# Normalizing flow neural networks by JKO scheme

Chen Xu[1]
joint work with Xiuyuan Cheng[2] and Yao Xie[1]

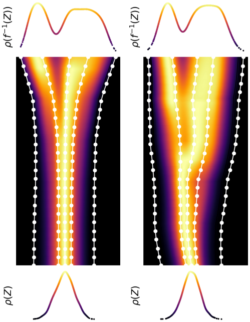
[1] School of Industrial and Systems Engineering,
Georgia Institute of Technology
[2] Department of Mathematics, Duke Unversity

# Goal

- Improve the design and training of **normalizing flows**. Namely, invertible transformation of $X \leftrightarrow Z$ given samples from $P_X$.

- Allow efficient sampling from $P_X$ and likelihood estimation $\log p(X)$.

- More computational and memory efficient than existing methods.




**(a)** JKO-iFlow **(b)** Usual CNF

**Figure 1:** $X \leftrightarrow Z, Z \sim \mathcal{N}(0, I_d)$.

# Mathematical background

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty} \rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.

# Mathematical background

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty}\rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.

- Non-unique flow: we consider flow induced by ODE of $x(t) \sim \rho(x,t)$

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

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

# Mathematical background

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty} \rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.

- Non-unique flow: we consider flow induced by ODE of $x(t) \sim \rho(x,t)$

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

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

- Transport regularization: $\mathcal{T} = \int_0^1 \mathbb{E}_{x\sim\rho(\cdot,t)} \|f(x,t)\|^2 dt$. Recovers the Wasserstein-2 optimal transport under the Benamou-Brenier formula [Villani 2009].

# Mathematical background (cont.)

- Flow induced by ODE of $x(t) \sim \rho(x,t)$

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

- Normalizing flow models learn $f$ using neural networks $f_\theta$.

# Mathematical background (cont.)

- Flow induced by ODE of $x(t) \sim \rho(x,t)$

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

- Normalizing flow models learn $f$ using neural networks $f_\theta$.
- Specifically, the objective is

$$\min_\theta \mathsf{KL}((T_\theta)_\# p_X \| p_Z) + \mathcal{R}(\theta). \tag{2}$$

- $T_\theta(x) = x + \int_0^1 f_\theta(x(s),s), x(0) = x$;
  $T_\#$ is the push-forward operation with $(T_\# p)(A) = p(T^{-1}(A))$ for a measureable set $A$.

# Mathematical background (cont.)

- Flow induced by ODE of $x(t) \sim \rho(x,t)$

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

- Normalizing flow models learn $f$ using neural networks $f_\theta$.
- Specifically, the objective is

$$\min_\theta \mathsf{KL}((T_\theta)_\# p_X \| p_Z) + \mathcal{R}(\theta). \tag{2}$$

- $T_\theta(x) = x + \int_0^1 f_\theta(x(s), s), x(0) = x$;
  $T_\#$ is the push-forward operation with $(T_\# p)(A) = p(T^{-1}(A))$ for a measureable set $A$.
- (2) is equivalent to maximizing $\log p(X)$ up to constants [Onken et al., 2021].

# Current approaches

- *Continuous* normalizing flow (CNF) [Grathwohl et al., 2019, Onken et al., 2021] based on Neural ODE [Chen et al., 2019].

# Current approaches

- *Continuous* normalizing flow (CNF) [Grathwohl et al., 2019, Onken et al., 2021] based on Neural ODE [Chen et al., 2019].
- Most existing continuous flows pre-specify the number of blocks $L$ to be trained
  - Namely, the integral from $[0, 1]$ is broken into a sequence of $L$ smaller integrals each with $f_{\theta_l}$, or the model $f_\theta$ itself is a composition of $L$ smaller ones of identical architecture.

# Current approaches

- *Continuous* normalizing flow (CNF) [Grathwohl et al., 2019, Onken et al., 2021] based on Neural ODE [Chen et al., 2019].
- Most existing continuous flows pre-specify the number of blocks $L$ to be trained
  - Namely, the integral from $[0, 1]$ is broken into a sequence of $L$ smaller integrals each with $f_{\theta_l}$, or the model $f_\theta$ itself is a composition of $L$ smaller ones of identical architecture.
- Challenges are
  - Design: how to specify $L$.
  - Computation: joint training of all $L$ blocks.
  - Memory: samples are passed through all $L$ blocks.

