Neural Ordinary Differential Equations

Romain Hermary

September 6, 2021

Introduction

Neural Ordinary Differential Equations (Chen et al., 2018)

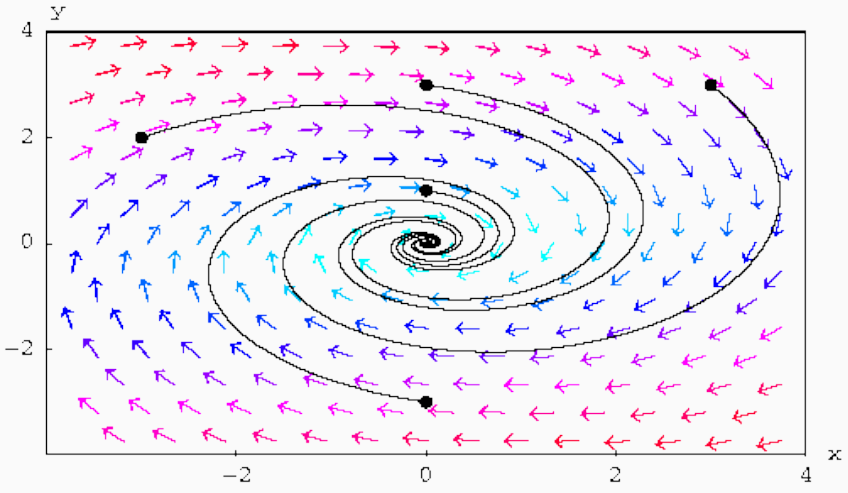
- **T. Chen**, **Yulia Rubanova**, **J. Bettencourt** and **D. Duvenaud** (University of Toronto, Vector Institute)
- Published in **Neural Information Processing Systems (NeurIPS)**, 2018
- **A***
- ∼**1.2k** citations

Context

- Supervised Learning
- Introduce a new family of deep neural network models
- From discrete sequence of hidden layers to continuous depth model
- Introduction of an ODE solver into a neural network
- Intrinsically linked to ResNet He et al. (2016)

Ordinary Differential Equations (ODEs)

