Flow-based generative models

– From Normalizing flow to Flow Matching

Chen Xu[1]

February 20, 2024

[1] School of Industrial and Systems Engineering,
Georgia Institute of Technology

ML Reading group, Toyota Reserch Institute

Outline

- Background and formulation
- Normalizing flow
- Flow Matching
- Summary and open questions

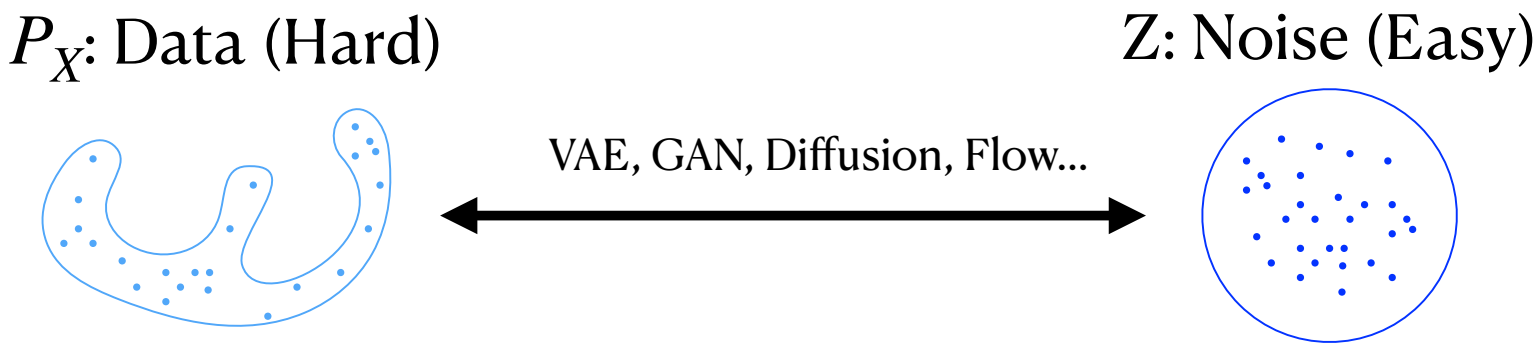
Desired takeaway: learn two types of flow models, which have the potential to outperform diffusion models.

Background

- Problem: given data $X \sim P_X$, we learn P_X by
  (1) sampling from P_X as in generative modeling.
  (2) understanding its properties (e.g., estimate $\log p_X(x)$).

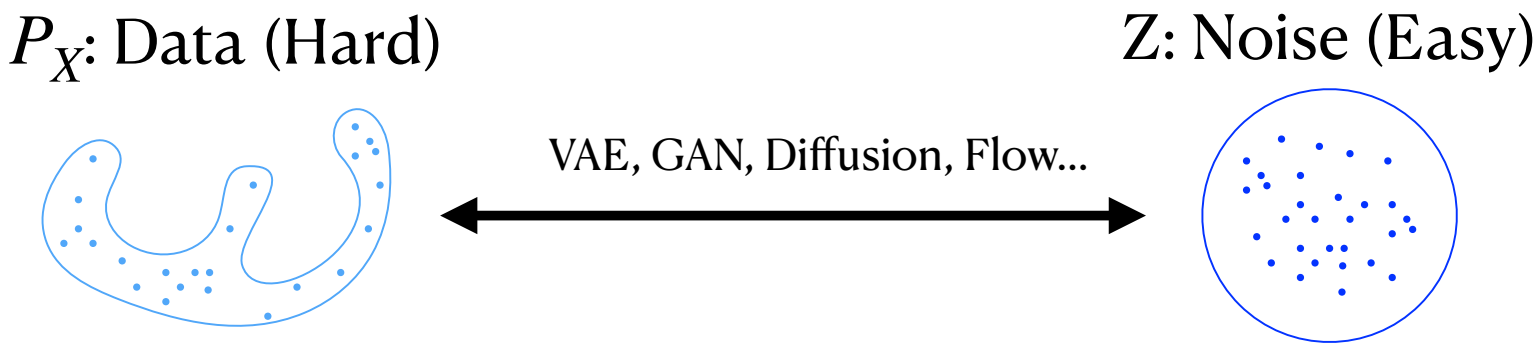
Background

- Problem: given data $X \sim P_X$, we learn P_X by
  (1) sampling from P_X as in generative modeling.
  (2) understanding its properties (e.g., estimate $\log p_X(x)$).
- The problem is not new:



