

OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport

AAAI 2021

Derek Onken
Emory University
derekonken.com



EMORY UNIVERSITY
FOUNDED 1836

Collaborators and Acknowledgments

UCLA



Samy Wu Fung
UCLA



Xingjian Li
Emory



Lars Ruthotto
Emory



EMORY
UNIVERSITY

Funding:



UNITEDHEALTH GROUP®

Special thanks: Organizers and staff of IPAM Long Program MLP 2019 and NVIDIA.

Overview

- **Background**

- ▶ Normalizing Flows
- ▶ Continuous Normalizing Flows

- **Mathematical Formulation**

- ▶ Optimal Transport
- ▶ Potential Function
- ▶ Hamilton-Jacobi-Bellman (HJB) Regularizers

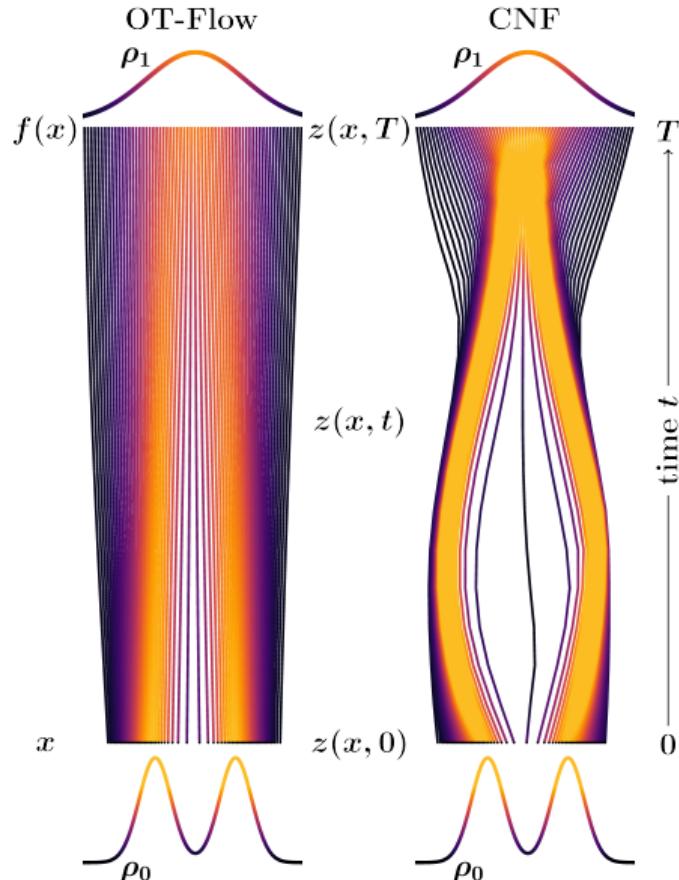
- **Numerical Implementation**

- ▶ Efficient Exact Trace Computation
- ▶ Discretize-then-Optimize

- **Results**

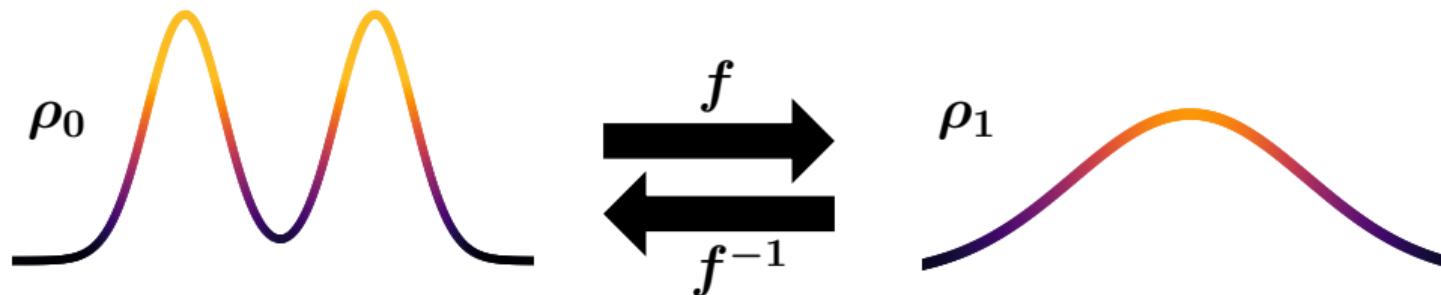
- ▶ 8x training speed-up
- ▶ 24x testing speed-up

- **Conclusion**



Normalizing Flows for Density Estimation

A normalizing flow^{1,2} is an invertible mapping $f: \mathbb{R}^d \rightarrow \mathbb{R}^d$ between an arbitrary probability distribution and a standard normal distribution with respective densities ρ_0 and ρ_1



By the change of variables formula, the flow satisfies

$$\log \rho_0(\mathbf{x}) = \log \rho_1(f(\mathbf{x})) + \log |\det \nabla f(\mathbf{x})| \quad \text{for all } \mathbf{x} \in \mathbb{R}^d. \quad (1)$$

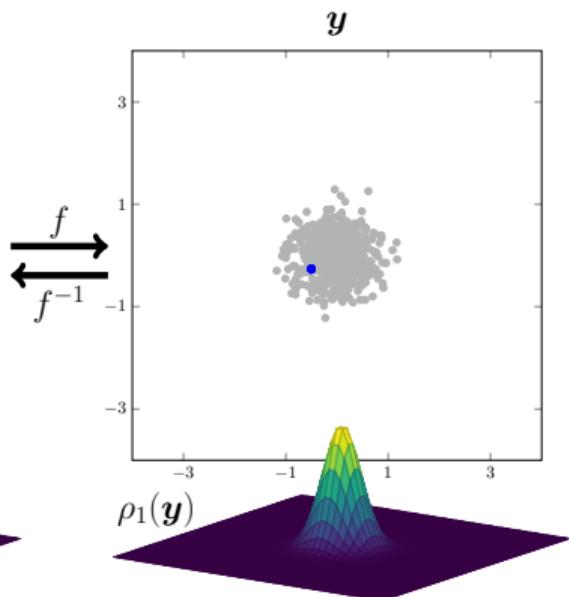
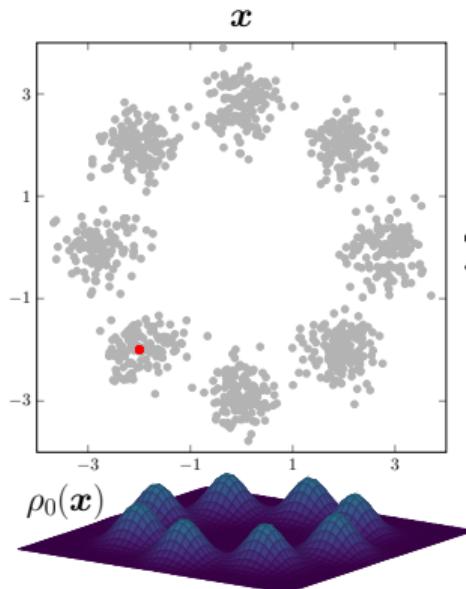
¹Rezende and Mohamed. "Variational Inference with Normalizing Flows". 2015.

²Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". 2019.

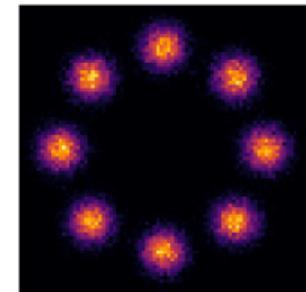
Two-Dimensional Example

Gaussian Mixture Problem

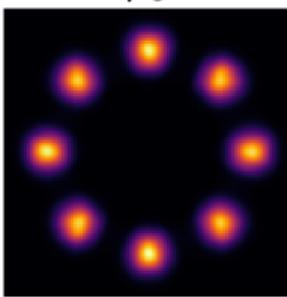
- sample \mathbf{x}_0
- $\mathbf{z}(\mathbf{x}_0, T) = f(\mathbf{x}_0)$



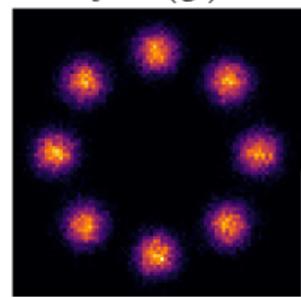
Data
 \mathbf{x}



Estimate
 ρ_0



Generation
 $f^{-1}(\mathbf{y})$



Continuous Normalizing Flows (CNFs)

Issue: log-determinants cost $\mathcal{O}(d^3)$ FLOPS in general

One Solution: replace the log-det with a trace computation for $\mathcal{O}(d^2)$ FLOPS in general

Using a neural ordinary differential equation (ODE)³ leads to the CNF⁴

$$\partial_t \begin{bmatrix} z(\mathbf{x}, t) \\ \ell(\mathbf{x}, t) \end{bmatrix} = \begin{bmatrix} \mathbf{v}(z(\mathbf{x}, t), t; \boldsymbol{\theta}) \\ \text{tr}(\nabla \mathbf{v}(z(\mathbf{x}, t), t; \boldsymbol{\theta})) \end{bmatrix}, \quad \begin{bmatrix} z(\mathbf{x}, 0) \\ \ell(\mathbf{x}, 0) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix}. \quad (2)$$

where

- $z(\mathbf{x}, t)$ are the features for initial state \mathbf{x} at time $t \in [0, T]$
- $\mathbf{v}: \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^d$ is a neural network layer with parameters $\boldsymbol{\theta}$
- $f(\mathbf{x}) = z(\mathbf{x}, T)$
- $\ell(\mathbf{x}, T) := \log \rho_0(\mathbf{x}) - \log \rho_1(f(\mathbf{x}))$

³Chen et al. "Neural Ordinary Differential Equations". 2018.

⁴Grathwohl et al. "FFJORD: Free-form Continuous Dynamics for Scalable Reversible...". 2019.

CNF Optimization Problem

For expected negative log-likelihood^{5,6}

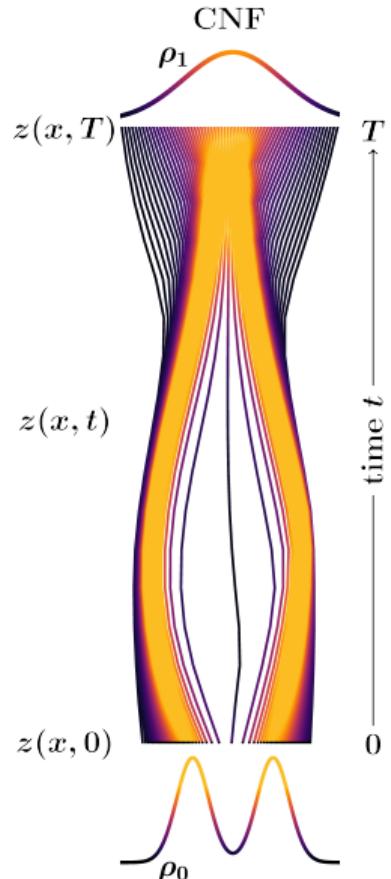
$$C(\mathbf{x}, T) := \frac{1}{2} \|z(\mathbf{x}, T)\|^2 - \ell(\mathbf{x}, T) + \frac{d}{2} \log(2\pi),$$

we optimize

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\rho_0(\mathbf{x})} \{C(\mathbf{x}, T)\}$$

subject to

$$\partial_t \begin{bmatrix} z(\mathbf{x}, t) \\ \ell(\mathbf{x}, t) \end{bmatrix} = \begin{bmatrix} \mathbf{v}(z(\mathbf{x}, t), t; \boldsymbol{\theta}) \\ \text{tr}(\nabla \mathbf{v}(z(\mathbf{x}, t), t; \boldsymbol{\theta})) \end{bmatrix}, \quad \begin{bmatrix} z(\mathbf{x}, 0) \\ \ell(\mathbf{x}, 0) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix}.$$



⁵Rezende and Mohamed. "Variational Inference with Normalizing Flows". 2015.

⁶Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". 2019.