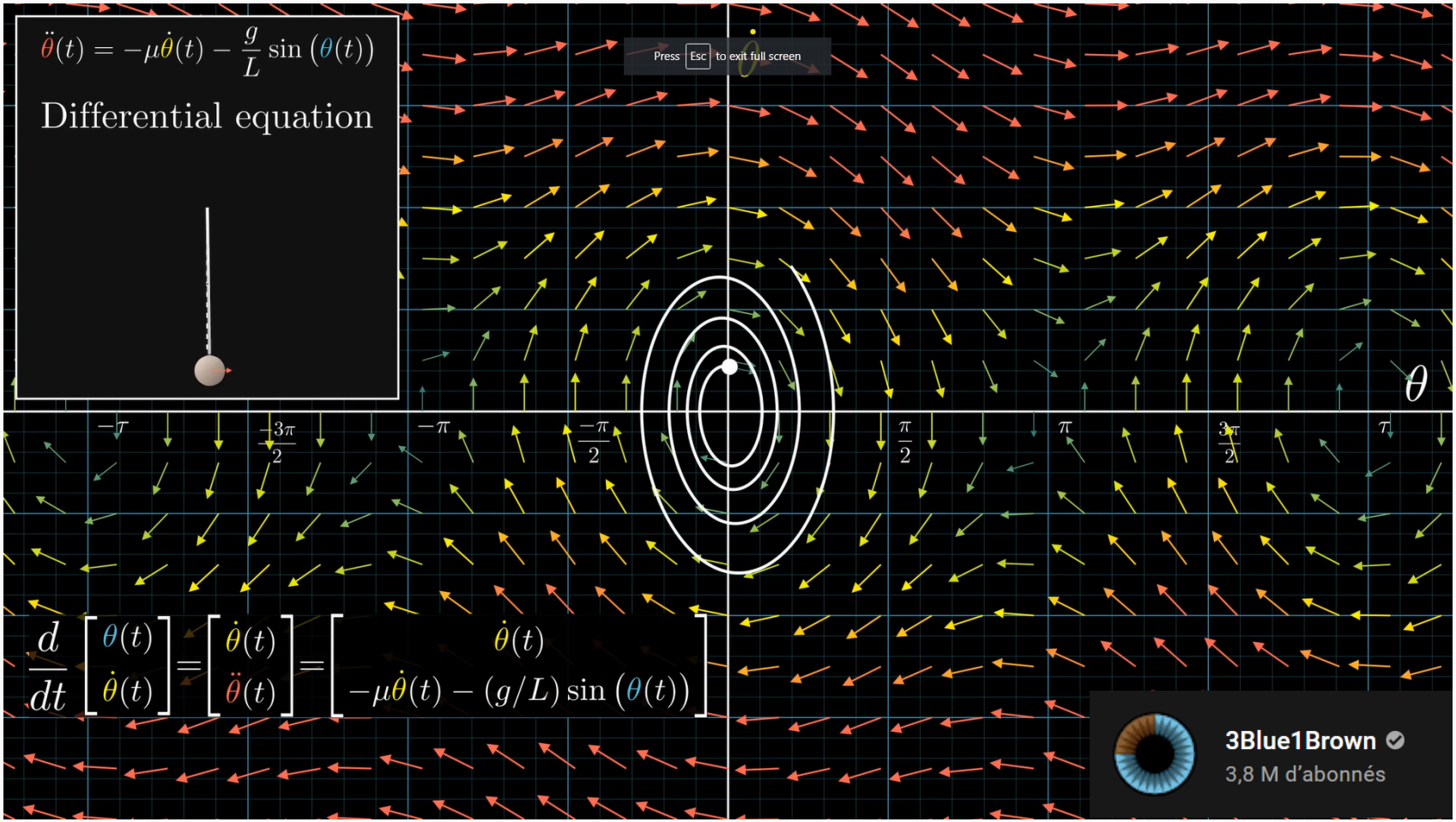
Ordinary Differential Equations: Basics



S.O.S. Math

- Describes evolution in time of some process that depends on one variable
- This change is described via a derivative