Background

- Problem: given data $X \sim P_X$, we learn P_X by
  (1) sampling from P_X as in generative modeling.
  (2) understanding its properties (e.g., estimate $\log p_X(x)$).
- The problem is not new:



- The focus of flow models is to learn a *continuous and invertible* transformation between the "hard" (original P_X) and "easy" (standard Gaussian) distributions.

Mathematical formulation

- Flow-based models can be represented as an *ordinary differential equation* over a distribution pair (P, Q). Given $x(0) = X \sim P$ and $x(1) = Y \sim Q$, consider the time-dependent random variable $x(t), t \in [0, 1]$:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

Mathematical formulation

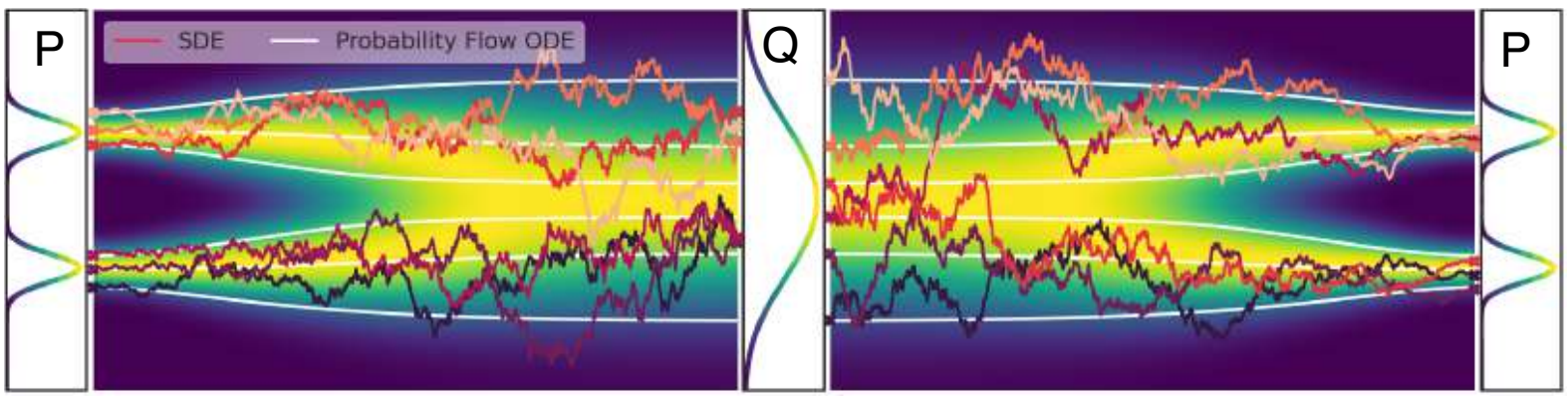
- Flow-based models can be represented as an *ordinary differential equation* over a distribution pair (P, Q). Given $x(0) = X \sim P$ and $x(1) = Y \sim Q$, consider the time-dependent random variable $x(t), t \in [0, 1]$:

$$dx(t)/dt = f(x(t), t), \tag{1}$$

$$\rightarrow x(t) = x(0) + \int_0^t f(x(s), s)ds. \tag{2}$$

- Unlike the *stochastic* differential equation below (used in diffusion models), (1) is deterministic given $x(0)$ or $x(1)$.

