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

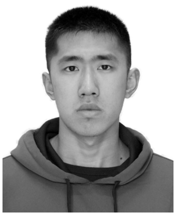
AAAI 2021

Derek Onken
Emory University
derekonken.com

Collaborators and Acknowledgments

UCLA

Samy Wu Fung
UCLA



Xingjian Li
Emory

Lars Ruthotto
Emory

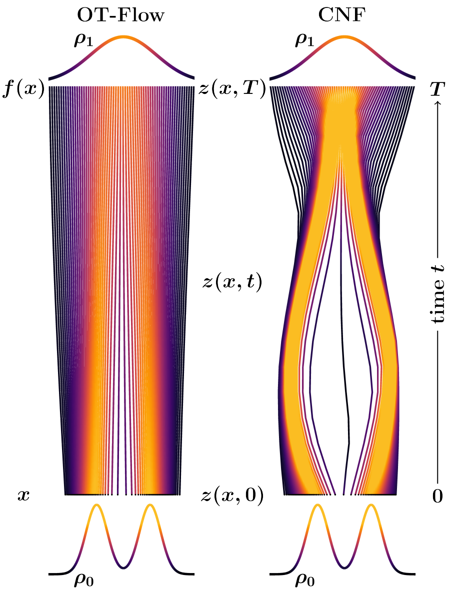
EMORY UNIVERSITY

Funding: NSF, BSF, ONR, AFOSR, UnitedHealth Group

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

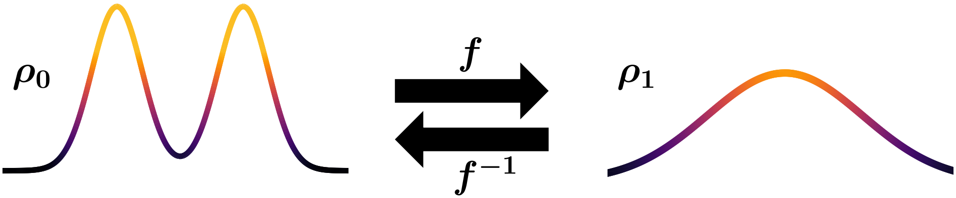
Overview

- **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 $f\colon \mathbb{R}^d \to \mathbb{R}^d$ between an arbitrary probability distribution and a standard normal distribution with respective densities ρ_0 and ρ_1



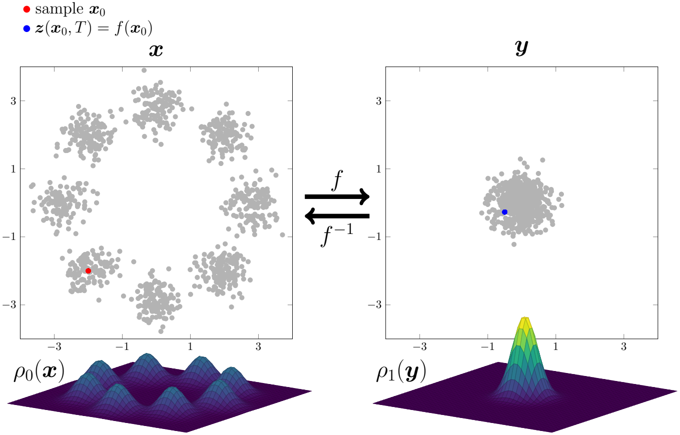
By the change of variables formula, the flow satisfies

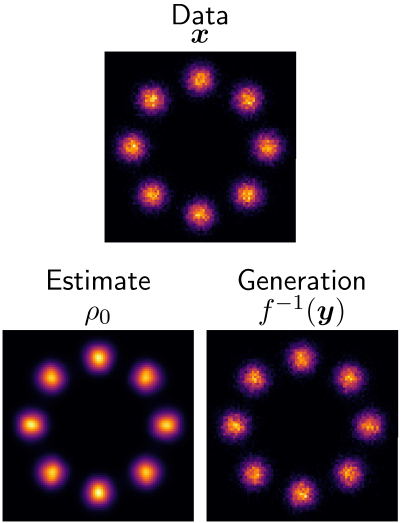
$$\log \rho_0(\boldsymbol{x}) = \log \rho_1(f(\boldsymbol{x})) + \log |\det \nabla f(\boldsymbol{x})| \quad \text{for all} \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

[1]Rezende and Mohamed. "Variational Inference with Normalizing Flows". 2015.
[2]Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". 2019.