Ordinary Differential Equations: Basics

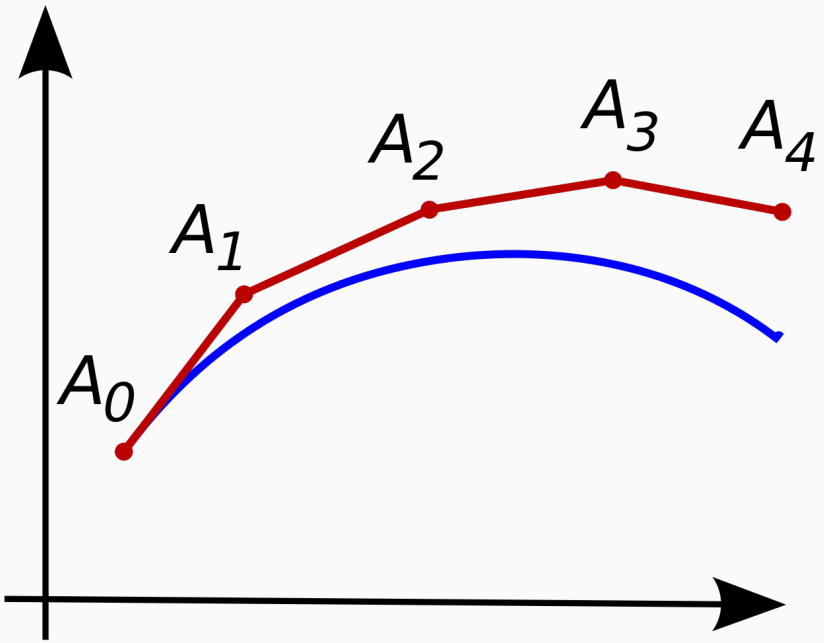


Differential equations, a tourist's guide

ODE solver

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0$$

$$y_{n+1} = y_n + hf(t_n, y_n)$$

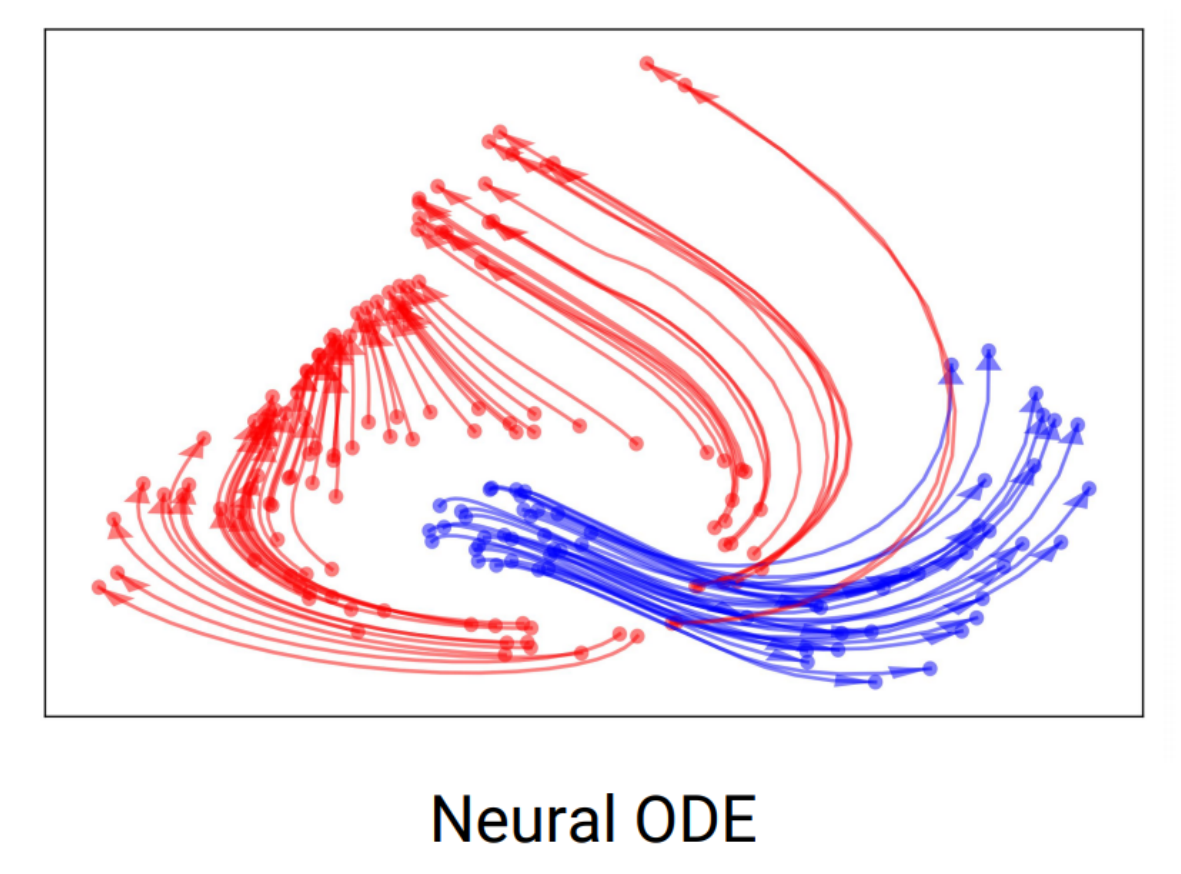


Wikipedia

- *Euler*'s method, Runge (1895); Hairer et al. (1993)... Modern ODE solvers:
- provide guarantees about the growth of approximation error
- monitor the level of error
- adapt their evaluation strategy on the fly to achieve the requested level of accuracy