$$dx(t) = \tilde{f}(x(t), t)dt + g(t)dw \qquad \text{(Ito's SDE)}$$

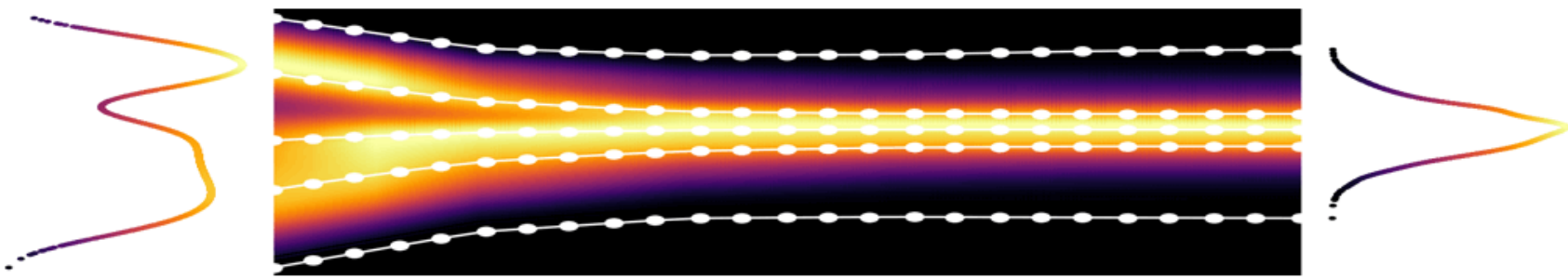


Mathematical formulation

- The central problem is to learn the velocity field $dx(t)/dt = f(x(t), t)$ via a neural network $f(x(t), t; \theta)$.

Mathematical formulation

- The central problem is to learn the velocity field $dx(t)/dt = f(x(t), t)$ via a neural network $f(x(t), t; \theta)$.
- Once $f(x(t), t; \theta)$ is learnt,
  - (Forward $\rightarrow$, starting at $x(0) \sim P$) Perform density estimation of P upon integrating f_θ over $t \in [0, 1]$.
  - (Backward $\leftarrow$, starting at $x(1) \sim Q$) Perform generation of new samples from P upon integrating f_θ over $t \in [1, 0]$.



Normalizing flow: formulation

- In the case of $Q = Z$ for $Z \sim \mathcal{N}(0, I)$, this is recognized as a *normalizing* flow [Chen et al., 2018, Grathwohl et al., 2019].

Normalizing flow: formulation

- In the case of $Q = Z$ for $Z \sim \mathcal{N}(0, I)$, this is recognized as a *normalizing* flow [Chen et al., 2018, Grathwohl et al., 2019].
- Given a measurable function T defined on $x \sim P$, consider the push-forward measure $T_{\#}P$ where $(T_{\#}P)(\cdot) = P(T^{-1}(\cdot))$. Intuitively, $T_{\#}P$ is obtained by transferring P using T.
- Define the solution map $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta), x = x(0)$.

Normalizing flow: formulation

- In the case of $Q = Z$ for $Z \sim \mathcal{N}(0, I)$, this is recognized as a *normalizing* flow [Chen et al., 2018, Grathwohl et al., 2019].
- Given a measurable function T defined on $x \sim P$, consider the push-forward measure $T_{\#}P$ where $(T_{\#}P)(\cdot) = P(T^{-1}(\cdot))$. Intuitively, $T_{\#}P$ is obtained by transferring P using T.
- Define the solution map $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta), x = x(0)$.
- Training of θ is done by minimizing the Kullback–Leibler divergence between $(T_\theta)_{\#}P$ and Z:

$$\min_\theta \mathrm{KL}((T_\theta)_{\#}P \| Z). \tag{3}$$

Normalizing flow: formulation

- In the case of $Q = Z$ for $Z \sim \mathcal{N}(0, I)$, this is recognized as a *normalizing* flow [Chen et al., 2018, Grathwohl et al., 2019].
- Given a measurable function T defined on $x \sim P$, consider the push-forward measure $T_{\#}P$ where $(T_{\#}P)(\cdot) = P(T^{-1}(\cdot))$. Intuitively, $T_{\#}P$ is obtained by transferring P using T.
- Define the solution map $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta), x = x(0)$.
- Training of θ is done by minimizing the Kullback–Leibler divergence between $(T_\theta)_{\#}P$ and Z:

$$\min_\theta \mathrm{KL}((T_\theta)_{\#}P \| Z). \tag{3}$$

- Note that (3) has a closed-form solution that can be numerically estimated on mini-batches.