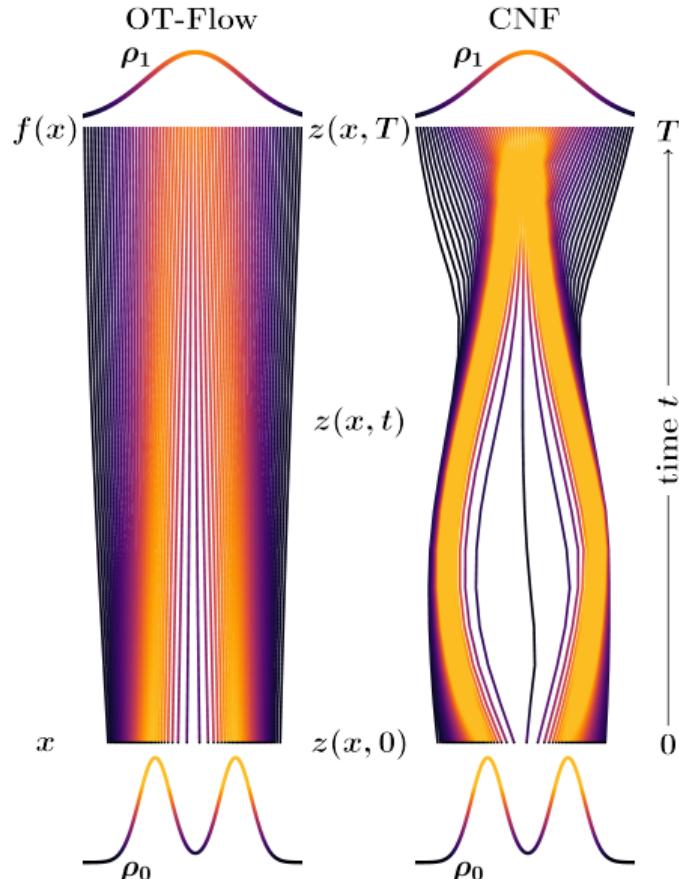
High Costs of CNFs

CNFs have high computation cost because

- Trajectories can be complicated leading to high number of function evaluations
- Expensive trace computation
 - ▶ State-of-the-art train with $\mathcal{O}(d)$ trace cost by using Hutchinson's trace estimator⁷

$$\text{tr}(\nabla \mathbf{v}) = \mathbb{E}_{\phi(\epsilon)} \left\{ \epsilon^\top \nabla \mathbf{v} \epsilon \right\}$$

for noise vector ϵ w/ density $\phi(\epsilon)$, $\mathbb{E}\{\epsilon\} = 0$, $\text{Cov}(\epsilon) = I$



⁷Hutchinson. "A Stochastic Estimator of the Trace of the Influence Matrix for...". 1990.

Straight Trajectories

Include some optimal transport (OT)
⇒ model named OT-Flow

Arclength of the trajectories

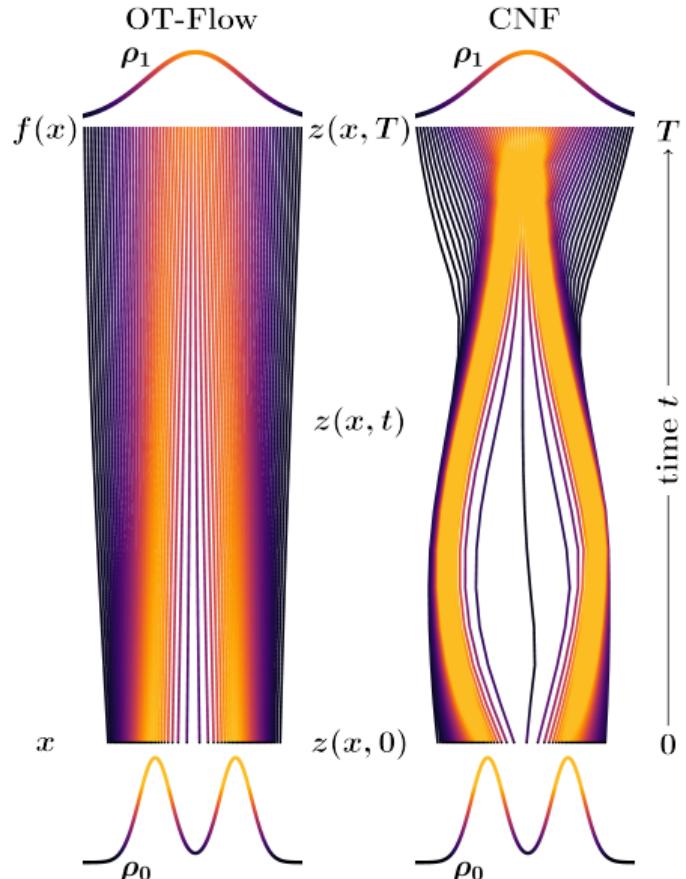
$$L(\mathbf{x}, T) = \int_0^T \frac{1}{2} \|\mathbf{v}(z(\mathbf{x}, t), t; \boldsymbol{\theta})\|^2 dt$$

We regularize the optimization problem

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\rho_0(\mathbf{x})} \left\{ C(\mathbf{x}, T) + L(\mathbf{x}, T) \right\} \quad (3)$$

subject to (2).

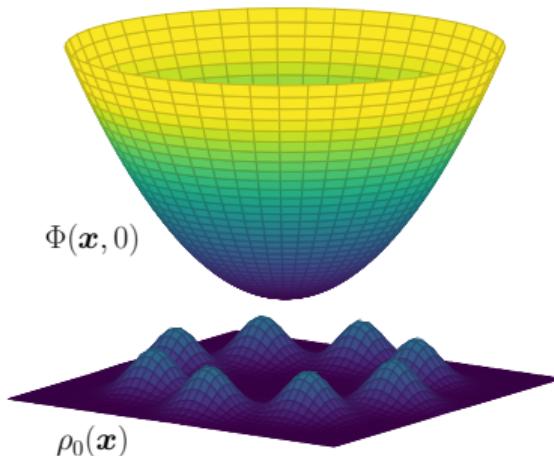
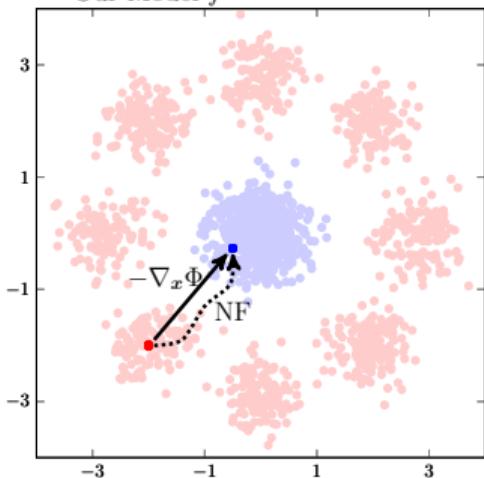
Now, a unique mapping exists.