Neural Ordinary Differential Equations (NODEs)

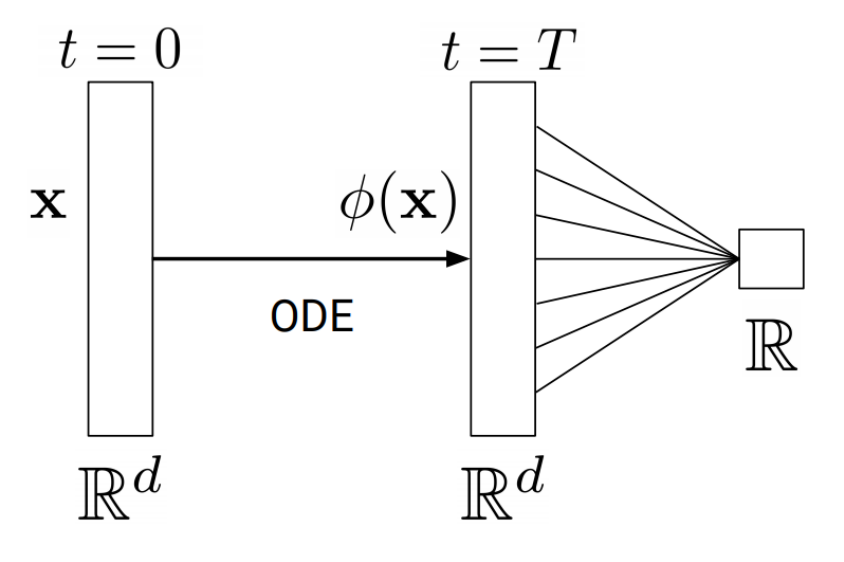
Neural ODEs (NODEs)



Dupont et al. (2019)

Neural ODEs (NODEs)

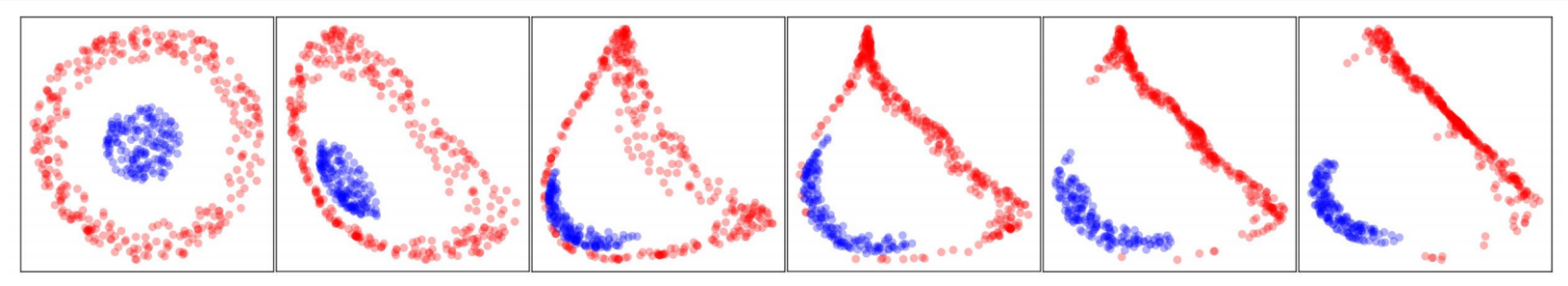
$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$



Dupont et al. (2019)

- Input as starting point, $\mathbf{h}(T)$ as an output
- f is the hidden unit dynamics
- Infinite layer like net
- Adaptative solvers adapting the step size

Neural ODEs (NODEs)



