Neural ODE's

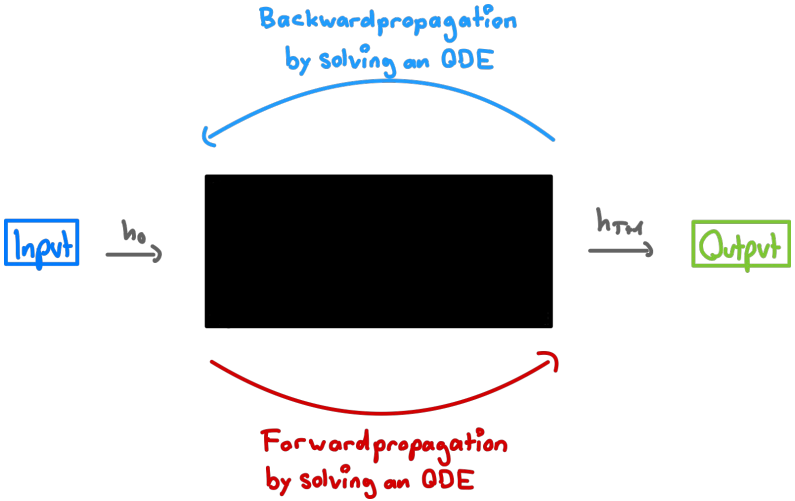
A Paper by **Ricky Chen, Julia Rubanova, Jesse Bettencourt, David Duvenaud**