Two-Dimensional Example

Gaussian Mixture Problem





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 CNF[4]

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

where

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

[3] Chen et al. "Neural Ordinary Differential Equations". 2018.
[4] Grathwohl et al. "FFJORD: Free-form Continuous Dynamics for Scalable Reversible . . . ". 2019.

CNF Optimization Problem

For expected negative log-likelihood[5,6]

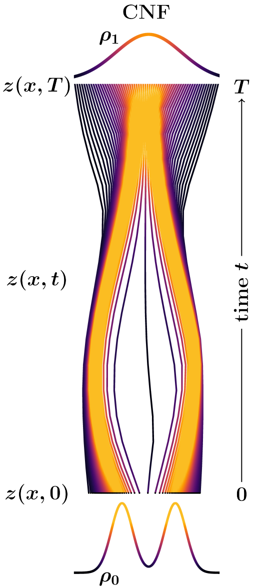
$$C(\boldsymbol{x},T) := \frac{1}{2}\|\boldsymbol{z}(\boldsymbol{x},T)\|^2 - \ell(\boldsymbol{x},T) + \frac{d}{2}\log(2\pi),$$

we optimize

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

subject to

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



[5] Rezende and Mohamed. "Variational Inference with Normalizing Flows". 2015.
[6] Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". 2019.

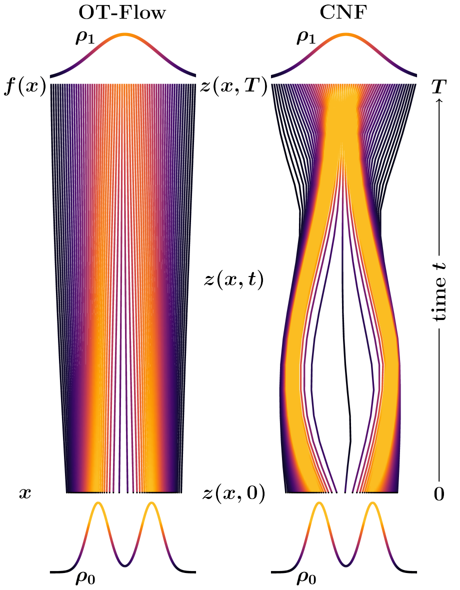
High Costs of CNFs

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 estimator[7]

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

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



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

Straight Trajectories

Include some optimal transport (OT)
$\Rightarrow$ model named OT-Flow

Arclength of the trajectories

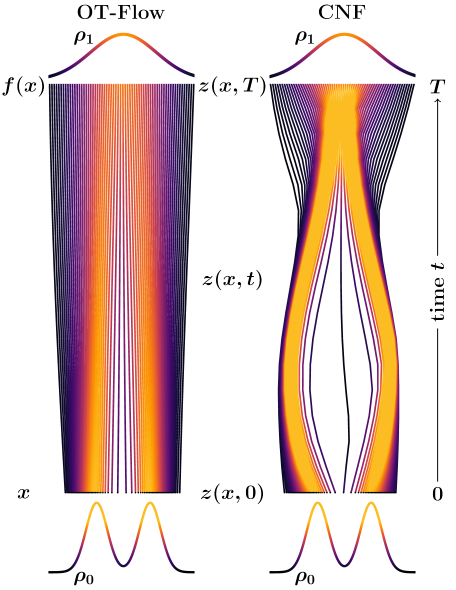
$$L(\boldsymbol{x},T) = \int_0^T \frac{1}{2} \|\mathbf{v}(\boldsymbol{z}(\boldsymbol{x},t),t;\boldsymbol{\theta})\|^2 \,\mathrm{d}t$$