# Main contribution

- Introduce block-wise training of CNF models, where each block is allowed simpler architecture.
- Efficient training with less computation and memory cost.
- Better generative performance and likelihood estimation vs CNF and diffusion models on simulated and real data.

# Proposed JKO-iFlow

- Our JKO-iFlow is inspired by the Jordan-Kinderleherer-Otto (JKO) scheme [Jordan et al., 1998]: starting at $p_0 = \rho_0 \in \mathcal{P}$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$p_{k+1} = \arg\min_{\rho \in \mathcal{P}} \mathsf{KL}(\rho \| \| p_Z) + \frac{1}{2h} W_2^2(p_k, \rho). \quad (1)$$

# Proposed JKO-iFlow

- Our JKO-iFlow is inspired by the Jordan-Kinderleherer-Otto (JKO) scheme [Jordan et al., 1998]: starting at $p_0 = \rho_0 \in \mathcal{P}$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$p_{k+1} = \arg\min_{\rho \in \mathcal{P}} \mathsf{KL}(\rho \| \| p_Z) + \frac{1}{2h} W_2^2(p_k, \rho). \quad (1)$$

- It is equivalent to solve for the following transport map:

$$T_{k+1} = \arg\min_{T:\mathbb{R}^d \to \mathbb{R}^d} \mathsf{KL}(T_\# p_k \| \| p_Z) + \frac{1}{2h} \mathbb{E}_{x \sim p_k} \| x - T(x) \|^2. \quad (2)$$

# Proposed JKO-iFlow

- Our JKO-iFlow is inspired by the Jordan-Kinderleherer-Otto (JKO) scheme [Jordan et al., 1998]: starting at $p_0 = \rho_0 \in \mathcal{P}$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$p_{k+1} = \arg\min_{\rho \in \mathcal{P}} \mathsf{KL}(\rho \| \| p_Z) + \frac{1}{2h} W_2^2(p_k, \rho). \quad (1)$$

- It is equivalent to solve for the following transport map:

$$T_{k+1} = \arg\min_{T:\mathbb{R}^d \to \mathbb{R}^d} \mathsf{KL}(T_{\#} p_k \| \| p_Z) + \frac{1}{2h} \mathbb{E}_{x \sim p_k} \| x - T(x) \|^2. \quad (2)$$

- Theoretical analyses of solving (1), which is the $W_2$ proximal GD problem, are recently presented in [Cheng et al., 2023].

# Proposed JKO-iFlow (cont.)

- Let $k$-th block $T_{\theta_k}(x) = x + \int_0^1 f_{\theta_k}(x(s), s)$ with parameters $\theta_k$.

# Proposed JKO-iFlow (cont.)

- Let $k$-th block $T_{\theta_k}(x) = x + \int_0^1 f_{\theta_k}(x(s), s)$ with parameters $\theta_k$.
- Using the instantaneous change-of-variable formula [Chen et al., 2018], we can show that up to constants

$$\mathsf{KL}((T_{\theta_k})_{\#}p_k \| p_Z) = \mathbb{E}_{x\sim p_k}\left[\|T_{\theta_k}(x)\|^2 - \int_0^1 \nabla \cdot f_{\theta_k}(x(s), s)ds\right].$$

# Proposed JKO-iFlow (cont.)

- Let $k$-th block $T_{\theta_k}(x) = x + \int_0^1 f_{\theta_k}(x(s), s)$ with parameters $\theta_k$.
- Using the instantaneous change-of-variable formula [Chen et al., 2018], we can show that up to constants

$$\mathsf{KL}((T_{\theta_k})_{\#}p_k \| p_Z) = \mathbb{E}_{x \sim p_k}\left[\|T_{\theta_k}(x)\|^2 - \int_0^1 \nabla \cdot f_{\theta_k}(x(s), s) ds\right].$$

- Thus, we train the $k$-th block given the trained $(k-1)$-th block.
- The full model $T_\theta = T_{\theta_K} \circ \ldots \circ T_{\theta_1}$, where $(T_\theta)_{\#}p_X \approx \mathcal{N}(0, I_d)$.

