as the $\mathbf{V}_q^-$?

Original Drift

$$\mathbf{V}_{p,q}(\mathbf{x}) = \mathbf{V}_p^+(\mathbf{x}) - \mathbf{V}_q^-(\mathbf{x}) = \mathcal{K}(\mathbf{x}, \mathbf{y}^-, \mathbf{y}^+) \approx \mathrm{Softmax}\left(\frac{-\|\mathbf{x}-\mathbf{y}\|}{\tau}\right)_{\mathbf{y}=\mathbf{y}^-,\mathbf{y}^+}$$

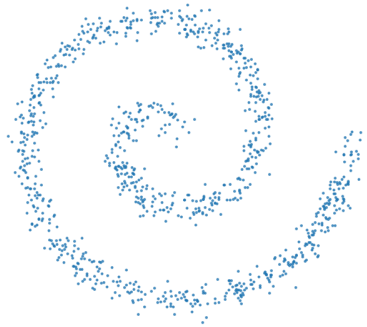
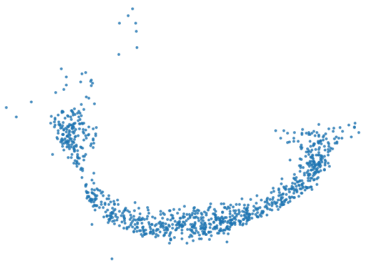
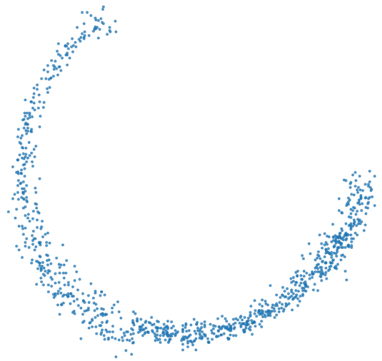
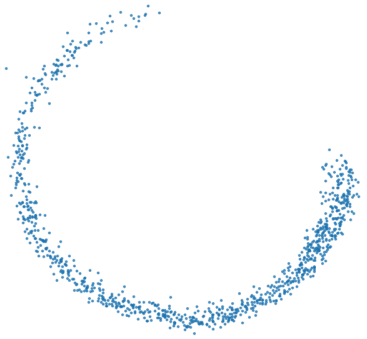
Drifting NF

$$\mathbf{V}_{p,q}(\mathbf{x}) = \mathbf{V}_p^+(\mathbf{x}) - \mathbf{V}_q^-(\mathbf{x}) = \mathrm{Softmax}\left(\frac{-\|\mathbf{x}-\mathbf{y}^+\|}{\tau}\right) - \lambda_{\mathrm{NF}} \cdot \underbrace{\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})}_{\text{vs } \mathbf{y}^- \sim p_\theta}$$

[WIP] Toy Experiment 1 : Without Even Explicitly Training NF

$$\mathcal{L} = \mathcal{L}_{\text{drift}}$$

$$\text{s.t. } \mathcal{L}_{\text{drift}} = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\left\| f_\theta(\boldsymbol{\epsilon}) - \text{sg}\left(f_\theta(\boldsymbol{\epsilon}) + \text{Softmax}\left(\frac{-\|f_\theta(\boldsymbol{\epsilon}) - \mathbf{y}^+\|}{\tau}\right) - \nabla_{\mathbf{x}} \log p(f_\theta(\boldsymbol{\epsilon}))\right)\right\|^2\right]$$

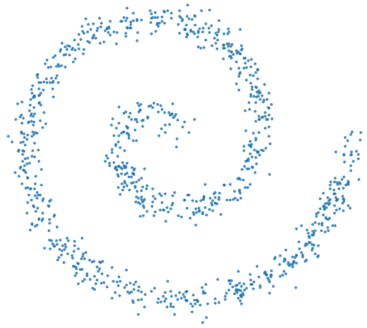
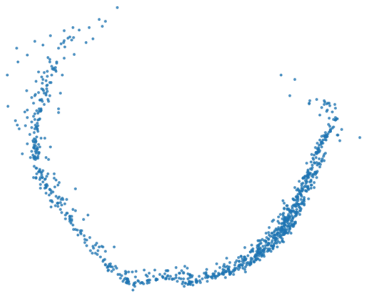
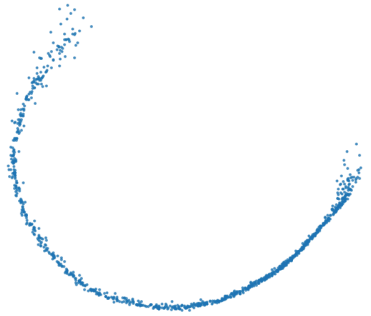
Training Data	$i = 600$	$i = 2,000$	$i = 16,000$
			

Possible Solutions : Temperature Annealing

[WIP] Toy Experiment 2 : Simultaneously Training NF and Drift

$$\mathcal{L} = \mathcal{L}_{\text{drift}} + \lambda \cdot \mathcal{L}_{\text{MLE\_NF}}$$

$$\text{where } \mathcal{L}_{\text{MLE\_NF}} = \log p(f(x)) - \log \left| \det \frac{\partial f(x)}{\partial x} \right|$$

Training Data	$(\lambda = 2)\ i = 600$	$i = 2,000$	$i = 2,700$ (still running...)
			

Problem : Slower training

Questions or thoughts

Thank you.