We regularize the optimization problem

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

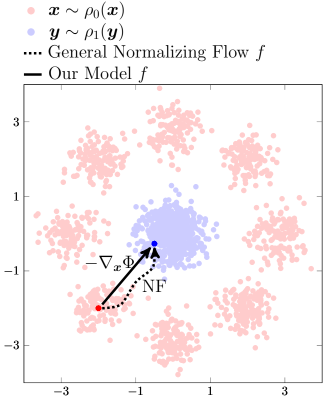
subject to (2).

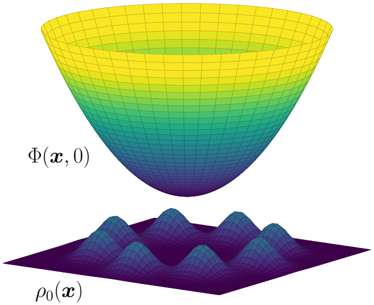
Now, a unique mapping exists.



Potential Function Φ

Apply the Pontryagin maximum principle[8] to (3)





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

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

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

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

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

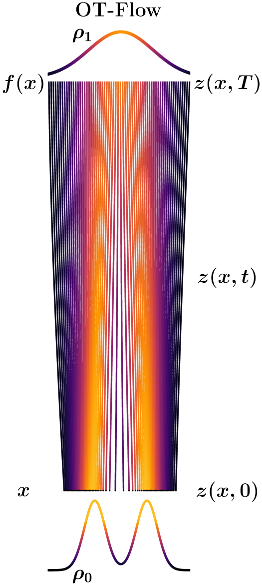
HJB Equations

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

Potential Φ satisfies the Hamilton-Jacobi-Bellman (HJB) equations[9]

$$
\begin{aligned}
-\partial_t \Phi(\boldsymbol{z}(\boldsymbol{x},t),t) &= -\frac{1}{2}\|\nabla\Phi(\boldsymbol{z}(\boldsymbol{x},t),t)\|^2, \quad 0 < t < T \\
\Phi(\boldsymbol{z}(\boldsymbol{x},T),T) &= 1 + \log\left(\rho_0(\boldsymbol{x})\right) - \log\left(\rho_1(\boldsymbol{z}(\boldsymbol{x},T))\right) - \ell(\boldsymbol{x},T)
\end{aligned}
$$

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



[9] Bellman. *Dynamic Programming*. 1957.

HJB Regularizer R

Penalize deviations from the HJB equation

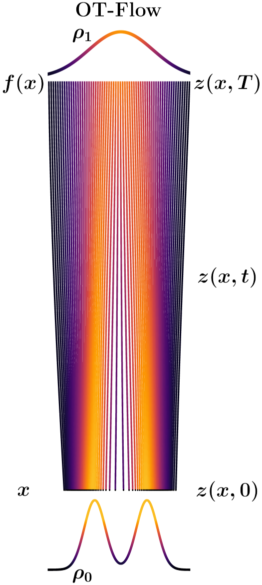
We add another regularizer, so the optimization problem is

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

subject to (2).