# Proposed JKO-iFlow (cont.)

- Let $k$-th block $T_{\theta_k}(x) = x + \int_0^1 f_{\theta_k}(x(s), s)$ with parameters $\theta_k$.
- Using the instantaneous change-of-variable formula [Chen et al., 2018], we can show that up to constants

$$\mathsf{KL}((T_{\theta_k})_{\#} p_k \| p_Z) = \mathbb{E}_{x \sim p_k}\left[\|T_{\theta_k}(x)\|^2 - \int_0^1 \nabla \cdot f_{\theta_k}(x(s), s) ds\right].$$

- Thus, we train the $k$-th block given the trained $(k-1)$-th block.
- The full model $T_\theta = T_{\theta_K} \circ \ldots \circ T_{\theta_1}$, where $(T_\theta)_{\#} p_X \approx \mathcal{N}(0, I_d)$.

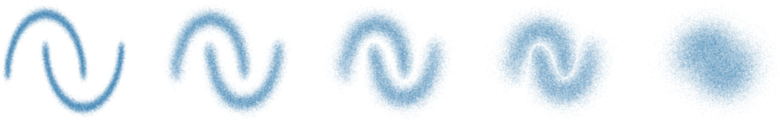


**Figure 2:** Toy example with 4 trained blocks. $P_X$ = two moons.

# Proposed JKO-iFlow (cont.)

- Benefits:
  - Simpler design and easier training of $f_{\theta_k}$.
  - Allow stopping criterion to determine number of blocks.
  - No sampling as diffusion models or variational learning.

# Proposed JKO-iFlow (cont.)

- Benefits:
  - Simpler design and easier training of $f_{\theta_k}$.
  - Allow stopping criterion to determine number of blocks.
  - No sampling as diffusion models or variational learning.

- Additional techniques:
  - *Reparametrization* adjusts the penalty term $h_k$ to encourage more even $W_2$ block movements, due to exponential convergence by JKO theory.
  - *Refinement* interpolates with $h_k = h_k/c$ to increase accuracy.

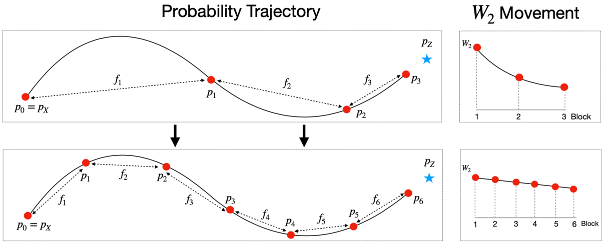



**Figure 1:** Before and after reparametrization and refinement.

# Experiments–simulation

- Baselines: two discrete-time flow [Berhmann et al., 2019, Xu et al., 2022], two continuous-time flow [Grathwohl et al., 2019, Onken et al., 2021], and one diffusion model [Song et al., 2021].

- **Takeaway:** JKO-iFlow shows better likelihood estimation and generative performance.

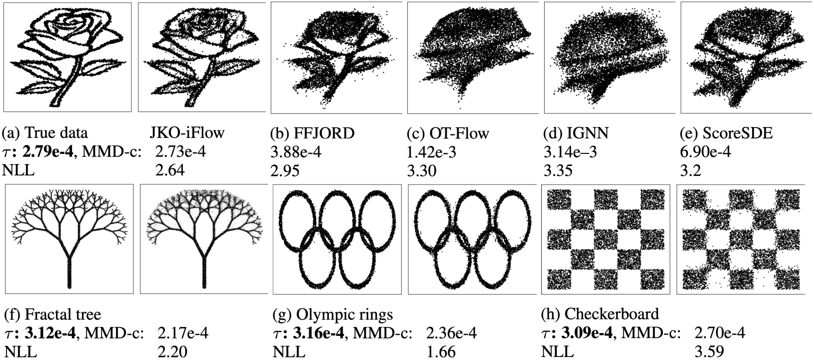
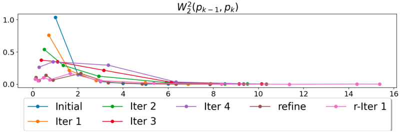


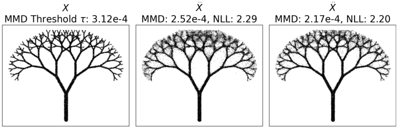

**Figure 1:** Two-dimensional datasets visualized as scatter plots.

# Experiments–simulation (cont.)

- Benefits of reparametrization + refinement.
- **Takeaway:** improved performance on edges, at which we have few samples.




