Neural Ordinary Differential Equations

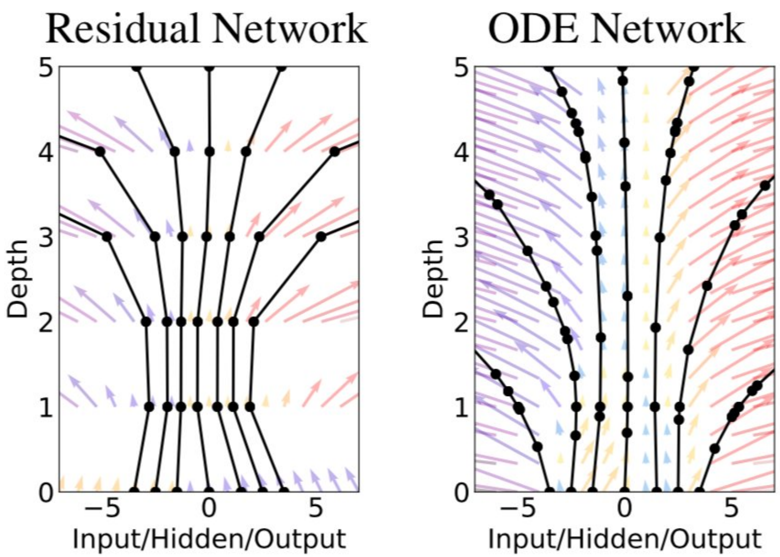
杜岳華

2019.3.15

Outline

- Introduction
- Core concept - 將離散層拉成連續層
- Neural network is an approximator for derivatives
- Solve for next state
- Replace ResNet with ODEs
- Continuous normalizing flow
- Generative latent time-series model
- Backpropagation through ODE solver
- Features and limitations
- Why ODE?

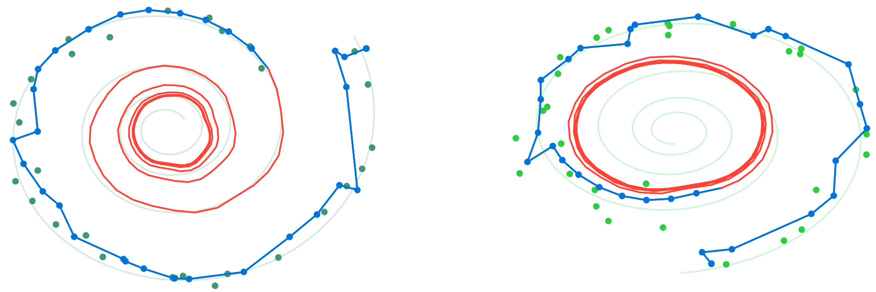
Introduction



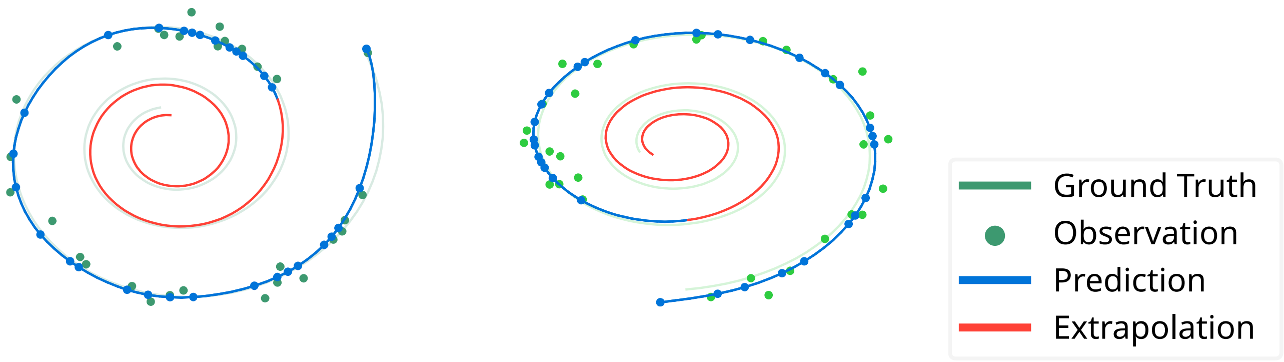
NeurIPS 2018 Best Papers

Instead of discrete sequences of layer, parameterize the continuous dynamics of hidden layer