Normalizing flow: implications

Many works have followed, including imposing regularization [Onken et al., 2021] and block-wise training of T_θ [Xu et al., 2023].

Normalizing flow: implications

Many works have followed, including imposing regularization [Onken et al., 2021] and block-wise training of T_θ [Xu et al., 2023].

- Regularization [Onken et al., 2021]:

  $\mathcal{R}(\theta) = \int_0^1 \frac{1}{2}\|f(x(t), t; \theta)\|^2 dt$, inspired by optimal transport.

Normalizing flow: implications

Many works have followed, including imposing regularization [Onken et al., 2021] and block-wise training of T_θ [Xu et al., 2023].

- Regularization [Onken et al., 2021]:

  $\mathcal{R}(\theta) = \int_0^1 \frac{1}{2}\|f(x(t), t; \theta)\|^2 dt$, inspired by optimal transport.

- Block-wise training [Xu et al., 2023]:

  Break down the training of a large $f(x(t), t; \theta)$ into smaller networks based on the JKO scheme.

Normalizing flow: implications

Benefits and drawbacks of normalizing flows.

Normalizing flow: implications

Benefits and drawbacks of normalizing flows.

- Benefits:
  - Direct likelihood estimation
  ($\min_\theta \mathrm{KL}((T_\theta)_\# P \| Z) = \max_\theta \log P(X;\theta) + C.$).
  - Better performance in low/mid dimensions.

Normalizing flow: implications

Benefits and drawbacks of normalizing flows.

- Benefits:
  - Direct likelihood estimation
  ($\min_\theta \mathrm{KL}((T_\theta)_\# P \| Z) = \max_\theta \log P(X;\theta) + C$.).
  - Better performance in low/mid dimensions.
- Drawbacks:
  - Computationally expensive due to ODE integration in training.
  - Scalability on high-dimensional P with large f_θ.

Normalizing flow: applications and results

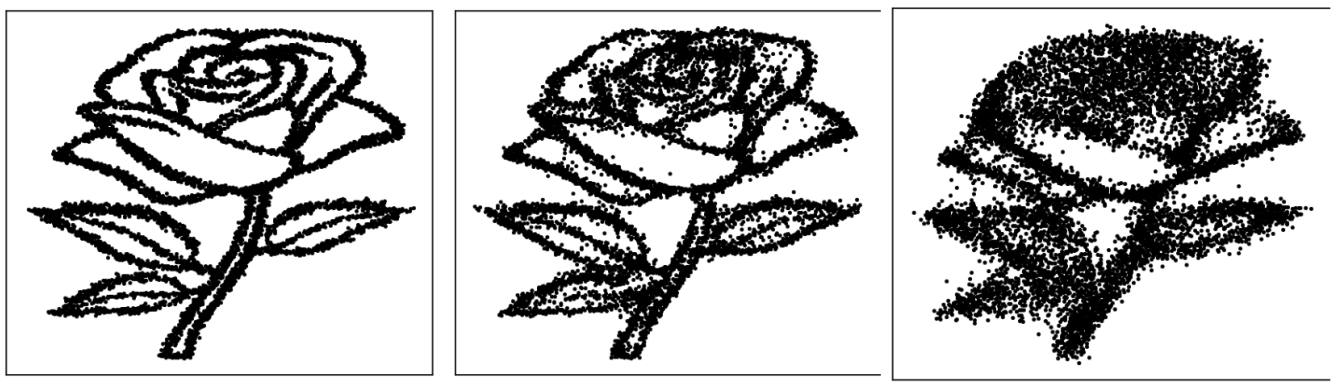
- Metrics (lower is better):

(1) Maximum mean discrepancy (MMD) between X_{test} and X_{gen}.

(2) Negative log-likelihood (i.e., estimate $\log P(X_{\text{test}})$).

