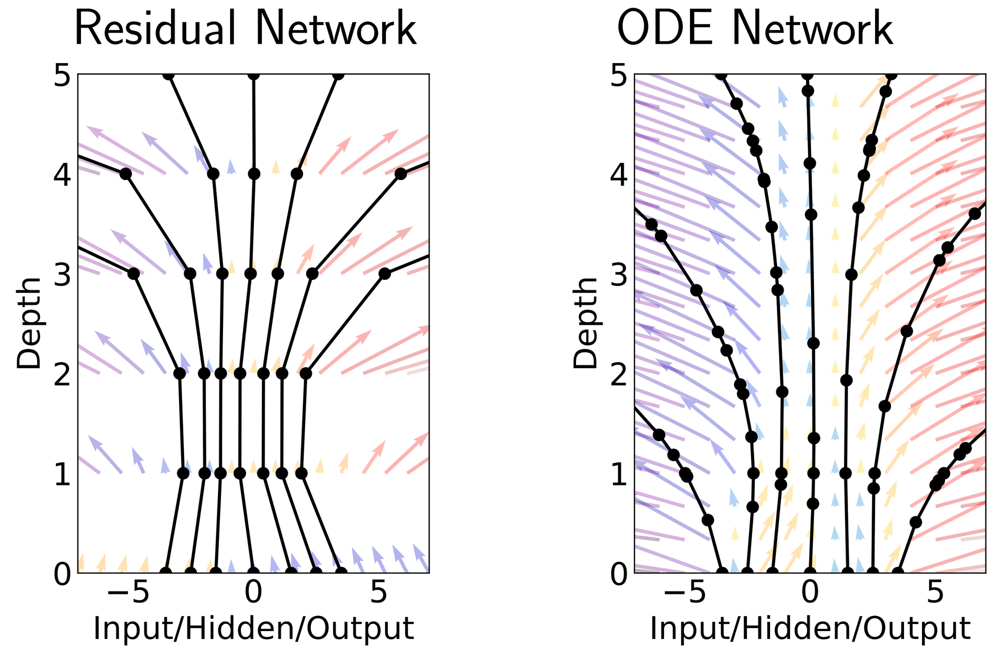
Neural Ordinary Differential Equations

Tian Qi Chen*, Yulia Rubanova*, Jesse Bettencourt*, David Duvenaud
*Equal Contribution   University of Toronto, Vector Institute

Introduction

We build a continuous depth residual network:



$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$ $\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$

Memory Efficient
- no backprop through solver operations
- no storing forward pass intermediates

Adaptive Computation
- evaluations scale with problem complexity
- solver monitors and controls approximation error

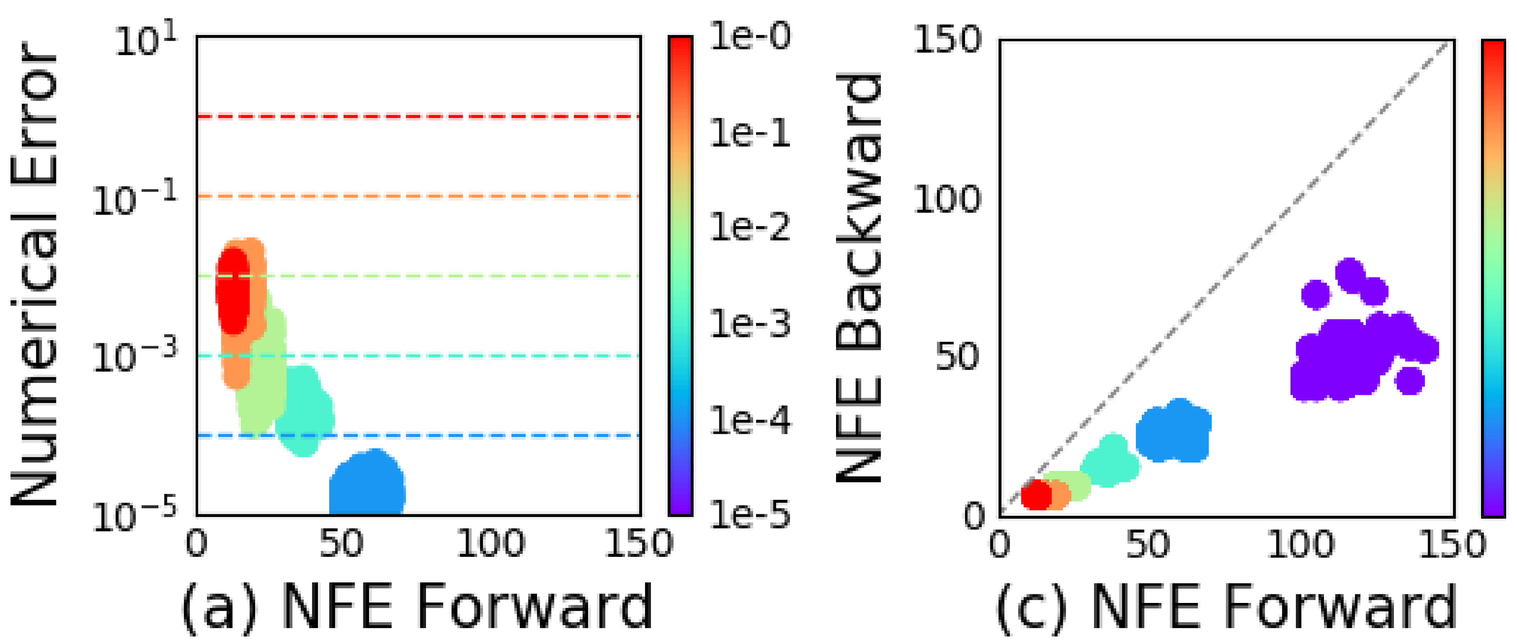
Parameter Efficiency
- dynamics are parameterized as continuous function of time
- nearby 'layers' automatically tied together