Some results



(a) Recurrent Neural Network



(b) Latent Neural Ordinary Differential Equation

Core concept - 將離散層拉成連續層

離散版本

$$\frac{dy}{dt} = \frac{y(t+\Delta) - y(t)}{\Delta}$$

連續版本

$$\frac{dy}{dt} = \lim_{\Delta \to 0} \frac{y(t+\Delta) - y(t)}{\Delta}$$

轉化成

$$y(t+\Delta) = y(t) + \Delta \frac{dy}{dt}$$

Core concept - 將離散層拉成連續層

前回饋網路（feed-forward network）

$$h_{t+1} = f(h_t, \theta)$$

ResNet with skip connection

$$h_{t+1} = h_t + f(h_t, \theta)$$

是不是很像？

$$y(t+\Delta) = y(t) + \Delta \frac{dy}{dt}$$

Core concept - 將離散層拉成連續層

ResNet with skip connection

$$h_{t+1} = h_t + f(h_t, \theta)$$

$\Delta = 1$ 代入

$$y(t+1) = y(t) + \frac{dy}{dt}$$

Neural network is an approximator for derivatives

$$\frac{dy}{dt} = f(h_t, \theta)$$

神經網路層 f 就可以被我們拿來計算微分 $\frac{dy}{dt}$ ！

$$y(t+\Delta) = y(t) + \Delta \frac{dy}{dt}$$

$$\downarrow$$

$$y(t+\Delta) = y(t) + \Delta f(t, h(t), \theta_t)$$

Solve for next state

Layer for approximation

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

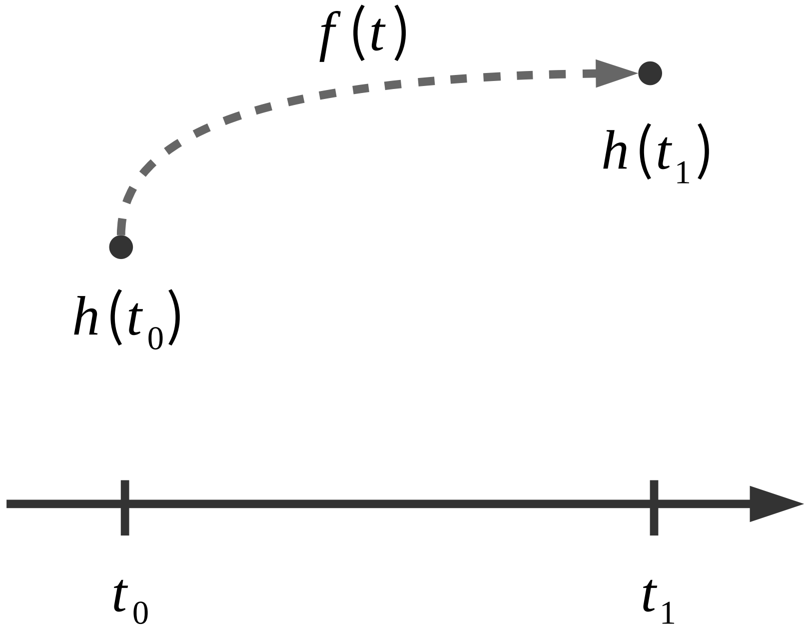
Integral for forwarding

$$h(t) = \int f(h(t), t, \theta)dt$$

Solve for next state

Integral for forwarding

$$h(t) = \int f(h(t), t, \theta)dt$$



Solve for next state

$\boldsymbol{h(t_0)}$ to $\boldsymbol{h(t_1)}$

$$h(t_1) = F(h(t), t, \theta)\Big|_{t=t_0}$$

Solving with ODE solver

$$h(t_1) = ODESolve(h(t_0), t_0, t_1, \theta, f)$$