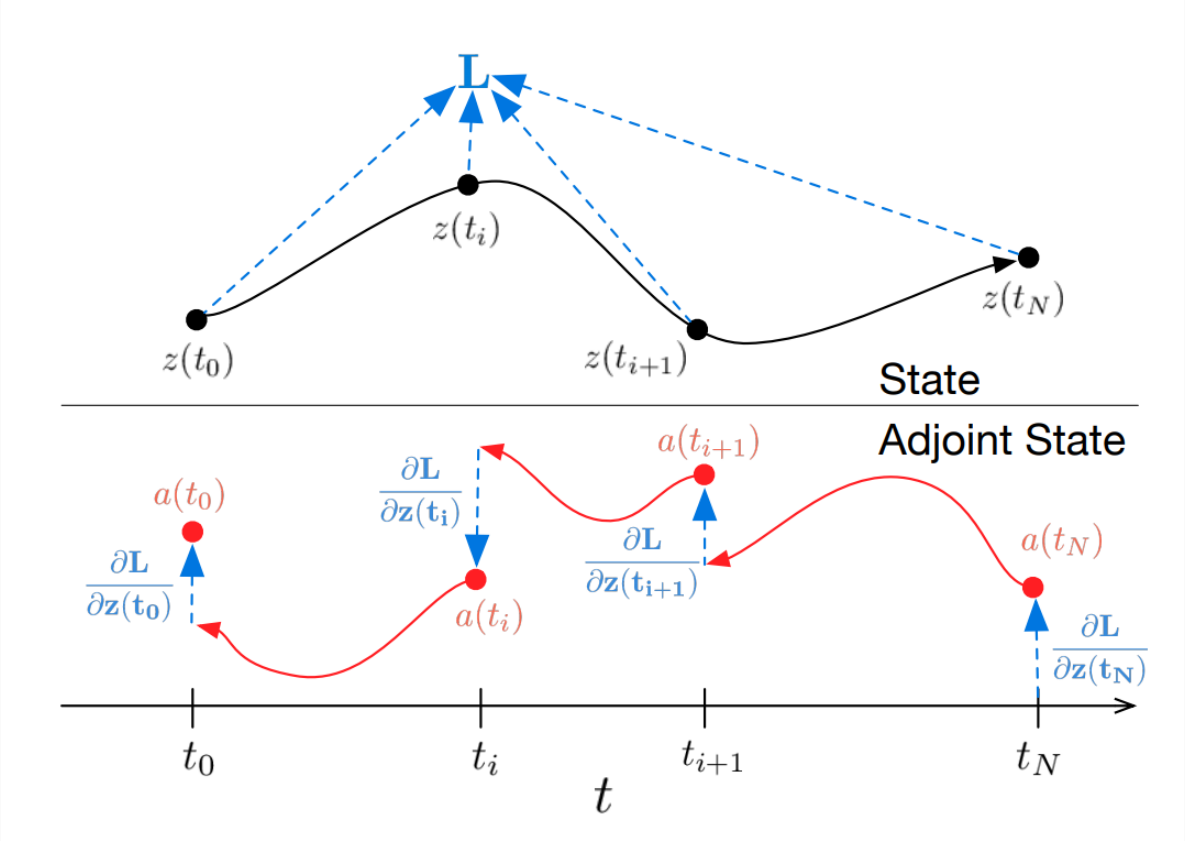
Dupont et al. (2019)

NODEs: Backpropagation

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right)$$

- We need the gradient of a loss function by the input and the dynamics parameters
- Without backpropagating through the operations of the solver (no forward storage)
- Allows to train the models with constant memory cost as a function of depth

Adjoint Sensitivity Method (Pontryagin, 1962)



- The adjoint system describes the derivative states at each point of the process backward
- Can be thought of as the instantaneous analog of the chain rule