Replacing Residual Networks with ODE-Net for Supervised Learning

ODE-net replaces ResNet blocks with

$$\text{output} = \int_{t_0}^{t_1} f_\theta(\mathbf{h}(t), t, \theta)dt = \text{ODESolve}(\text{input}, f_\theta, t_0, t_1, \theta)$$

where f_θ is an MLP or conv layer

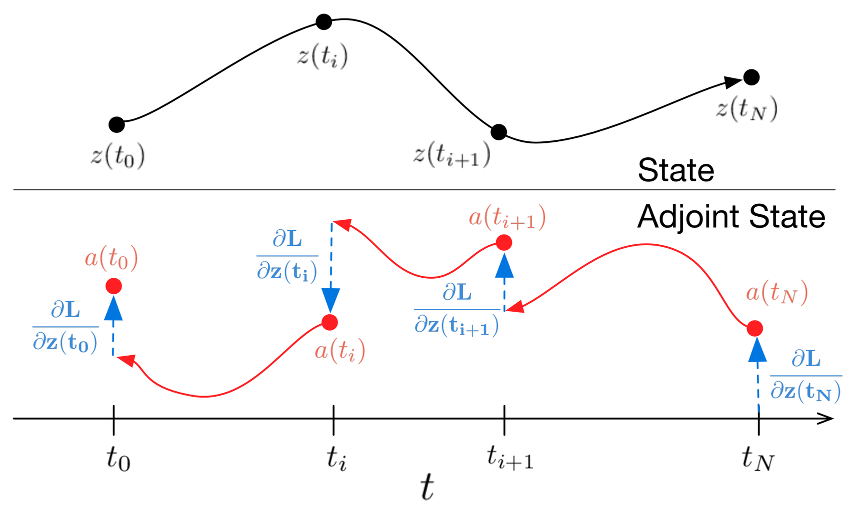


Statistics of a trained ODE-Net in Number of Function Evaluations

	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	$\mathcal{O}(1)$	$\mathcal{O}(\tilde{L})$

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

Reverse-mode automatic differentiation of ODE solutions



Adjoint sensitivity method requires solving augmented system backwards in time. All computed in single call to ODE solver, concatenating original, adjoint, other partials into single vector. This adjoint state is updated by gradient at each observation.

Theorem (Instantaneous Change of Variables)

Change of variables theorem to compute exact changes in probability of samples transformed through bijective f:

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

Assuming that f is uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t, then:

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

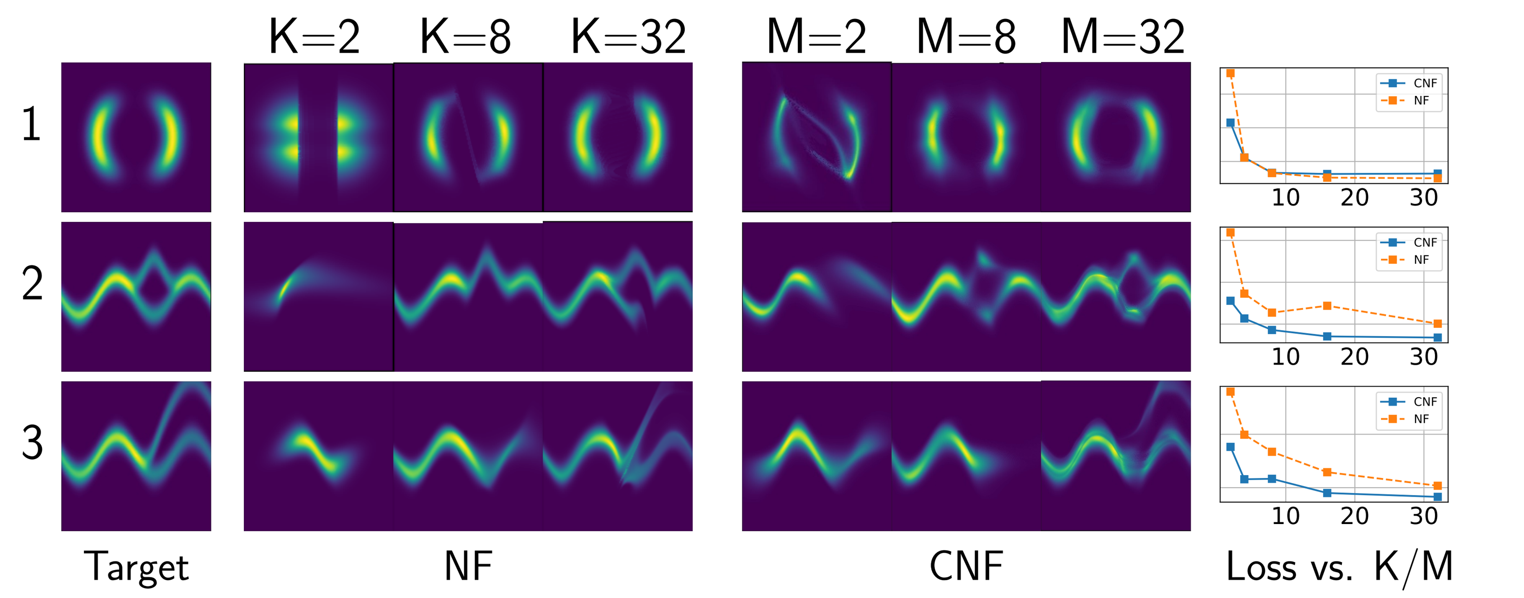
Continuous Normalizing Flows

Planar normalizing flow (Rezende and Mohamed, 2015):

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

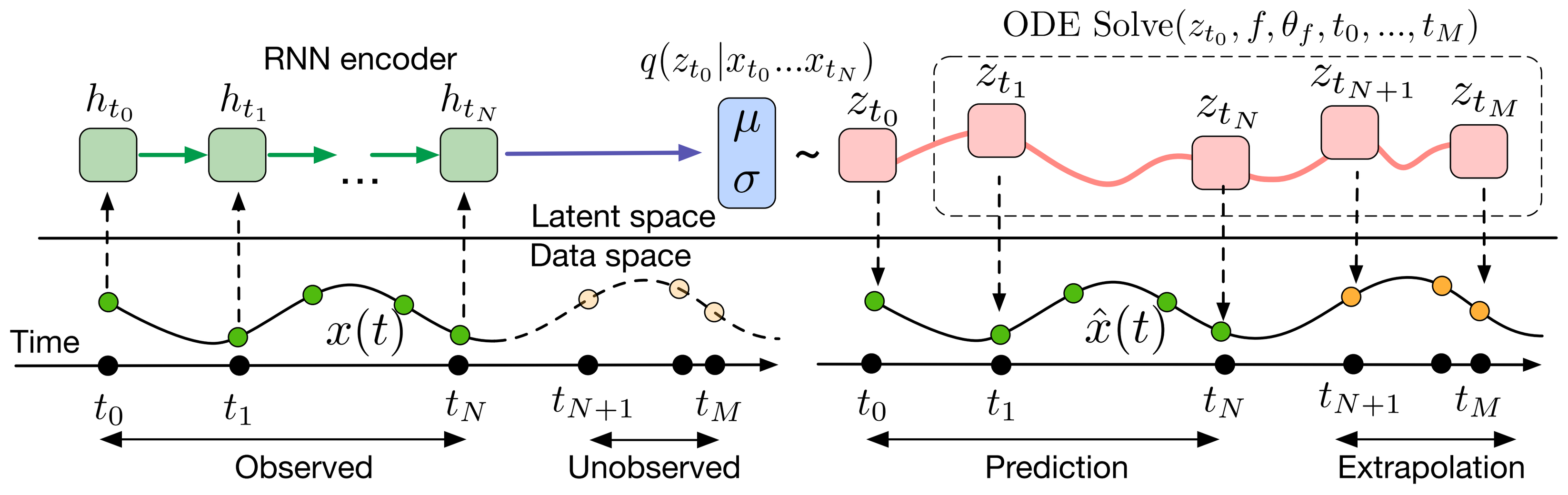
Continuous analog of the planar flow:

$$\frac{d\mathbf{z}(t)}{dt} = uh(w^{\mathsf{T}}\mathbf{z}(t) + b)$$
$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -u^{\mathsf{T}}\frac{\partial h}{\partial \mathbf{z}(t)}$$