Normalizing flow: applications and results

- Metrics (lower is better):
(1) Maximum mean discrepancy (MMD) between X_{test} and X_{gen}.
(2) Negative log-likelihood (i.e., estimate $\log P(X_{\text{test}})$).
- Outperforms diffusion models (e.g., ScoreSDE) in dimension $d = 2$:



(a) True data	JKO-iFlow	(e) ScoreSDE
τ: **2.79e-4**, MMD-c:	2.73e-4	6.90e-4
NLL	2.64	3.2

Xu et al., 2023. Normalizing flow neural networks by JKO scheme. *NeurIPS 2023, spotlight.*

Normalizing flow: applications and results

Better likelihood estimation than ScoreSDE in $d = 43$:

Model	# Param	Training		Testing
		# Batches	Batch size	NLL
JKO-iFlow	112K	2.72K	2000	10.55
ScoreSDE (same f_θ)	112K	33.90K	2000	20.70
ScoreSDE (larger f_θ)	831K	100K	2000	10.47

Xu et al., 2023. Normalizing flow neural networks by JKO scheme. *NeurIPS 2023, spotlight.*

Flow Matching: formulation

- Recall the ultimate goal is to learn f in *an* ODE $dx(t)/dt = f(x(t), t)$, where $x(0) \sim P$ and $x(1) \sim Q$.

Flow Matching: formulation

- Recall the ultimate goal is to learn f in *an* ODE $dx(t)/dt = f(x(t), t)$, where $x(0) \sim P$ and $x(1) \sim Q$.
- Flow matching [Lipman et al., 2023] proposes a direct way:

$$\min_{\theta} \|dx(t)/dt - f(x(t), t; \theta)\|_2^2, \tag{4}$$

where $dx(t)/dt$ is pre-specified given *an* interpolation of $x(t)$.

Flow Matching: formulation

- Recall the ultimate goal is to learn f in *an* ODE $dx(t)/dt = f(x(t), t)$, where $x(0) \sim P$ and $x(1) \sim Q$.
- Flow matching [Lipman et al., 2023] proposes a direct way:

$$\min_{\theta} \|dx(t)/dt - f(x(t), t; \theta)\|_2^2, \tag{4}$$

  where $dx(t)/dt$ is pre-specified given *an* interpolation of $x(t)$.
- Ex. Linear interpolation

$$\begin{aligned} & x(t) = x(0) + t(x(1) - x(0)) \\ \rightarrow \quad & dx(t)/dt = x(1) - x(0). \end{aligned} \tag{5}$$

Flow Matching: implications

Comparing to normalizing flow

- Training objective: recall the solution map

$T_\theta(x) = x + \int_0^1 f(x(s), s; \theta), x = x(0).$

$$\min_\theta \mathrm{KL}((T_\theta)_\# P \| Z). \qquad \text{(Normalizing flow)}$$

$$\min_\theta \| dx(t)/dt - f(x(t), t; \theta) \|_2^2. \qquad \text{(Flow matching)}$$

Flow Matching: implications

Comparing to normalizing flow

- Training objective: recall the solution map

  $T_\theta(x) = x + \int_0^1 f(x(s), s; \theta), x = x(0).$

  $$\min_\theta \mathrm{KL}((T_\theta)_\# P \| Z). \qquad \text{(Normalizing flow)}$$

  $$\min_\theta \| dx(t)/dt - f(x(t), t; \theta) \|_2^2. \qquad \text{(Flow matching)}$$

- Implication: flow matching is scalable to high-dimensional P and large f_θ, as its training involves no ODE integration.

Flow Matching: implications

Comparing to diffusion models in terms of training:

$$\min_{\theta} \lambda(t) \| s(x(t), t; \theta) - \nabla_{x(t)} \log p(x(t)|x(0)) \|_2^2. \quad \text{(Diffusion model)}$$

$$\min_{\theta} \| f(x(t), t; \theta) - dx(t)/dt \|_2^2. \quad \text{(Flow matching)}$$

Flow Matching: implications

Comparing to diffusion models in terms of training:

$$\min_\theta \lambda(t)\|s(x(t),t;\theta) - \nabla_{x(t)} \log p(x(t)|x(0))\|_2^2. \quad \text{(Diffusion model)}$$

$$\min_\theta \|f(x(t),t;\theta) - dx(t)/dt\|_2^2. \quad \text{(Flow matching)}$$

Remarks

- $\lambda(t)$ (noise scheduler) choice is crucial.
- Identical parametrization of s_θ and f_θ can be used.
- Flow matching is not restricted to having one end being Gaussian (recall $dx(t)/dt = x(1) - x(0)$ is always well-defined). However, tractability of $\nabla_{x(t)} \log p(x(t)|x(0))$ is an issue for general pairs of $(x(0), x(1))$.