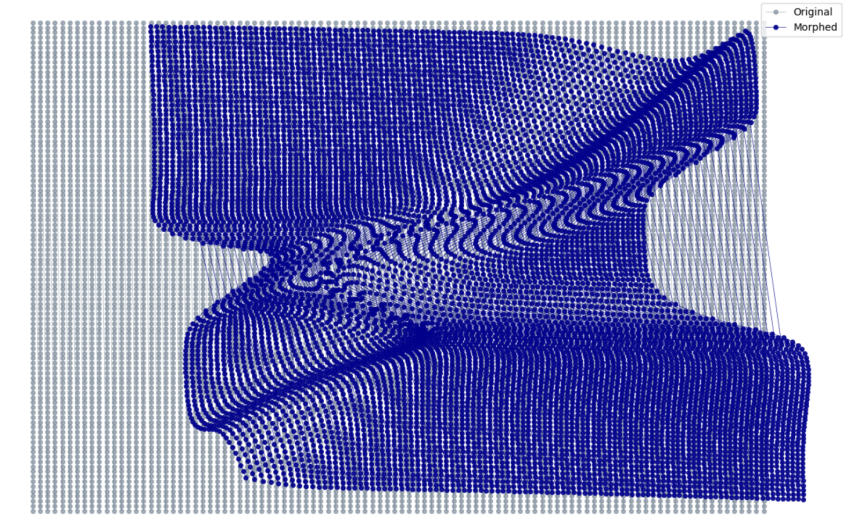
Results (Chen et al., 2018)

	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$

Performance on MNIST. †From LeCun et al. (1998)