Potential Function Φ

Apply the Pontryagin maximum principle⁸ to (3)

- $x \sim \rho_0(x)$
- $y \sim \rho_1(y)$
- General Normalizing Flow f
- Our Model f



There exists a scalar potential function $\Phi: \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}$ such that

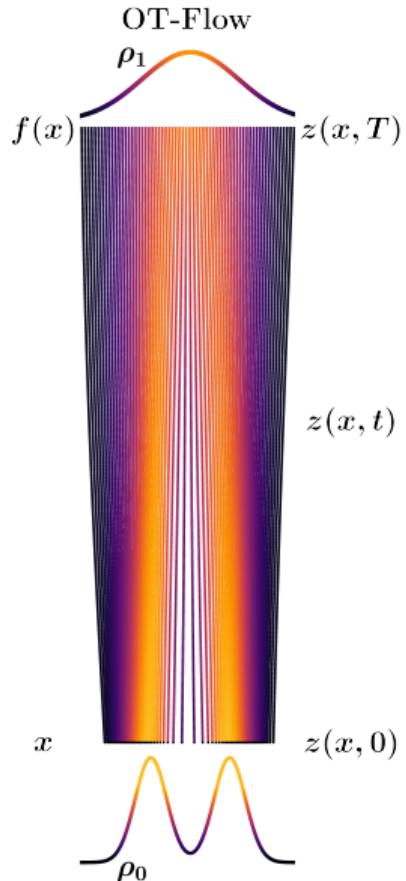
$$\mathbf{v}(x, t; \theta) = -\nabla \Phi(x, t; \theta).$$

Analogous to classical physics, samples move in a manner to minimize their potential.

We parametrize potential Φ instead of \mathbf{v} .

⁸Pontryagin et al. *The Mathematical Theory of Optimal Processes*. 1962.

HJB Equations



The optimality conditions of (3) lead to another regularizer.

Potential Φ satisfies the Hamilton-Jacobi-Bellman (HJB) equations⁹

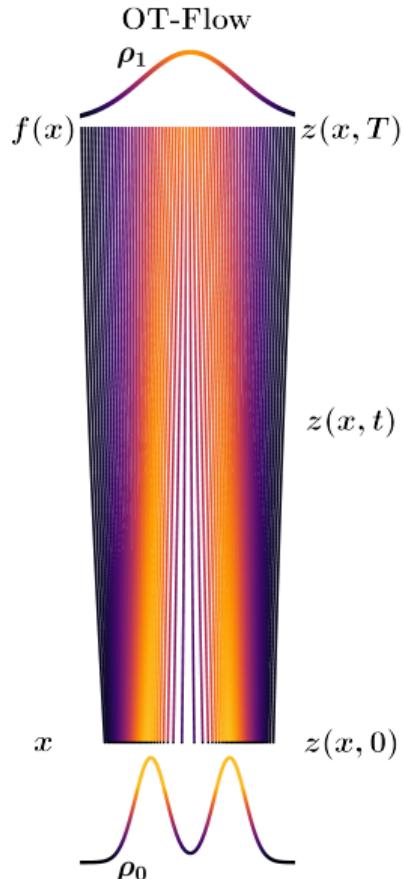
$$-\partial_t \Phi(z(x, t), t) = -\frac{1}{2} \|\nabla \Phi(z(x, t), t)\|^2, \quad 0 < t < T$$

$$\Phi(z(x, T), T) = 1 + \log(\rho_0(x)) - \log(\rho_1(z(x, T))) - \ell(x, T)$$

Terminal condition $\Phi(z(x, T), T)$ derives from the variational derivative of the Kullback-Leibler (KL) divergence

⁹Bellman. *Dynamic Programming*. 1957.

HJB Regularizer R



Penalize deviations from the HJB equation

We add another regularizer, so the optimization problem is

$$\min_{\theta} \mathbb{E}_{\rho_0(\mathbf{x})} \left\{ C(\mathbf{x}, T) + L(\mathbf{x}, T) + R(\mathbf{x}, T) \right\}$$

subject to (2).

The HJB regularizer¹⁰ is computed as

$$R(\mathbf{x}, T) = \int_0^T \left| \partial_t \Phi(\mathbf{z}(\mathbf{x}, t), t) - \frac{1}{2} \|\nabla \Phi(\mathbf{z}(\mathbf{x}, t), t)\|^2 \right| dt.$$

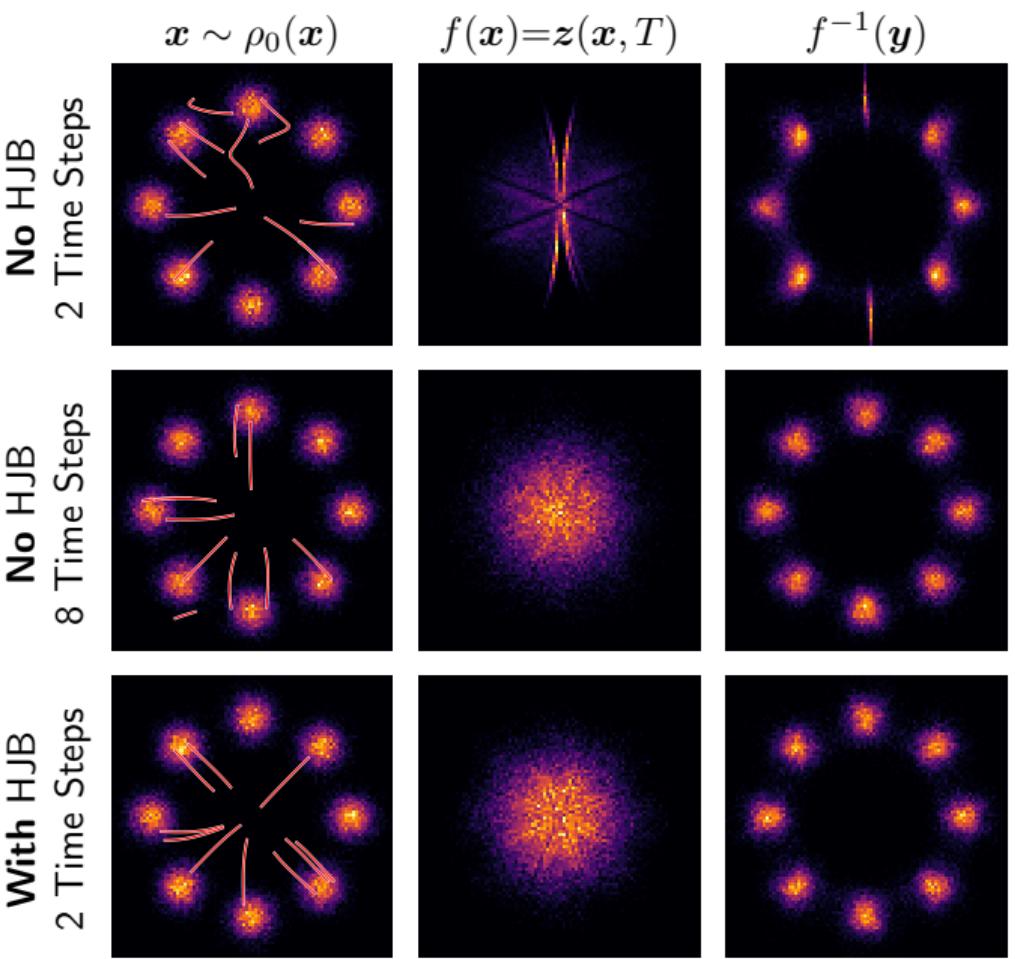
¹⁰Yang and Karniadakis. "Potential Flow Generator With L_2 Optimal Transport Regularity...". 2020.