The HJB regularizer[10] is computed as

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



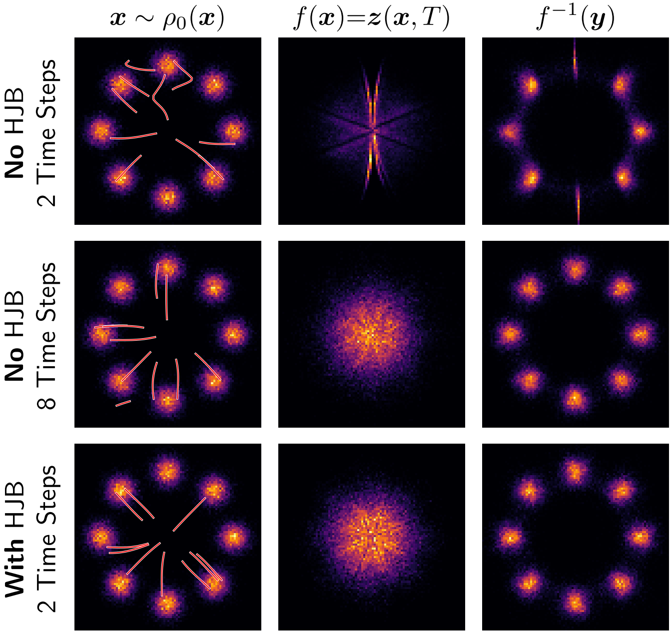
[10] Yang and Karniadakis. "Potential Flow Generator With L_2 Optimal Transport Regularity . . . ". 2020.

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.

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

subject to

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

with initial conditions

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

Trace Integration

Uniqueness of OT-Flow:

How we calculate $\ell(\boldsymbol{x}, T) = \int_0^T -\operatorname{tr}\left(\nabla^2 \Phi(\boldsymbol{z}(\boldsymbol{x}, t), t; \boldsymbol{\theta})\right) \mathrm{d}t$

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]

[11] Gholami, Keutzer, and Biros. "ANODE: Unconditionally Accurate Memory-Efficient Gradients for . . .". 2019.
[12] Onken and Ruthotto. "Discretize-Optimize vs. Optimize-Discretize for Time-Series . . .". 2020.

Improving the Trace Computation

Competitive in Time Complexity

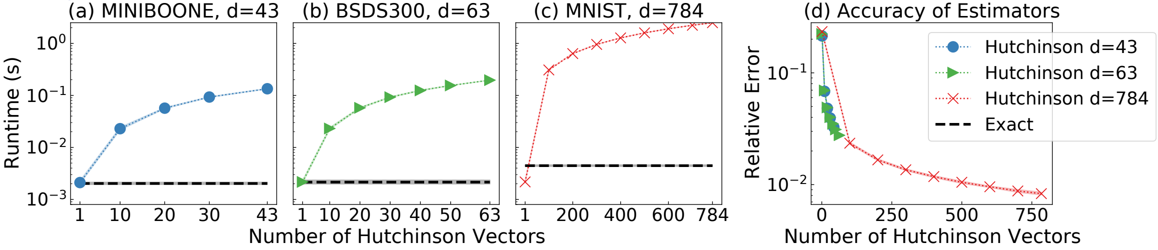
General Trace Computation: $\mathcal{O}(d^2)$ FLOPS