Backpropagation

Optimization

$$\mathcal{L}(t_0, t, \theta) = \mathcal{L}(ODESolve(\cdot))$$

Require gradient respect to parameters

$$\frac{\partial \mathcal{L}}{\partial h(t)}$$

$$h(t_0) \longleftarrow h(t_1) \longleftarrow \ldots \longleftarrow h(t_n)$$

Adjoint method

Adjoint state

$$\frac{\partial \mathcal{L}}{\partial h(t)} = -a(t)$$

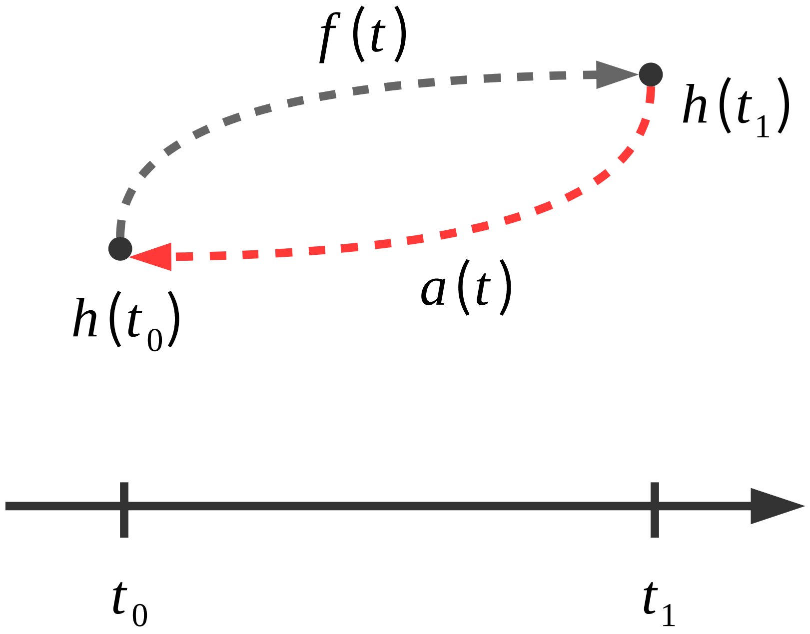
How to find $\boldsymbol{a(t)}$

$$a(t) = \int -a(t)^T \frac{\partial f}{\partial h} dt = -\frac{\partial \mathcal{L}}{\partial h(t)}$$

Adjoint method

Find $\boldsymbol{a(t)}$ from $\boldsymbol{f}$

$$a(t) = \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(h(t), t, \theta)}{\partial h(t)} dt$$



Augmented state

$$\frac{d\theta}{dt} = 0$$

$$\frac{dt}{dt} = 1$$

For computation efficiency, let

$$\begin{bmatrix} h \\ \theta \\ t \end{bmatrix}$$

be a augmented state

Augmented state function

$$f_{aug}(\begin{bmatrix} h \\ \theta \\ t \end{bmatrix}) = \begin{bmatrix} f(h(t), t, \theta) \\ 0 \\ 1 \end{bmatrix}$$

Augmented state dynamics

$$\frac{d}{dt}\begin{bmatrix} h \\ \theta \\ t \end{bmatrix} = f_{aug}(\begin{bmatrix} h \\ \theta \\ t \end{bmatrix})$$

Augmented adjoint state

$$a_{aug} = \begin{bmatrix} a \\ a_\theta \\ a_t \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial h} \\ \frac{\partial \mathcal{L}}{\partial \theta} \\ \frac{\partial \mathcal{L}}{\partial t} \end{bmatrix}$$