Flow Matching: applications and results

Flow matching is better in terms of lower NLL (for density estimation) and lower FID (for generation) at fewer number of function evaluations (NFE).

	CIFAR-10			ImageNet 32×32			ImageNet 64×64		
Model	NLL↓	FID↓	NFE↓	NLL↓	FID↓	NFE↓	NLL↓	FID↓	NFE↓
Ablations									
DDPM	3.12	7.48	274	3.54	6.99	262	3.32	17.36	264
Score Matching	3.16	19.94	242	3.56	5.68	178	3.40	19.74	441
ScoreFlow	3.09	20.78	428	3.55	14.14	195	3.36	24.95	601
Ours									
FM ^{w}/ Diffusion	3.10	8.06	183	3.54	6.37	193	3.33	16.88	187
FM ^{w}/ OT	**2.99**	**6.35**	**142**	**3.53**	**5.02**	**122**	**3.31**	**14.45**	**138**

Lipman et al., 2023. Flow Matching for Generative Modeling. *ICLR 2023.*

Flow Matching: applications and results

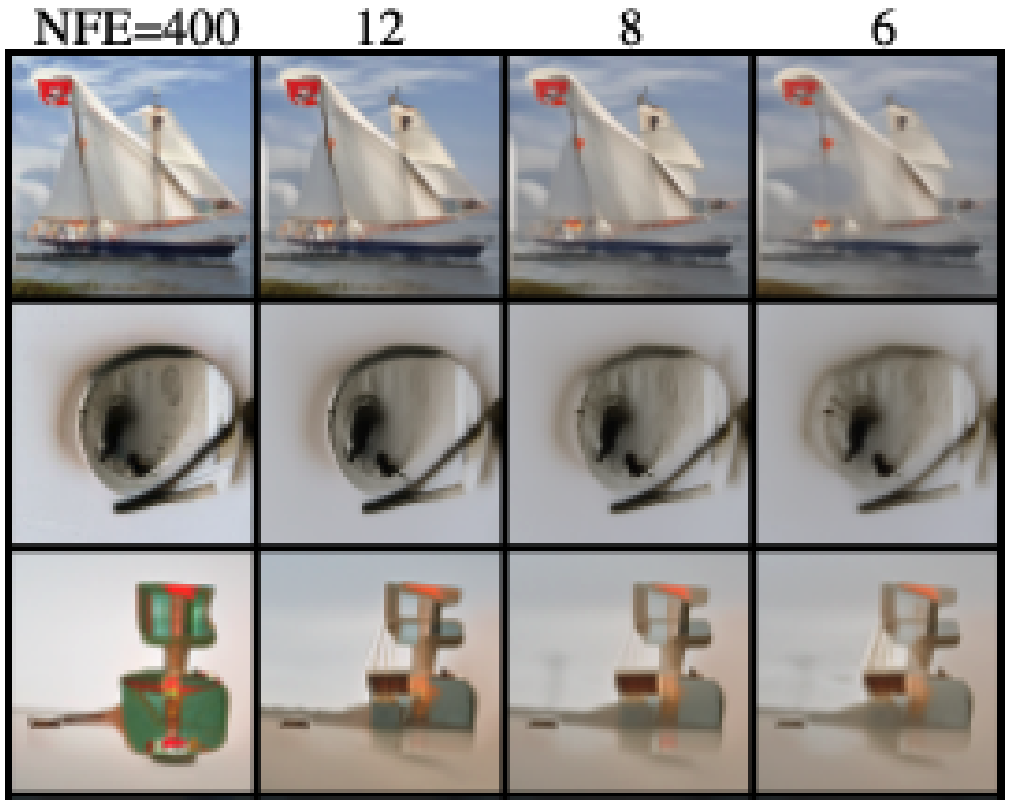
Take less NFEs to reach equally good performances during inference.