HJB Regularizer Effectiveness

Compare three models:

- No HJB regularizer with only 2 time steps
- No HJB regularizer with only 8 time steps
- Using HJB regularizer with only 2 time steps

HJB regularizer gives similar results to using many time steps



OT-Flow Formulation

We incorporate the time integration in the ODE solver.

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\rho_0(\mathbf{x})} \left\{ C(\mathbf{x}, T) + L(\mathbf{x}, T) + R(\mathbf{x}, T) \right\}$$

subject to

$$\partial_t \begin{pmatrix} \mathbf{z}(\mathbf{x}, t) \\ \ell(\mathbf{x}, t) \\ L(\mathbf{x}, t) \\ R(\mathbf{x}, t) \end{pmatrix} = \begin{pmatrix} -\nabla \Phi(\mathbf{z}(\mathbf{x}, t), t; \boldsymbol{\theta}) \\ -\text{tr}(\nabla^2 \Phi(\mathbf{z}(\mathbf{x}, t), t; \boldsymbol{\theta})) \\ \frac{1}{2} \|\nabla \Phi(\mathbf{z}(\mathbf{x}, t), t; \boldsymbol{\theta})\|^2 \\ |\partial_t \Phi(\mathbf{z}(\mathbf{x}, t), t; \boldsymbol{\theta}) - \frac{1}{2} \|\nabla \Phi(\mathbf{z}(\mathbf{x}, t), t; \boldsymbol{\theta})\|^2| \end{pmatrix}$$

with initial conditions

$$\mathbf{z}(\mathbf{x}, 0) = \mathbf{x} \quad \text{and} \quad \ell(\mathbf{x}, 0) = L(\mathbf{x}, 0) = R(\mathbf{x}, 0) = 0$$

Trace Integration

Uniqueness of OT-Flow:

How we calculate $\ell(\mathbf{x}, T) = \int_0^T -\text{tr} (\nabla^2 \Phi(\mathbf{z}(\mathbf{x}, t), t; \boldsymbol{\theta})) \, dt$

OT-Flow:

- Trace
 - ▶ Exact Trace Computation
- Time Integration
 - ▶ Discretize-then-optimize (DTO)^{11,12}

Comparatively, state-of-the-art:

- Trace (during training)
 - ▶ Hutchinson's Estimator
- Time Integration
 - ▶ Optimize-then-discretize (OTD)^{11,12}

¹¹Gholami, Keutzer, and Biros. "ANODE: Unconditionally Accurate Memory-Efficient Gradients for...". 2019.

¹²Onken and Ruthotto. "Discretize-Optimize vs. Optimize-Discretize for Time-Series...". 2020.

Improving the Trace Computation

Competitive in Time Complexity

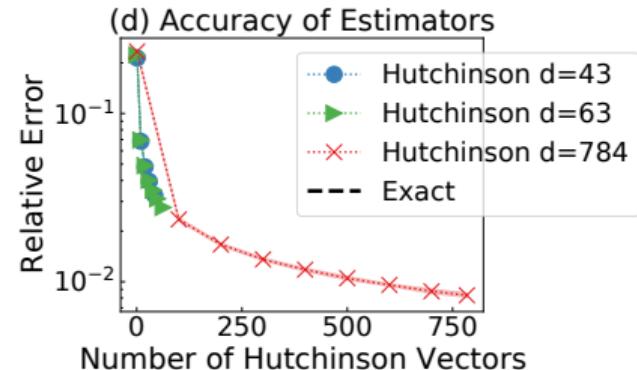
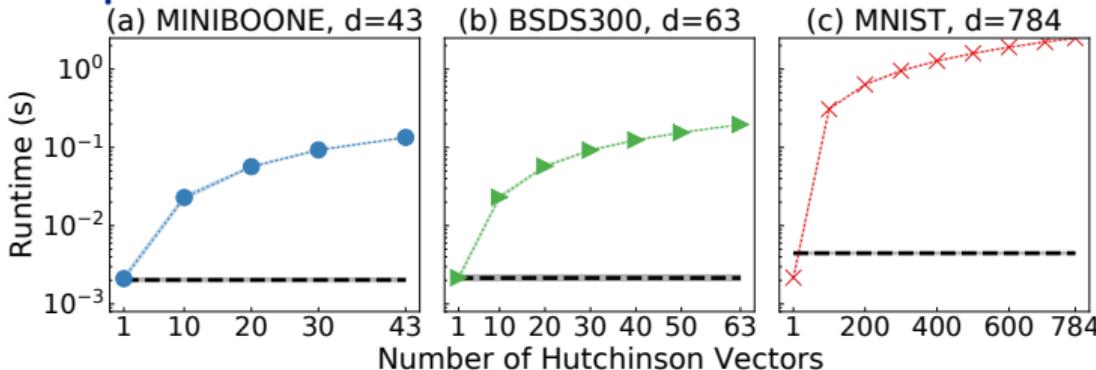
General Trace Computation: $\mathcal{O}(d^2)$ FLOPS