Augmented adjoint state dynamics

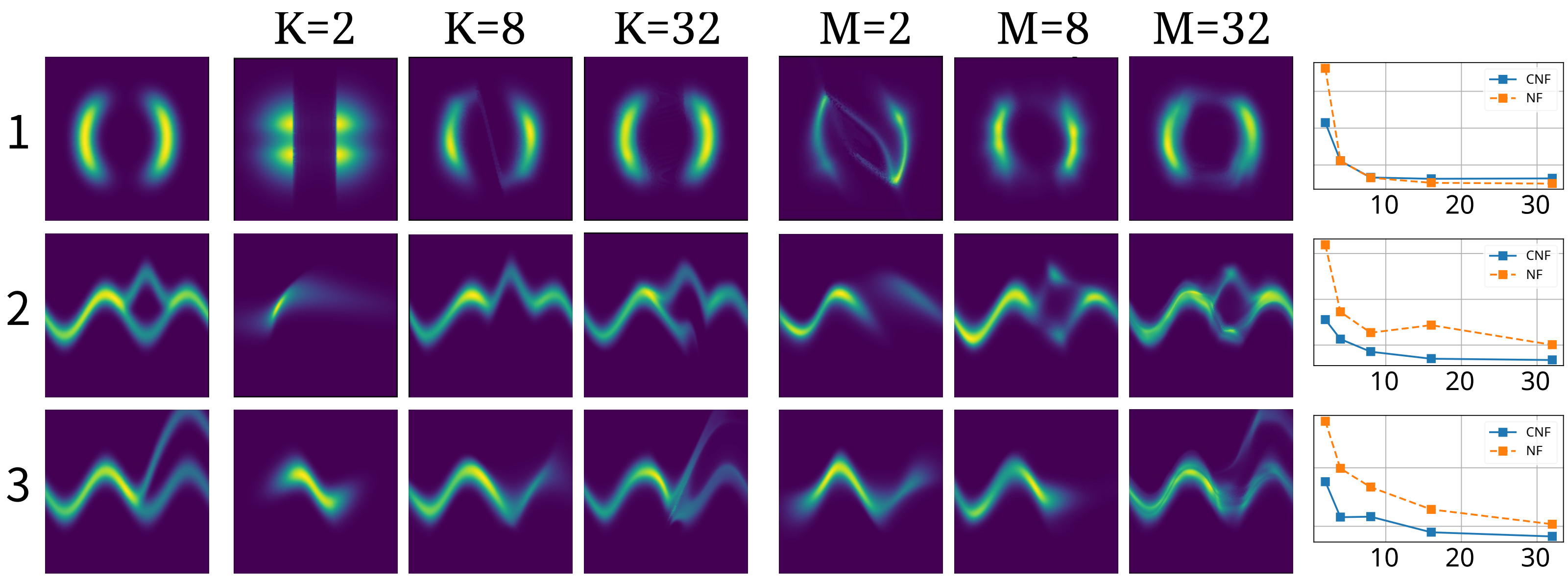
$$\frac{da_{aug}}{dt} = - \begin{bmatrix} a\frac{\partial f}{\partial h} \\ a\frac{\partial f}{\partial \theta} \\ a\frac{\partial f}{\partial t} \end{bmatrix}$$

Replace ResNet with ODEs

- RK-Net: use Runge-Kutta integrator
- ODE-Net: use ODESolve
- Experiment on TensorFlow with GPU
- $\tilde{L}$: the numbers of function evaluations

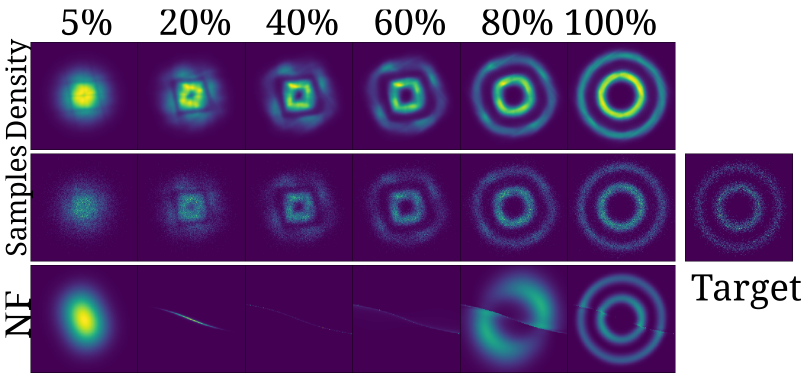
	Test Error	# Params	Memory	Time
1-Layer MLP[†]	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$O(L)$	$O(L)$
RK-Net	0.47%	0.22 M	$O(\tilde{L})$	$O(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\boldsymbol{O(1)}$	$O(\tilde{L})$