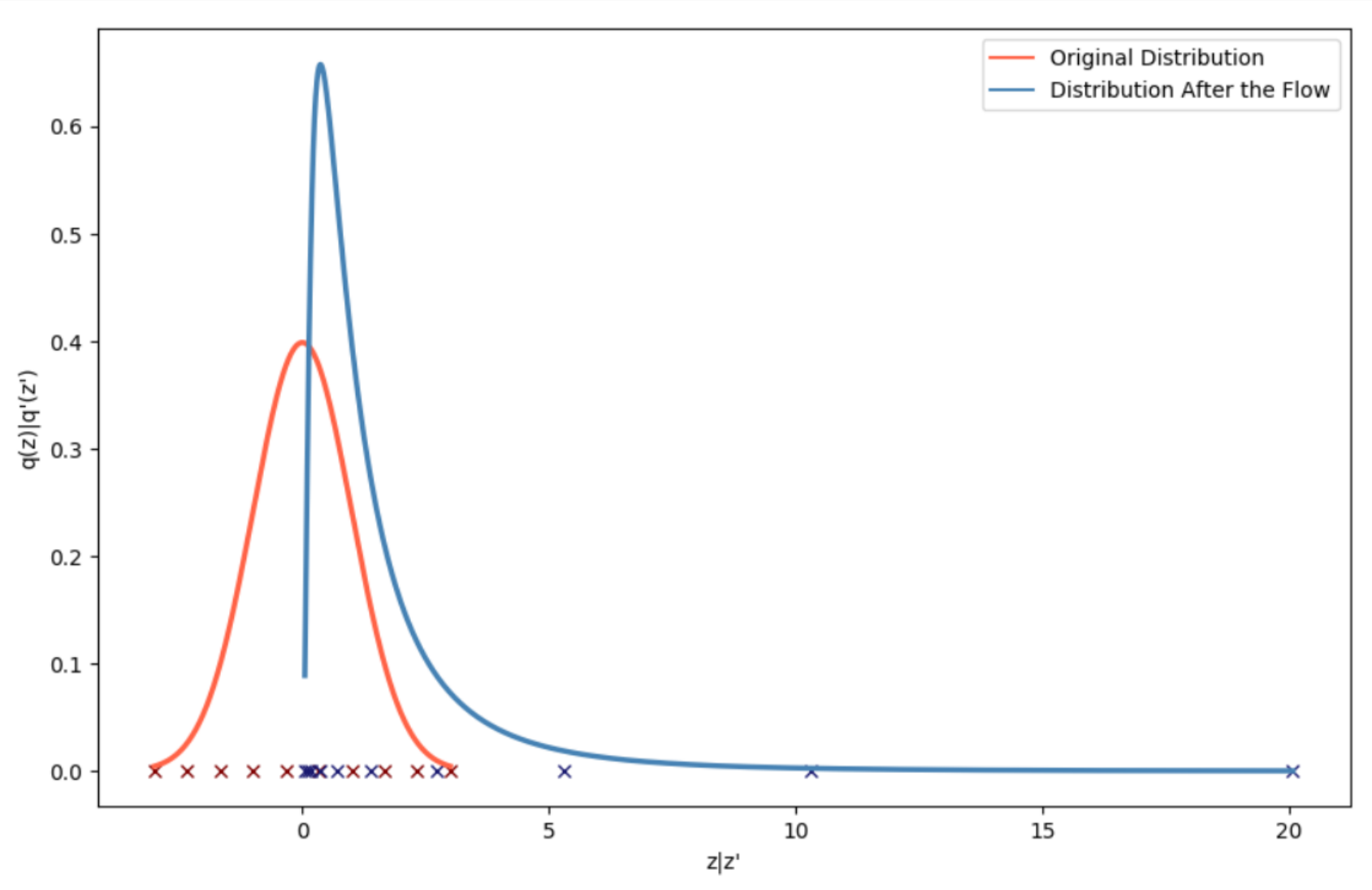
Continuous Normalizing Flows

Continuous Normalizing Flows



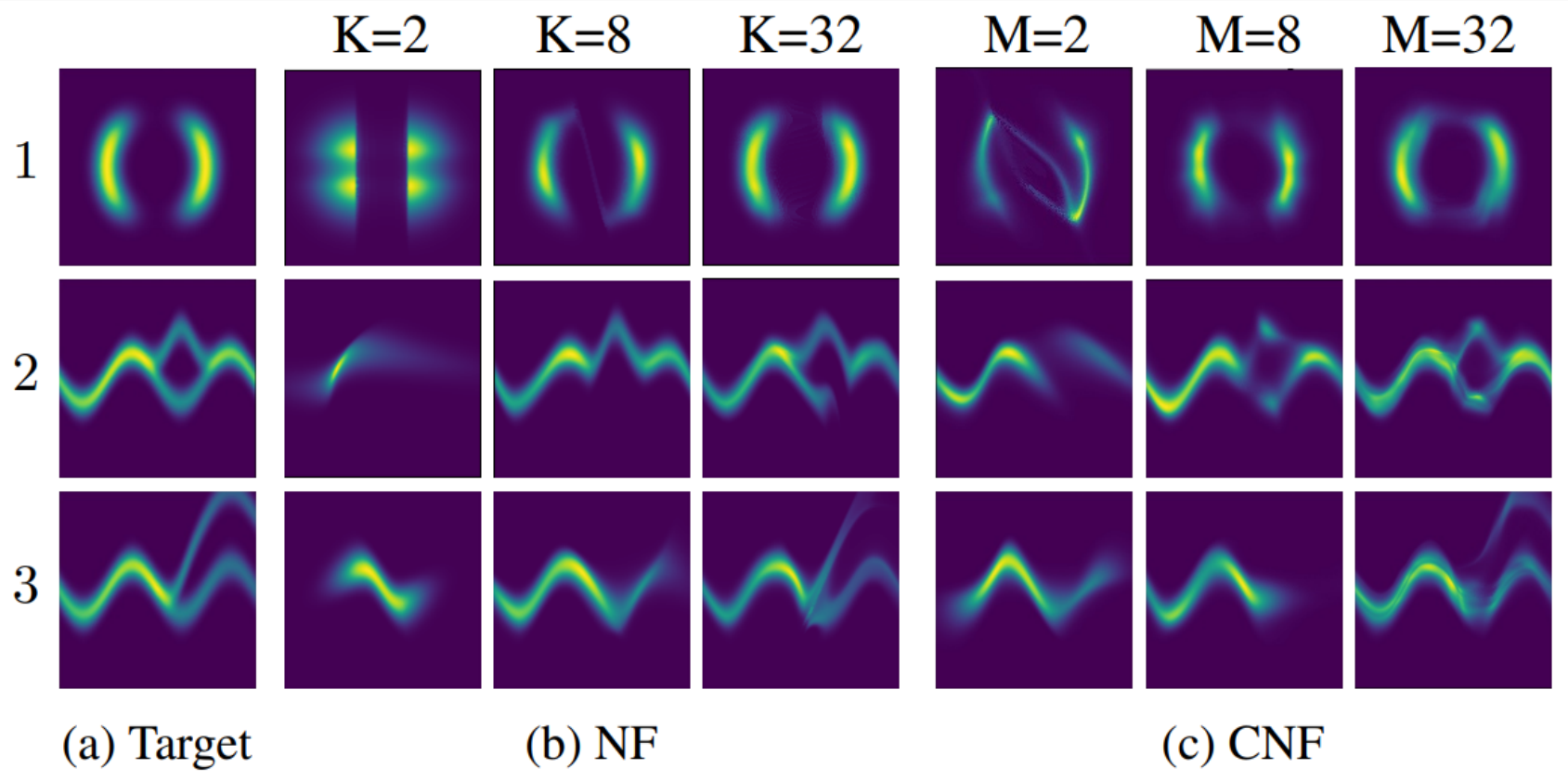
- Consists in a chain of invertible mappings that can be applied to transform an initial probability distribution into a vastly different one
- Rezende and Mohamed (2015)
- Dinh et al. (2015)
- Use the change of variables theorem to compute exact changes in probability