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

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

$\Big\}$ assumes model's hidden dimension is fixed

Competitive in Runtime



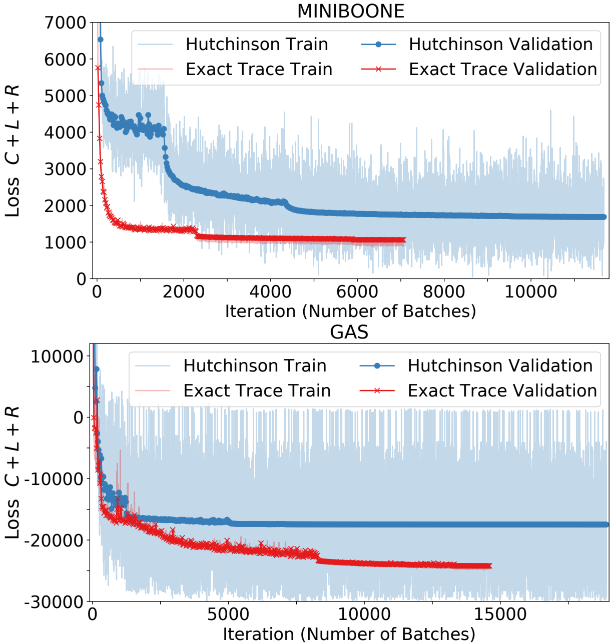
Improving the Trace Computation

Improved Convergence

Exact Trace ⇒ 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

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

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

Gradient

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

where

- space-time inputs $\boldsymbol{s} = (\boldsymbol{x}, t) \in \mathbb{R}^{d+1}$
- neural network $N(\boldsymbol{s};\boldsymbol{\theta}_N)\colon \mathbb{R}^{d+1} \to \mathbb{R}^m$ (we choose ResNet)
- $\boldsymbol{\theta}$ consists of all the trainable weights:
  $\boldsymbol{w} \in \mathbb{R}^m$, $\boldsymbol{\theta}_N \in \mathbb{R}^p$, $\boldsymbol{A} \in \mathbb{R}^{r\times(d+1)}$, $\boldsymbol{b} \in \mathbb{R}^{d+1}$, $c \in \mathbb{R}$ where $r = \min(10, d)$

Exact Trace Computation

Analytic Gradient and Trace Computation

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

Forward propagation

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

$$\begin{aligned}
\boldsymbol{u}_0 &= \sigma(\boldsymbol{K}_0\boldsymbol{s} + \boldsymbol{b}_0) \\
\boldsymbol{u}_1 &= \boldsymbol{u}_0 + h\sigma(\boldsymbol{K}_1\boldsymbol{u}_0 + \boldsymbol{b}_1) \\
&\vdots \qquad\qquad \vdots \\
\boldsymbol{u}_M &= \boldsymbol{u}_{M-1} + h\sigma(\boldsymbol{K}_M\boldsymbol{u}_{M-1} + \boldsymbol{b}_M)
\end{aligned}$$

where

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

Exact Trace Computation

Analytic Gradient Computation

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

Forward propagation

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

$$\begin{aligned} \boldsymbol{u}_0 &= \sigma(\boldsymbol{K}_0 \boldsymbol{s} + \boldsymbol{b}_0) \\ \boldsymbol{u}_1 &= \boldsymbol{u}_0 + h\sigma(\boldsymbol{K}_1 \boldsymbol{u}_0 + \boldsymbol{b}_1) \\ \vdots \quad & \quad \quad \vdots \\ \boldsymbol{u}_M &= \boldsymbol{u}_{M-1} + h\sigma(\boldsymbol{K}_M \boldsymbol{u}_{M-1} + \boldsymbol{b}_M) \end{aligned}$$

Backpropagation (chain rule)

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

Laplacian

Compute

$$\operatorname{tr}\left(\nabla^2 \Phi(\boldsymbol{s};\boldsymbol{\theta})\right) = \operatorname{tr}\left(\boldsymbol{E}^\top \left(\nabla_{\boldsymbol{s}}^2 (N(\boldsymbol{s};\boldsymbol{\theta}_N)\boldsymbol{w}) + \boldsymbol{A}^\top \boldsymbol{A}\right) \boldsymbol{E}\right)$$
$$\text{for} \quad \boldsymbol{E} = \texttt{eye(d+1,d)}$$

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

(details in paper)

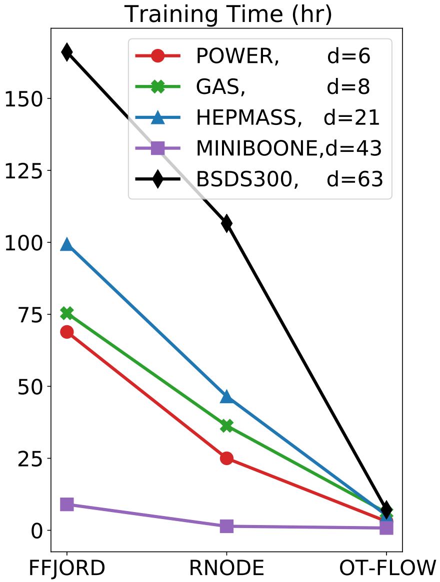
Other OT approaches in CNFs

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

[13] Grathwohl et al. "FFJORD: Free-form Continuous Dynamics for Scalable Reversible . . . ". 2019.
[14] Finlay et al. "How to Train your Neural ODE: the World of Jacobian and Kinetic Regularization". 2020.
[15] Zhang and Wang. "Monge-Ampére Flow for Generative Modeling". 2018.
[16] Yang and Karniadakis. "Potential Flow Generator With L_2 Optimal Transport Regularity . . . ". 2020.