Trace Estimators: $\mathcal{O}(d)$ FLOPS

Our Exact Trace in OT-Flow $\mathcal{O}(d)$ FLOPS

} assumes model's hidden dimension is fixed

Competitive in Runtime



Improving the Trace Computation

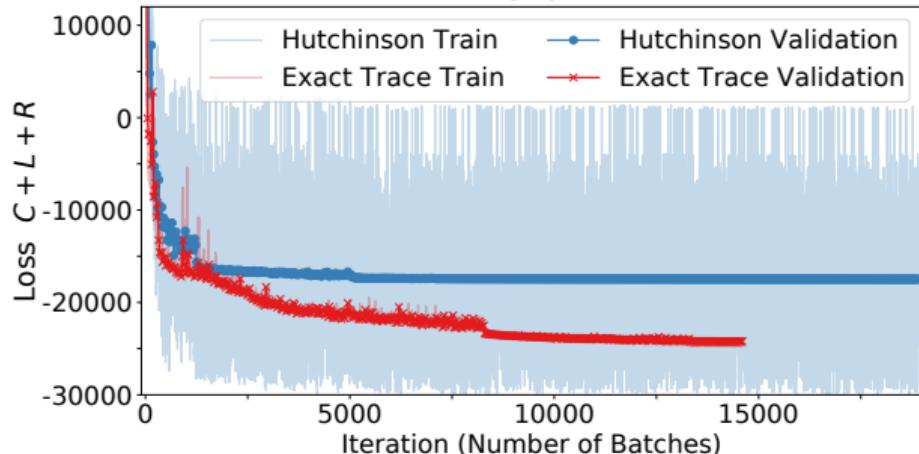
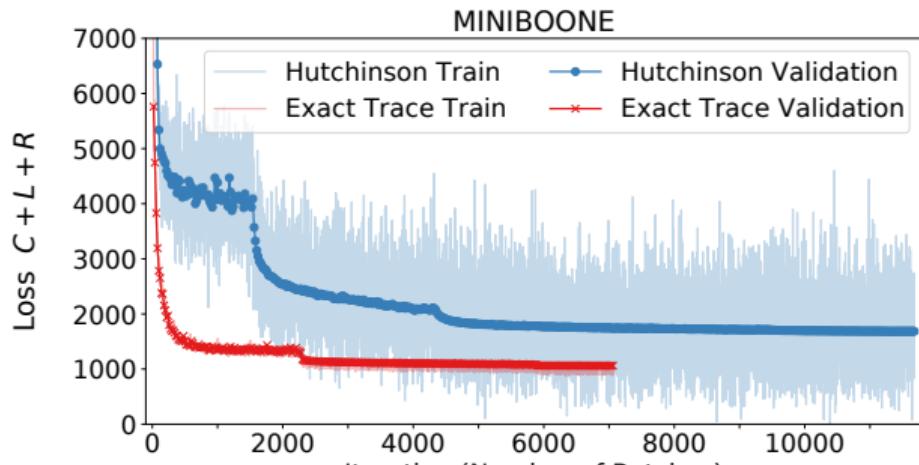
Improved Convergence

Exact Trace \Rightarrow Improved Convergence

Compare OT-Flow (using exact trace)
against a replicate model using
Hutchinson's trace estimator

OT-Flow converges

- 1) in fewer iterations
- 2) with less training variance



Exact Trace Computation

Our model

Neural Network

$$\Phi(\mathbf{s}; \boldsymbol{\theta}) = \mathbf{w}^\top N(\mathbf{s}; \boldsymbol{\theta}_N) + \frac{1}{2} \mathbf{s}^\top (\mathbf{A}^\top \mathbf{A}) \mathbf{s} + \mathbf{b}^\top \mathbf{s} + c,$$

where $\boldsymbol{\theta} = (\mathbf{w}, \boldsymbol{\theta}_N, \mathbf{A}, \mathbf{b}, c)$

Gradient

$$\nabla_{\mathbf{s}} \Phi(\mathbf{s}; \boldsymbol{\theta}) = \nabla_{\mathbf{s}} N(\mathbf{s}; \boldsymbol{\theta}_N) \mathbf{w} + (\mathbf{A}^\top \mathbf{A}) \mathbf{s} + \mathbf{b}$$

where

- space-time inputs $\mathbf{s} = (\mathbf{x}, t) \in \mathbb{R}^{d+1}$
- neural network $N(\mathbf{s}; \boldsymbol{\theta}_N): \mathbb{R}^{d+1} \rightarrow \mathbb{R}^m$ (we choose ResNet)
- $\boldsymbol{\theta}$ consists of all the trainable weights:

$$\mathbf{w} \in \mathbb{R}^m, \boldsymbol{\theta}_N \in \mathbb{R}^p, \mathbf{A} \in \mathbb{R}^{r \times (d+1)}, \mathbf{b} \in \mathbb{R}^{d+1}, c \in \mathbb{R} \quad \text{where } r = \min(10, d)$$

Exact Trace Computation

Analytic Gradient and Trace Computation

N is an $(M + 1)$ -layer ResNet

Forward propagation

Compute $N(\mathbf{s}; \boldsymbol{\theta}_N) = \mathbf{u}_M$.

where

$$\mathbf{u}_0 = \sigma(\mathbf{K}_0 \mathbf{s} + \mathbf{b}_0)$$

$$\mathbf{u}_1 = \mathbf{u}_0 + h\sigma(\mathbf{K}_1 \mathbf{u}_0 + \mathbf{b}_1)$$

$$\vdots \quad \vdots$$

$$\mathbf{u}_M = \mathbf{u}_{M-1} + h\sigma(\mathbf{K}_M \mathbf{u}_{M-1} + \mathbf{b}_M)$$

- fixed step size $h > 0$
- ResNet weights $\boldsymbol{\theta}_N$ are
 - ▶ $\mathbf{K}_0 \in \mathbb{R}^{m \times (d+1)}$
 - ▶ $\mathbf{K}_1, \dots, \mathbf{K}_M \in \mathbb{R}^{m \times m}$
 - ▶ $\mathbf{b}_0, \dots, \mathbf{b}_M \in \mathbb{R}^m$
- $\sigma(\mathbf{x}) = \log(\exp(\mathbf{x}) + \exp(-\mathbf{x}))$
 - ▶ the antiderivative of hyperbolic tangent
 - ▶ so, $\sigma'(\mathbf{x}) = \tanh(\mathbf{x})$

Exact Trace Computation

Analytic Gradient Computation

N is an $(M + 1)$ -layer ResNet

Forward propagation

Compute $N(\mathbf{s}; \boldsymbol{\theta}_N) = \mathbf{u}_M$.