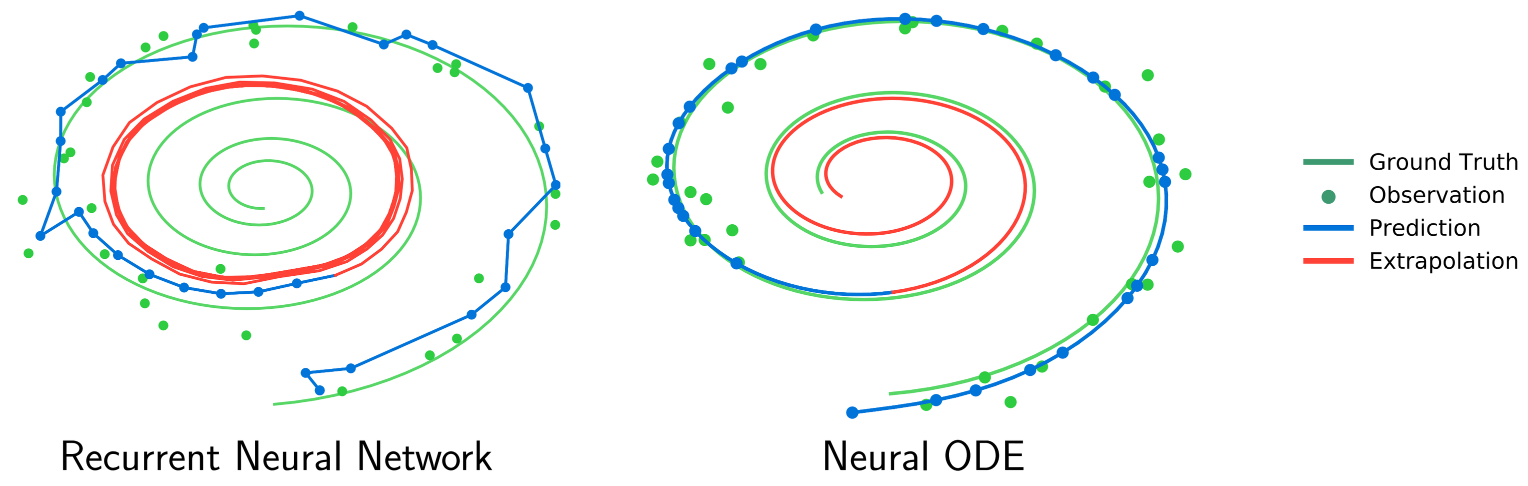


Comparison of normalizing flows versus continuous normalizing flows. The model capacity of NF determined by depth (K), while CNF can also increase capacity by increasing width (M).

Continuous-time Generative Time Series Modelling



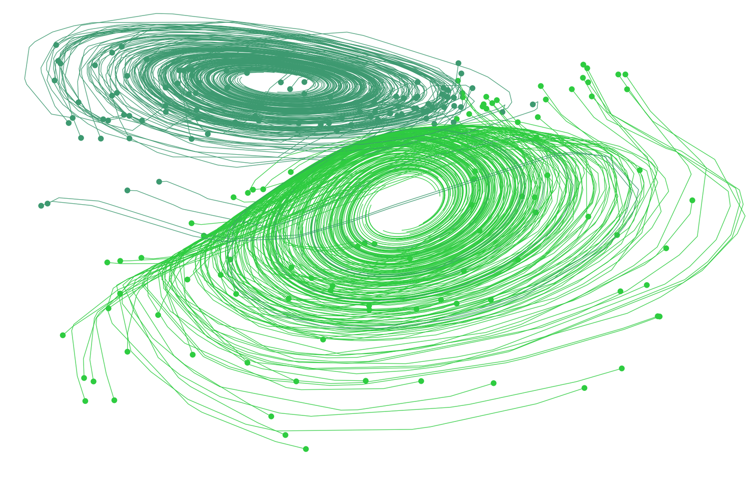
Computation graph of continuous latent time series model.



Recurrent Neural Network   Neural ODE

Neural ODE learns smooth latent dynamics from noisy observations.

$$\mathbf{z}_{t_0} \sim p(\mathbf{z}_{t_0})$$
$$\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \ldots, \mathbf{z}_{t_M} = \text{ODESolve}(\mathbf{z}_{t_0}, f, \theta_f, t_0, \ldots, t_M)$$
$$\text{each} \quad \mathbf{x}_{t_i} \sim p(\mathbf{x}|\mathbf{z}_{t_i}, \theta_{\mathbf{x}})$$



Learned latent dynamics distinguishes between spiral directions.

Conclusion

- Black-box ODE solvers as modelling component.
- New models for time-series, supervised learning, and density estimation.
- Adaptive evaluation and allows explicit control of tradeoff between computation speed and accuracy.
- Derive instantaneous version of change of variables formula and develop continuous-time normalizing flows.