Continuous Normalizing Flows: Example



Medium: Normalizing Flows Are Not Magic

Results



Comparison of normalizing flows versus continuous normalizing flows. The model capacity of normalizing flows is determined by their depth (K), while continuous normalizing flows can also increase capacity by increasing width (M), making them easier to train.

Ordinary Differential Equations : advantages and downsides

- Constant memory costs
- Require fewer parameters than ResNet (He et al., 2016)
- Can not control the depth
- Can control solving time/error, trade precision for time
- Picard's existence theorem (Coddington and Levinson, 1955)

References

Chen, T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. In *NeurIPS*.

Coddington, E. and Levinson, N. (1955). Theory of ordinary differential equations.

Dinh, L., Krueger, D., and Bengio, Y. (2015). Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516.

Dupont, E., Doucet, A., and Teh, Y. (2019). Augmented neural odes. In *NeurIPS*.

Hairer, E., Nørsett, S. P., and Wanner, G. (1993). Solving ordinary differential equations i: Nonstiff problems.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition.

Pontryagin, L. (1962). Mathematical theory of optimal processes.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *ICML*.

Runge, C. (1895). Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46:167–178.

Article Analysis

The End

Any Questions?

Adjoint Sensitivity Method (Pontryagin, 1962)

$$\mathbf{a}(t) = \partial L / \partial \mathbf{z}(t)$$

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^{\mathsf{T}} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^{\mathsf{T}} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt$$

Normalizing Flow

$$\mathbf{z}_1 = f(\mathbf{z}_0) \implies \log p(\mathbf{z}_1) = \log p(\mathbf{z}_0) - \log \left| \det \frac{\partial f}{\partial \mathbf{z}_0} \right|$$

$$\mathbf{z}(t+1) = \mathbf{z}(t) + uh(w^\intercal \mathbf{z}(t) + b), \quad \log p(\mathbf{z}(t+1)) = \log p(\mathbf{z}(t)) - \log \left| 1 + u^\intercal \frac{\partial h}{\partial \mathbf{z}} \right|$$

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)} \right)$$