(a) Per-block $W_2^2$ over reparameterization iterations and refinement ('r-Iter 1' means one reparameterization iteration after refinement).




(b) Results at Iter 4 (middle) and r-Iter 1 (right). MMD and NLL values are shown in the title.

**Figure 1:** $W_2$ movement before and after reprametrization and refinement, as well as the generated samples.

# Experiments–real data

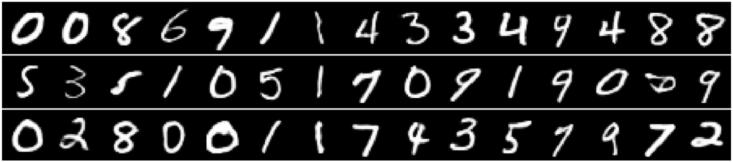
- High-dimensional tabular daatsets ($d = 6, 8, 43, 63$).
- **Takeaway:** competitive or better performance under much less number of mini-batch SGD with same model capacity.

| Data Set | Model | # Param | Training | | | | Testing | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Time (h) | # Batches | Time/Batches (s) | Batch size | MMD-m | MMD-1 | NLL |
| | | | | | | | **$\tau$: 1.73e-4** | **$\tau$: 2.90e-4** | |
| | JKO-iFlow | 76K, L=4 | 0.07 | 0.76K | 3.51e-1 | 10000 | 9.86e-5 | 2.40e-4 | -0.12 |
| | OT-Flow | 76K | 0.36 | 7.58K | 1.71e-1 | 10000 | 7.58e-4 | 5.35e-4 | 0.32 |
| **POWER** | FFJORD | 76K, L=4 | 0.67 | 7.58K | 3.18e-1 | 10000 | 9.89e-4 | 1.16e-3 | 0.63 |
| $d = 6$ | IGNN | 304K, L=16 | 0.29 | 7.58K | 1.38e-1 | 10000 | 1.93e-3 | 1.59e-3 | 0.95 |
| | IResNet | 304K, L=16 | 0.41 | 7.58K | 1.95e-1 | 10000 | 3.92e-3 | 2.43e-2 | 3.37 |
| | ScoreSDE | 76K | 0.06 | 7.58K | 2.85e-2 | 10000 | 9.12e-4 | 6.08e-3 | 3.41 |
| | ScoreSDE | 76K | 0.60 | 75.80K | 2.85e-2 | 10000 | 7.12e-4 | 5.04e-3 | 3.33 |
| | JKO-iFlow | 57K, L=3 | 0.05 | 0.76K | 2.63e-1 | 10000 | 3.86e-4 | 7.20e-4 | -0.06 |
| | | | | | | | **$\tau$: 1.85e-4** | **$\tau$: 2.73e-4** | |
| | JKO-iFlow | 76K, L=4 | 0.07 | 0.76K | 3.32e-1 | 5000 | 1.52e-4 | 5.00e-4 | -7.65 |
| | OT-Flow | 76K | 0.23 | 7.60K | 1.09e-1 | 5000 | 1.99e-4 | 5.16e-4 | -6.04 |
| **GAS** | FFJORD | 76K, L=4 | 0.65 | 7.60K | 3.08e-1 | 5000 | 1.87e-3 | 3.28e-3 | -2.65 |
| $d = 8$ | IGNN | 304K, L=16 | 0.34 | 7.60K | 1.61e-1 | 5000 | 6.74e-3 | 1.43e-2 | -1.65 |
| | IResNet | 304K, L=16 | 0.46 | 7.60K | 2.18e-1 | 5000 | 3.20e-3 | 2.73e-2 | -1.17 |
| | ScoreSDE | 76K | 0.03 | 7.60K | 1.42e-2 | 5000 | 1.05e-3 | 8.36e-4 | -3.69 |
| | ScoreSDE | 76K | 0.30 | 76.00K | 1.42e-2 | 5000 | 2.23e-4 | 3.38e-4 | -5.58 |
| | JKO-iFlow | 95K, L=5 | 0.09 | 0.76K | 4.15e-1 | 5000 | 1.51e-4 | 3.77e-4 | -7.80 |
| | | | | | | | **$\tau$: 2.46e-4** | **$\tau$: 3.75e-4** | |
| | JKO-iFlow | 112K, L=4 | 0.03 | 0.34K | 3.61e-1 | 2000 | 9.66e-4 | 3.79e-4 | 12.55 |
| | OT-Flow | 112K | 0.21 | 3.39K | 2.23e-1 | 2000 | 6.58e-4 | 3.79e-4 | 11.44 |
| **MINIBOONE** | FFJORD | 112K, L=4 | 0.28 | 3.39K | 2.97e-1 | 2000 | 3.51e-3 | 4.12e-4 | 23.77 |
| $d = 43$ | IGNN | 448K, L=16 | 0.63 | 3.39K | 6.69e-1 | 2000 | 1.21e-2 | 4.01e-4 | 26.45 |
| | IResNet | 448K, L=16 | 0.71 | 3.39K | 7.54e-1 | 2000 | 2.13e-3 | 4.16e-4 | 22.36 |
| | ScoreSDE | 112K | 0.01 | 3.39K | 6.37e-3 | 2000 | 5.86e-1 | 4.33e-4 | 27.38 |
| | ScoreSDE | 112K | 0.10 | 33.90K | 6.37e-3 | 2000 | 4.17e-3 | 3.87e-4 | 20.70 |
| | | | | | | | **$\tau$: 1.38e-4** | **$\tau$: 1.01e-4** | |
| | JKO-iFlow | 396K, L=4 | 0.05 | 1.03K | 1.85e-1 | 1000 | 2.24e-4 | 1.91e-4 | -153.82 |
| | OT-Flow | 396K | 0.62 | 10.29K | 2.17e-1 | 1000 | 5.43e-1 | 6.49e-1 | -104.62 |
| **BSDS300** | FFJORD | 396K, L=4 | 0.54 | 10.29K | 1.89e-1 | 1000 | 5.60e-1 | 6.76e-1 | -37.80 |
| $d = 63$ | IGNN | 990K, L=10 | 1.71 | 10.29K | 5.98e-1 | 1000 | 5.64e-1 | 6.86e-1 | -37.68 |
| | IResNet | 990K, L=10 | 2.05 | 10.29K | 7.17e-1 | 1000 | 5.50e-1 | 5.50e-1 | -33.11 |
| | ScoreSDE | 396K | 0.01 | 10.29K | 3.50e-3 | 1000 | 5.61e-1 | 6.60e-1 | -7.55 |
| | ScoreSDE | 396K | 0.10 | 102.90K | 3.50e-3 | 1000 | 5.61e-1 | 6.62e-1 | -7.31 |
| | JKO-iFlow | 396K, L=4 | 0.08 | 1.03K | 2.76e-1 | 5000 | 1.41e-4 | 8.83e-5 | -156.68 |