Continuous normalizing flow

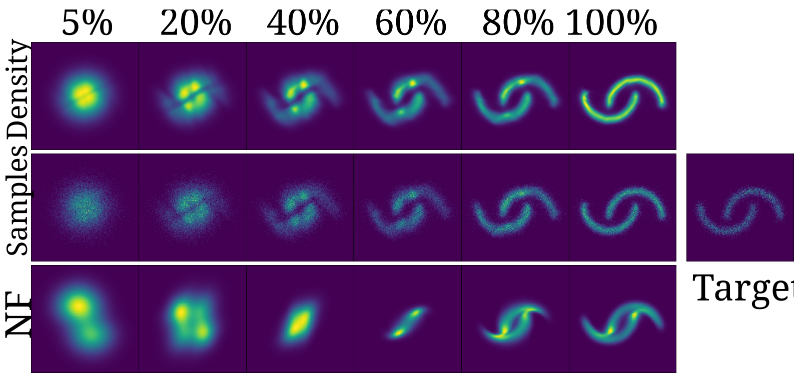


(a) Target (b) NF (c) CNF (d) Loss vs. K/I

Continuous normalizing flow

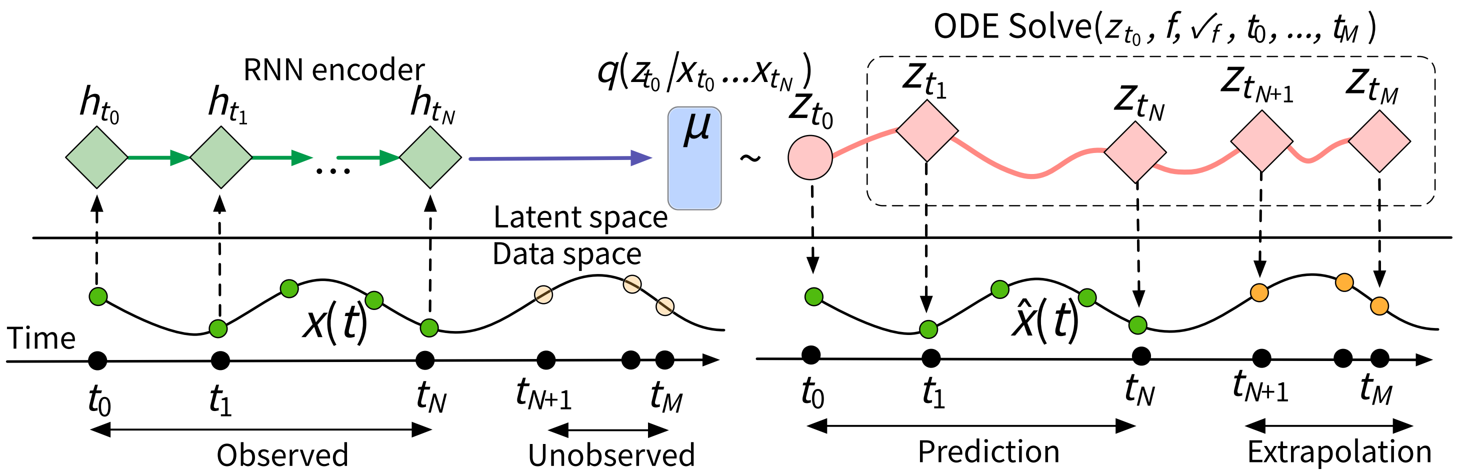


(a) Two Circles

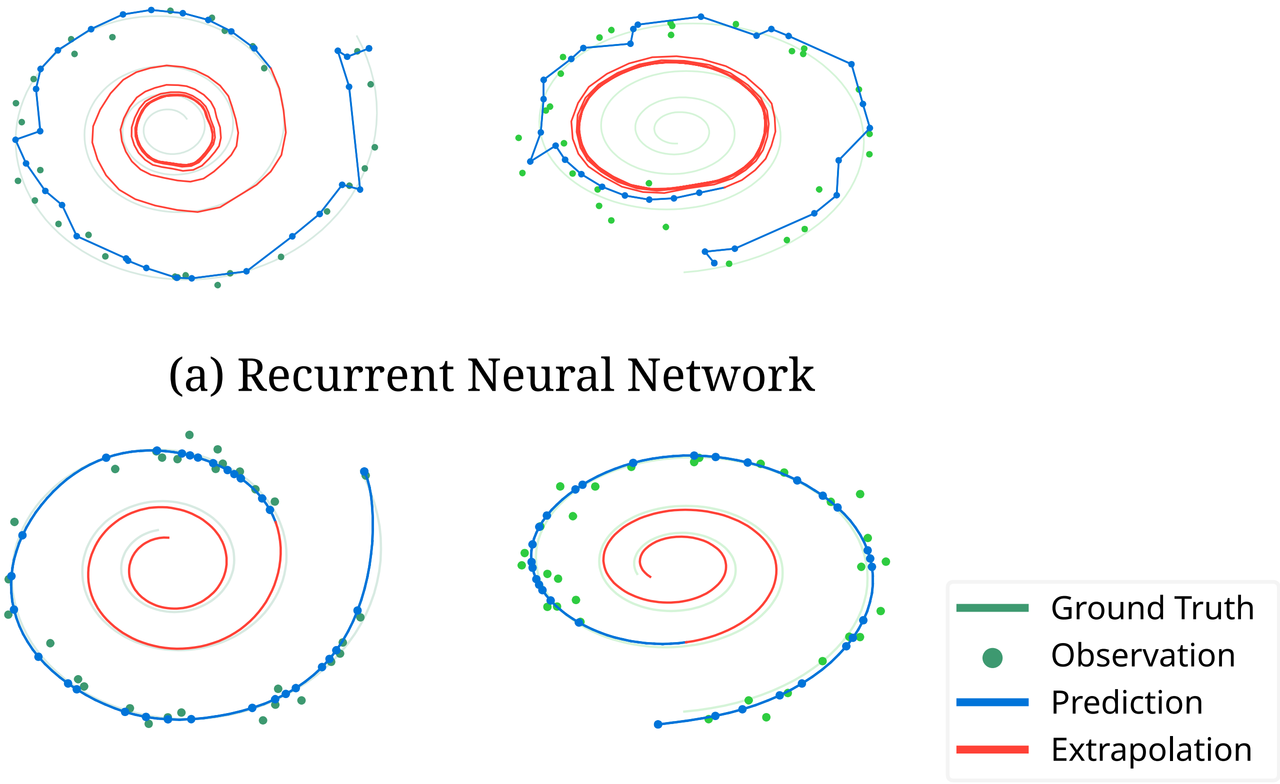