$$\mathbf{u}_0 = \sigma(\mathbf{K}_0 \mathbf{s} + \mathbf{b}_0)$$

$$\mathbf{u}_1 = \mathbf{u}_0 + h\sigma(\mathbf{K}_1 \mathbf{u}_0 + \mathbf{b}_1)$$

$$\vdots \quad \vdots$$

$$\mathbf{u}_M = \mathbf{u}_{M-1} + h\sigma(\mathbf{K}_M \mathbf{u}_{M-1} + \mathbf{b}_M)$$

Backpropagation (chain rule)

Compute $\nabla_{\mathbf{s}} N(\mathbf{s}; \boldsymbol{\theta}_N) \mathbf{w} = \mathbf{z}_0$ analytically

Laplacian

Compute

$$\text{tr} (\nabla^2 \Phi(\mathbf{s}; \boldsymbol{\theta})) = \text{tr} (\mathbf{E}^\top (\nabla_{\mathbf{s}}^2 (N(\mathbf{s}; \boldsymbol{\theta}_N) \mathbf{w}) + \mathbf{A}^\top \mathbf{A}) \mathbf{E})$$

for $\mathbf{E} = \text{eye}(d+1, d)$

with cost $\mathcal{O}(m^2 \cdot d)$ FLOPS.

(details in paper)

Other OT approaches in CNFs

Model	Formulation					Training Implementation			Inference
	ODEs (2)	Φ	L	R	$\ \nabla \mathbf{v}\ _F^2$	Solver	DTO/OTD	Trace	Trace
FFJORD ¹³	✓	✗	✗	✗	✗	RK(4)5	OTD	Hutch w/ Rad	AD exact
RNODE ¹⁴	✓	✗	✓	✗	✓	RK 4	OTD	Hutch w/ Rad	AD exact
M-A Flows ¹⁵	✓	✓	✗	✗	✗	RK 4	DTO	Hutch w/ Gauss	
PFGs ¹⁶	✓	✓	✗	✓	✗	RK 1	DTO		AD exact
OT-Flow	✓	✓	✓	✓	✗	RK 4	DTO		efficient exact

RK: Runge-Kutta, OTD: optimize-then-discretize, DTO: discretize-then-optimize, AD: automatic differentiation, Hutch: Hutchinson's trace estimator where ϵ from Rademacher or Gaussian distribution

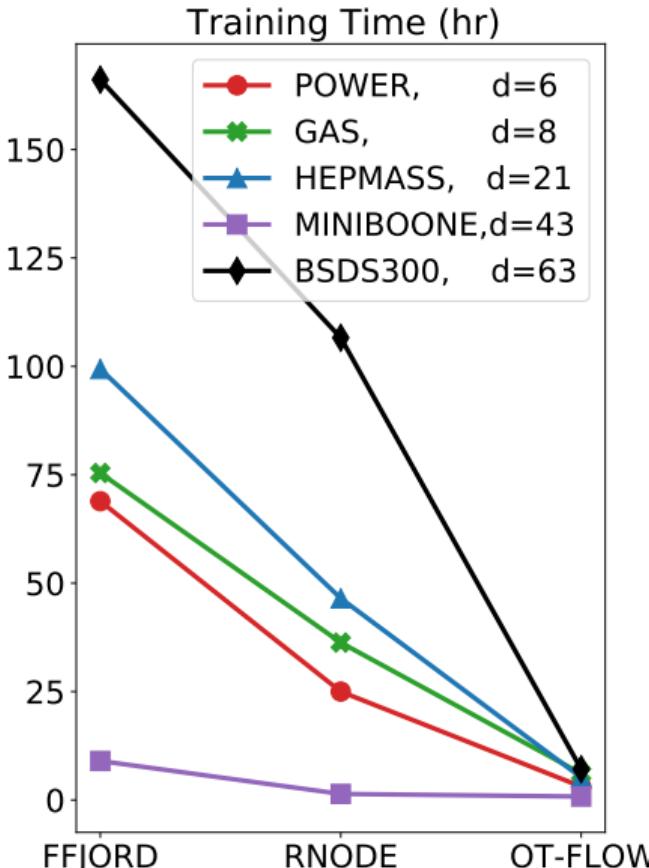
¹³Grathwohl et al. "FFJORD: Free-form Continuous Dynamics for Scalable Reversible...". 2019.

¹⁴Finlay et al. "How to Train your Neural ODE: the World of Jacobian and Kinetic Regularization". 2020.

¹⁵Zhang and Wang. "Monge-Ampère Flow for Generative Modeling". 2018.

¹⁶Yang and Karniadakis. "Potential Flow Generator With L_2 Optimal Transport Regularity...". 2020.

Fast Training

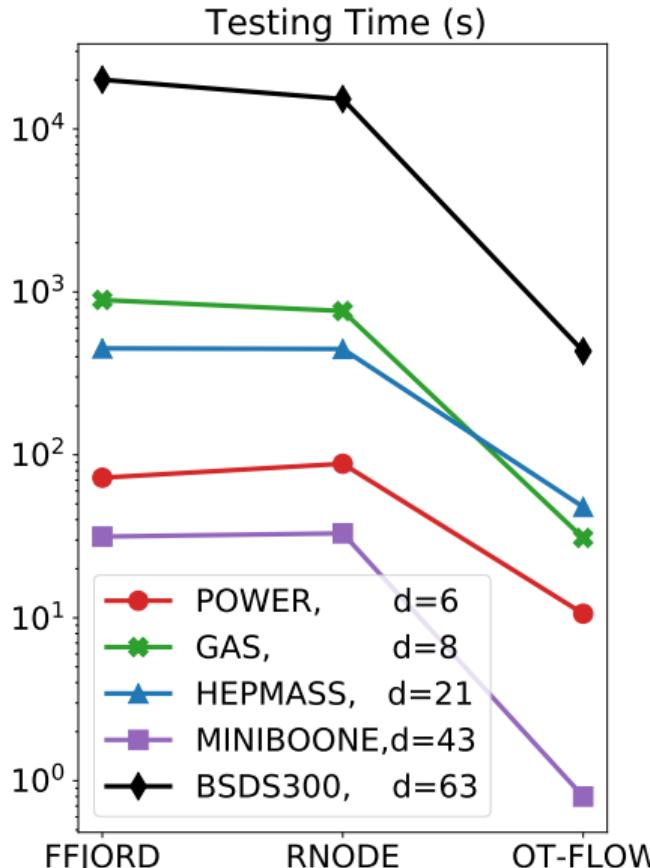


OT-Flow has 8x training speed-up on average

Reasons

- OT-inspired regularization leads to straight trajectories that are inexpensive to integrate.
- Exact trace computation
 - ▶ Competitive in time
 - ▶ Better in convergence
- The potential flows approach results in fewer weights and a smaller model.