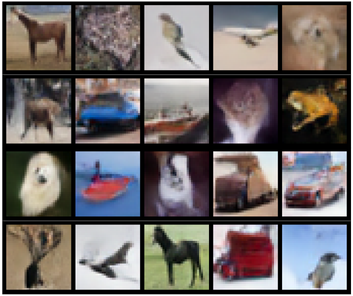
**Figure 1:** Quantitative metrics (MMD and NLL)

# Experiments–real data (cont.)

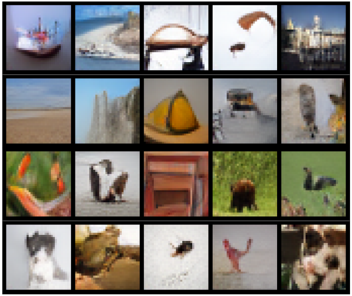
- Image data in the latent space of pre-trained variational auto-encoders [Esser et al., 2021].



(a) Generated MNIST digits. FID: 7.95.



(b) Generated CIFAR10 images. FID: 29.10.



(c) Generated Imagenet-32 images. FID: 20.10.

# Conclusions

- Propose JKO-iFlow, a neural ODE model that trains each residual block in a step-wise fashion.
- Leads to improved performance with less computation against flow and diffusion models.

Xu, C., Cheng, X., and Xie, Y. Normalizing flow neural networks by JKO scheme. *NeurIPS 2023, spotlight.*