(b) Two Moons

Generative latent time-series model



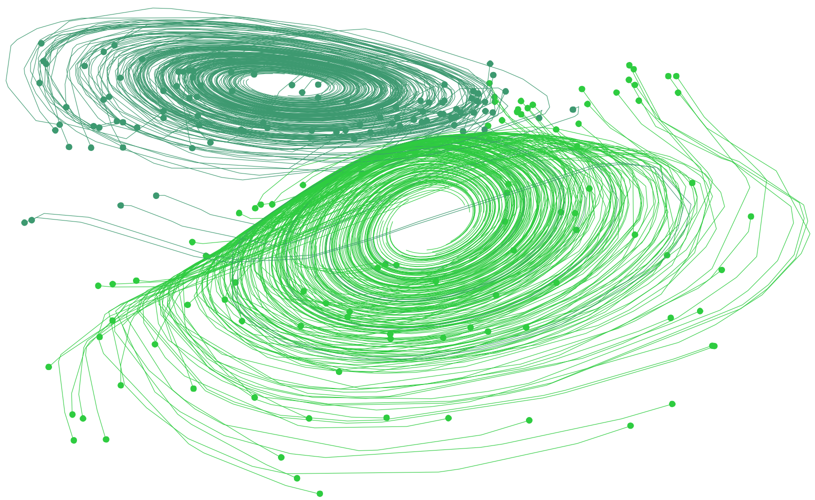
Generative latent time-series model



(a) Recurrent Neural Network

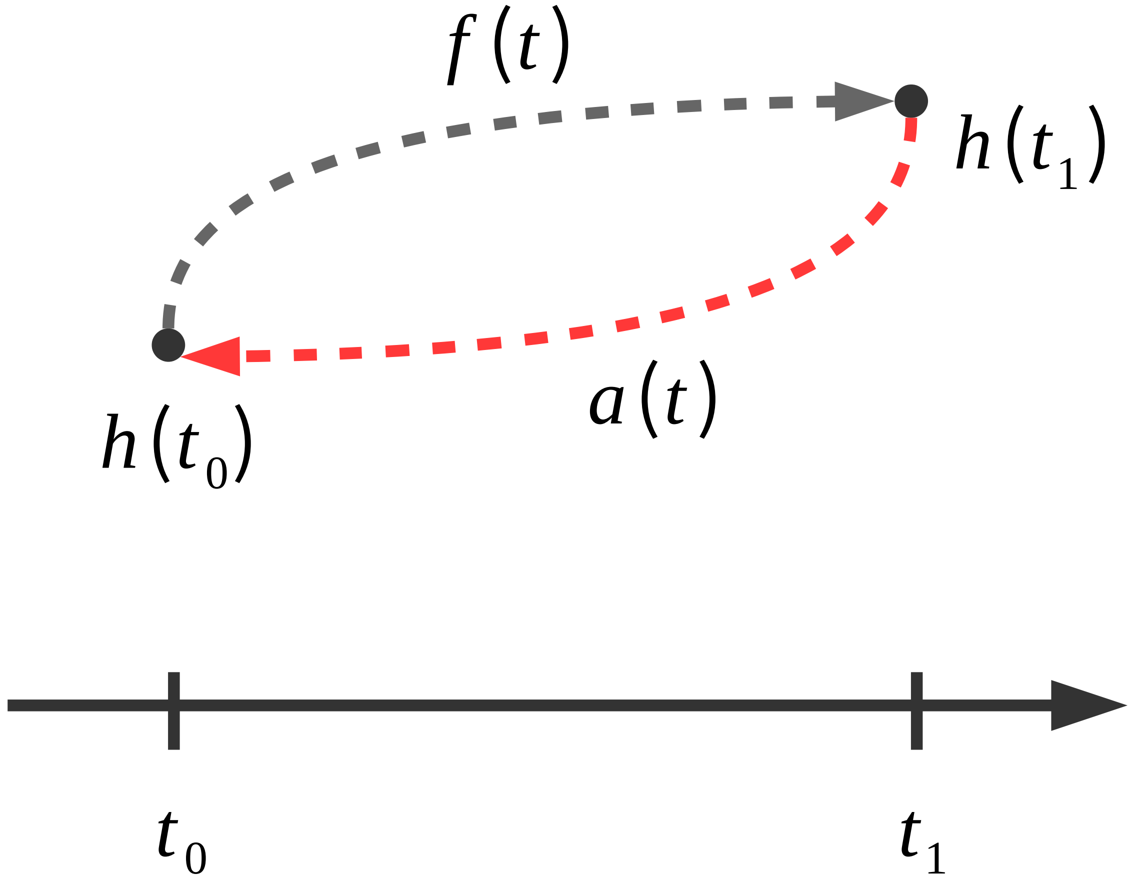
(b) Latent Neural Ordinary Differential Equation