Fast Inference

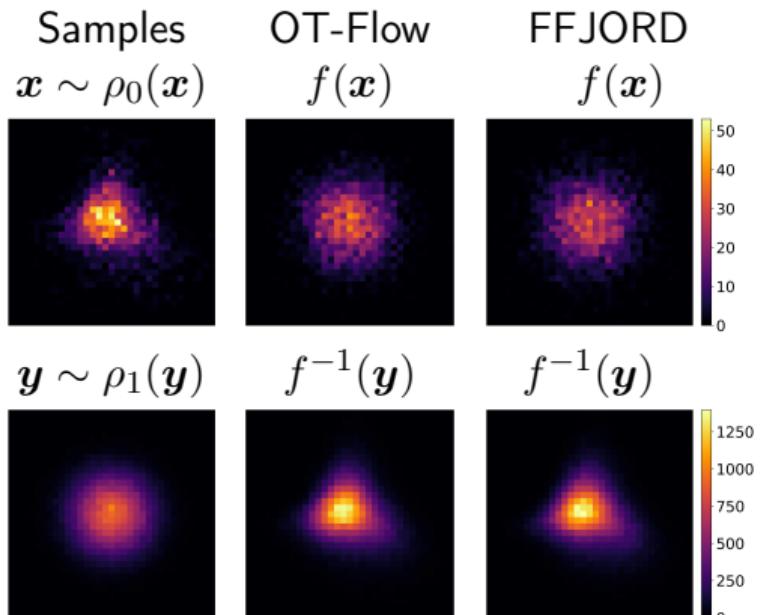


OT-Flow has 24x testing speed-up on average

Reasons

- Inference uses exact trace (no estimates)
 - ▶ State-of-the-art approaches use AD to obtain exact trace with $\mathcal{O}(d^2)$
 - ▶ Meanwhile, our exact trace is $\mathcal{O}(d)$

More Results



Two of the 43 dimensions in the MINIBOONE CNF



MNIST synthetic generation. Original images boxed in red.

Conclusions

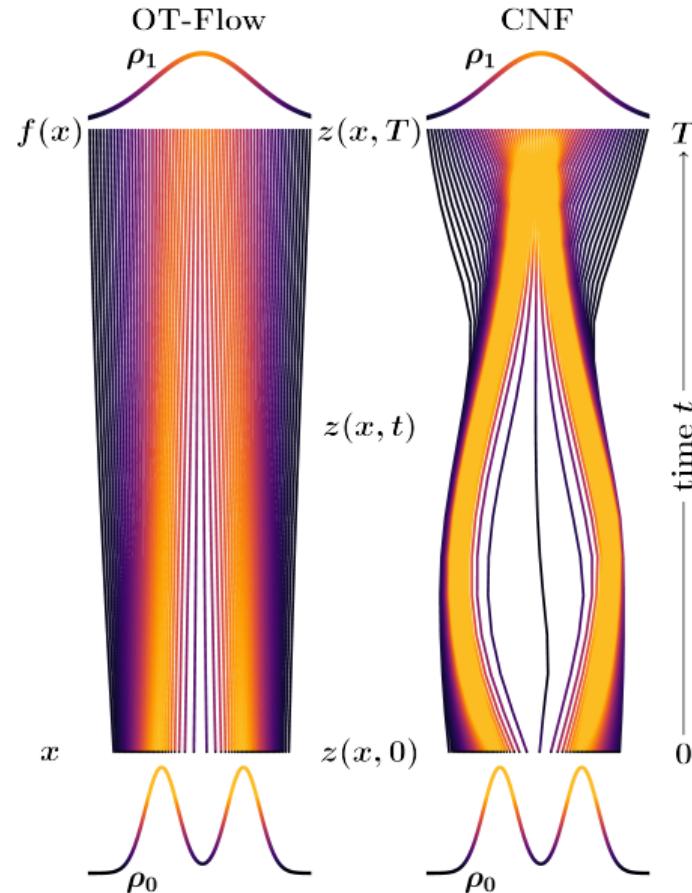
Formulation

- CNF + OT \Rightarrow well-posed
- HJB regularizer reduces training costs

Implementation

- Discretize-then-optimize + Runge-Kutta 4
 \Rightarrow efficient ODE solve
- Efficient exact trace improves CNF training

 Public Code
github.com/EmoryMLIP/OT-Flow



References I

- Bellman, Richard (1957). *Dynamic Programming*. Princeton University Press, Princeton, N. J.
- Chen, Tian Qi et al. (2018). “Neural Ordinary Differential Equations”. In: *Advances in Neural Information Processing Systems*, pp. 6571–6583.
- Finlay, Chris et al. (2020). “How to Train your Neural ODE: the World of Jacobian and Kinetic Regularization”. In: *International Conference on Machine Learning (ICML)*.
- Gholami, Amir, Kurt Keutzer, and George Biros (2019). “ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs”. In: *International Joint Conferences on Artificial Intelligence (IJCAI)*.
- Grathwohl, Will et al. (2019). “FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models”. In: *International Conference on Learning Representations (ICLR)*.
- Hutchinson, Michael F (1990). “A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines”. In: *Communications in Statistics-Simulation and Computation* 19.2, pp. 433–450.

References II

- Onken, Derek and Lars Ruthotto (2020). "Discretize-Optimize vs. Optimize-Discretize for Time-Series Regression and Continuous Normalizing Flows". In: *arXiv:2005.13420*.
- Papamakarios, George et al. (2019). "Normalizing Flows for Probabilistic Modeling and Inference". In: *arXiv preprint arXiv:1912.02762*.
- Pontryagin, L. S. et al. (1962). *The Mathematical Theory of Optimal Processes*. Translated by K. N. Trirogoff; edited by L. W. Neustadt. Interscience Publishers John Wiley & Sons, Inc. New York-London, pp. viii+360.
- Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning (ICML)*, pp. 1530–1538.
- Yang, L. and G. E. Karniadakis (2020). "Potential Flow Generator With L_2 Optimal Transport Regularity for Generative Models". In: *IEEE Transactions on Neural Networks and Learning Systems*.
- Zhang, Linfeng, Lei Wang, et al. (2018). "Monge-Ampère Flow for Generative Modeling". In: *arXiv preprint arXiv:1809.10188*.