Fast Training

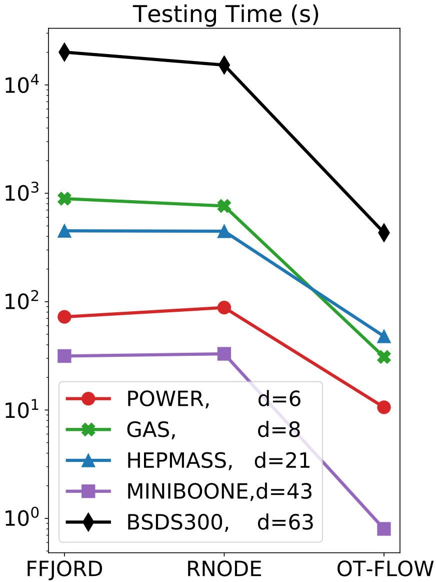


OT-Flow has 8x training speed-up on average

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

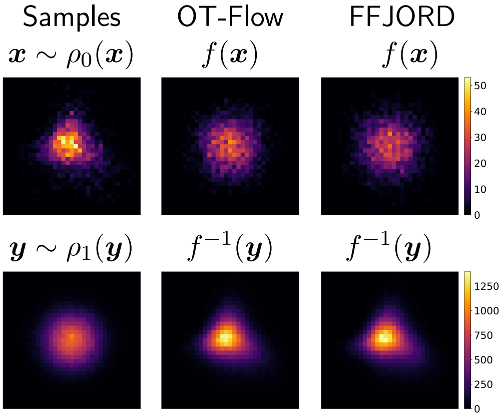


OT-Flow has 24x testing speed-up on average

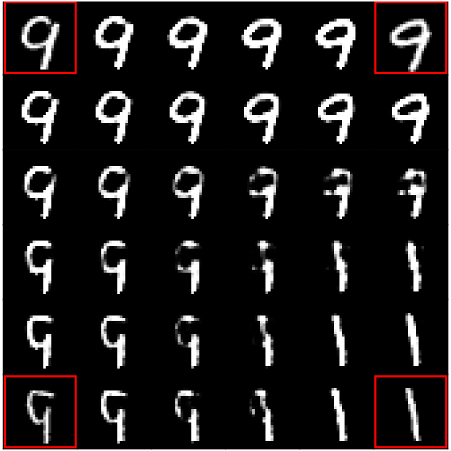
Reasons

- 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

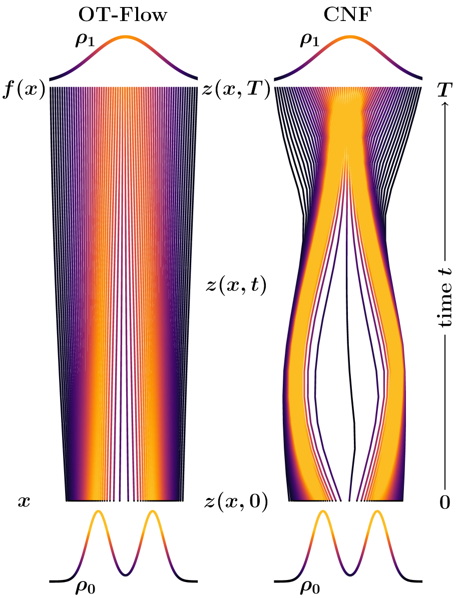
Formulation

- 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-Ampére Flow for Generative Modeling". In: *arXiv preprint arXiv:1809.10188*.