Generative latent time-series model



(c) Latent Trajector

Backpropagation through ODE solver



Features and limitations

Minibatch

Uniqueness

Picard's existence thm.

> *The solution of an initial value problem exists and is unique, if the differential equation is **uniformly Lipschitz continuous** in z and **continuous** in t.*

Reversibility

Ajoint method is not reversible.

Why ODE?

Efficiency

Borrow concepts and interpretations from science

Why efficiency?

Continuous assumption guarantee convergence

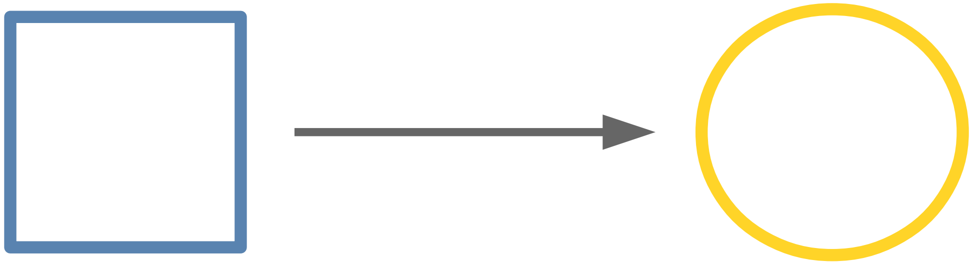
$$\int f^2 dt$$

converges

Continuous brings topology

Layer is a discrete mapping

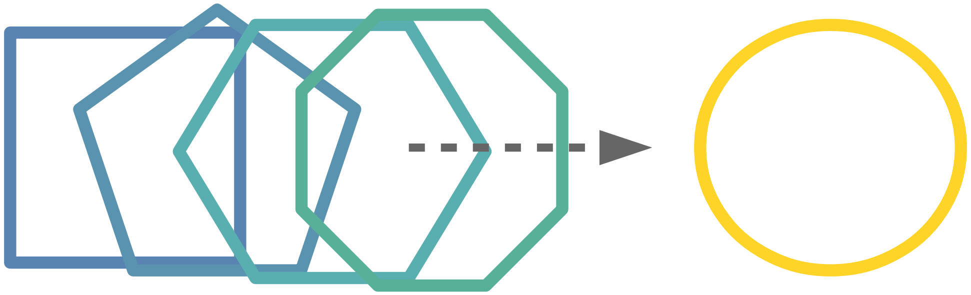
$$h_{t+1} = \sigma(Wh_t + b)$$



Continuous brings topology

Layer becomes continuous function

$$h(t+\Delta t) = \sigma(Wh(t)+b)$$



Thank you for attention

References

- Understanding Neural ODE's (https://jontysinai.github.io/jekyll/update/2019/01/18/understanding-neural-odes.html)
- Neural Ordinary Differential Equations (https://arxiv.org/abs/1806.07366)