	ImageNet 32×32 NFE @ FID = 10	**ImageNet 64×64** NFE @ FID = 10
Diffusion	≥ 40	≥ 40
MultisampleFM w/ BatchOT	12	12

Pooladian et al., 2023. Multisample Flow Matching: Straightening Flows with Minibatch Couplings. *ICML 2023.*

Flow Matching: applications and results

Reasonable generative performance under small NFEs.



Pooladian et al., 2023. Multisample Flow Matching: Straightening Flows with Minibatch Couplings. *ICML 2023.*

Flow Matching: applications and results

Fast text-to-image generation (even in one euler step due to straightness).

(B) One-step InstaFlow-0.9B + SDXL-Refiner (1024 × 1024)

(C) Images from one-step InstaFlow-1.7B **(0.12s per image,** 512 × 512)

Liu et al., 2024. InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation. *ICLR 2024.*

Summary

- Highlight the use of flow model in generative modeling and likelihood estimation.
- Advocated two approaches: normalizing flow and flow matching.

Summary

- Highlight the use of flow model in generative modeling and likelihood estimation.
- Advocated two approaches: normalizing flow and flow matching.
- Summary table (1 being the best):

	Training	Inference	Performance
Normalizing flow	3	1	low/mid: 1; high: 3
Flow Matching	1	1	low/mid: 2; high: 1
Diffusion models	$\approx$1	3	low/mid: 2; high: 1

Disclaimer: the table is based on general observation (to the best of my knowledge).

Open questions

- Comparison against diffusion models on large-scale datasets, and even be a substitute.
- Faster inference with simpler designs.
- Application for scientific problems (AI4Science).

References I

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

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.

Derek Onken, S Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

References II

Chen Xu, Xiuyuan Cheng, and Yao Xie. Normalizing flow neural networks by JKO scheme. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Appendix: Obtain $\log p(x)$ in flow models

- Given a new test $x(0) = x$, we have by the instantaneous change-of-variable formula [Chen et al., 2018] that

$$\log p(x(1)) - \log p(x(0)) = -\int_0^1 \text{div}(f(x(t), t))dt. \quad (6)$$

- If $\log p(x(1))$ is known (i.e., when $Q = \mathcal{N}(0, I)$), we can numerically estimate $\log p(x(0))$ with (6).
- $\text{div}(f(x(t), t))$ can be numerically estimated via trace estimator or evaluated using autograd.

Appendix: Normalizing flow simplified loss and pseudo-code

- Using the same change-of-variable formula, we can re-write (3) as (up to constants):

$$\min_{\theta} \mathbb{E}_{x\sim P}\left[\|T_\theta(x)\|^2 - \int_0^1 \mathrm{div}(f(x(t),t;\theta))dt\right] \qquad (7)$$

- To efficiently train in high-dimensions, we use a finite-difference trace estimator [Xu et al., 2023] to evaluate div in (7).
- Training is done by mini-batch SGD upon drawing $x(0) \sim P$.

Appendix: Flow matching pseudo-code

1. Randomly draw batches $x(0) \sim P, x(1) \sim Q$.
2. Randomly sample $t \sim U[0, 1]$.
3. Compute $x(t)$ and $dx(t)/dt$ to evaluate (4).
4. Run gradient descent on θ.

Appendix: Comparison with diffusion models

- ODE, e.g., JKO flow
- **Particles** $x_0 \sim p$, push particles by velocity field $v(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$

$$\dot{x}_t = v(x_t, t)$$

- **Distribution**: Continuity equation $X_t \sim \rho_t$

$$\partial_t \rho_t + \nabla \cdot (\rho_t v_t) = 0.$$

- SDE, score matching
- **Noisy samples** $X_0 \sim P$, sample noisy trajectory

$$dX_t = -\nabla V(X_t)dt + \sqrt{2}dW_t$$

- **Distribution**: Fokker-Plank equation $X_t \sim \rho_t$

$$\partial_t \rho_t = \nabla \cdot (\rho_t \nabla V + \nabla \rho_t)$$

Same when setting $v(x, t) = -\nabla V(x) - \nabla \log \rho_t$ "score function"