Yuri Simantob

@ysimantob

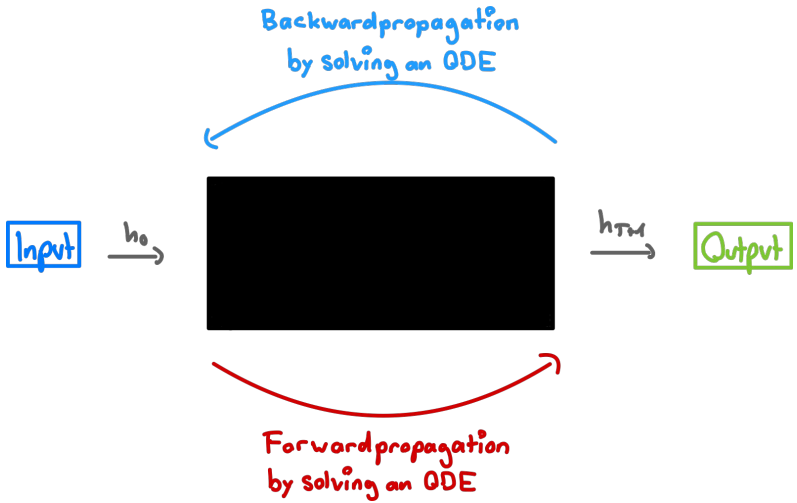
ETH zürich

Neural ODE's: A Brief Introduction



ETH zürich

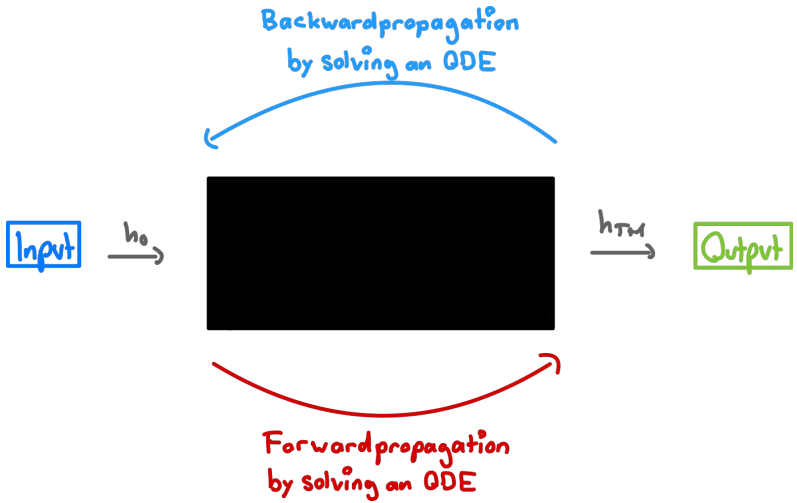
Neural ODE's: A Brief Introduction



- One can trade-off accuracy with runtime

ETH zürich

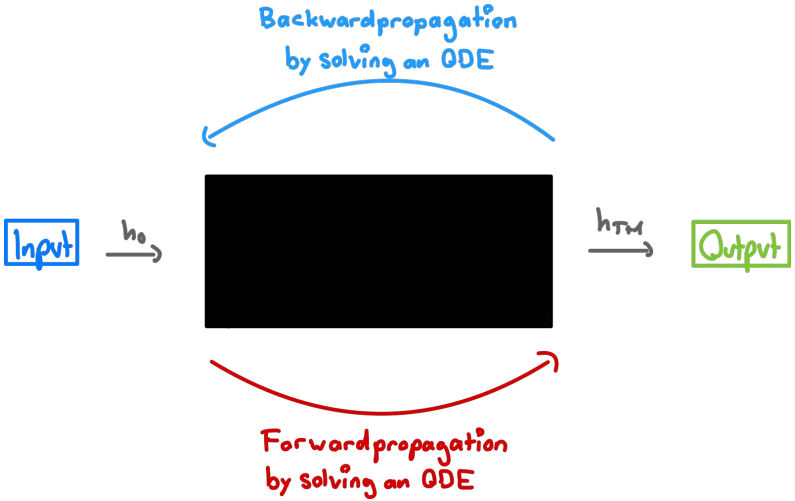
Neural ODE's: A Brief Introduction



- One can trade-off accuracy with runtime
- Simpler model

ETH zürich

Neural ODE's: A Brief Introduction



- One can trade-off accuracy with runtime
- Simpler model
- Memory-Efficient Propagations

ETH zürich

Overview of The Presentation

- How It Works: Forward & Backwardpropagation
- ODENets vs. ResNets
- Theoretical Application: Continuous Normalizing Flows
- Practical Application: Generative Latent Time-Series Models
- Criticism, Further Research & Applications

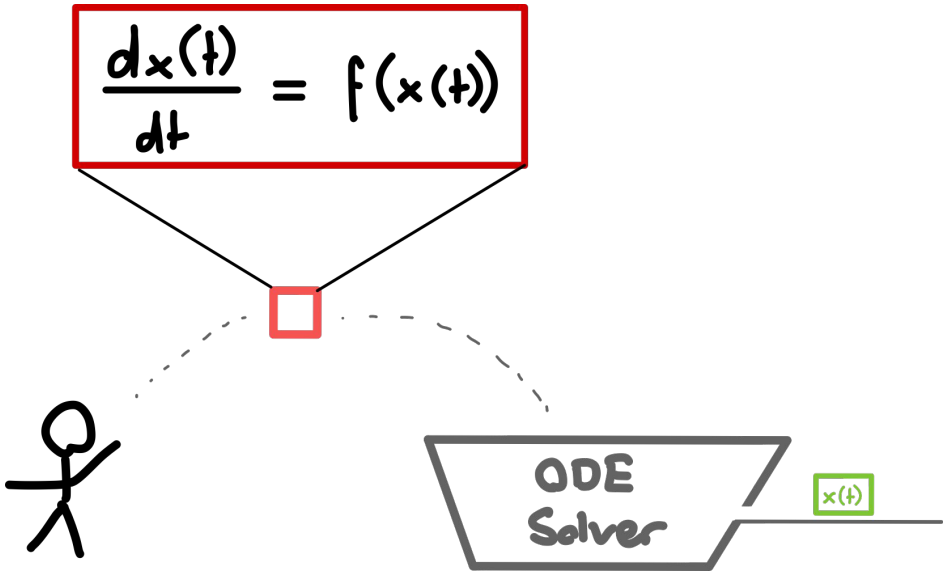
ETH zürich

Recap: ODEs

$$\frac{dx(t)}{dt} = f(x(t))$$

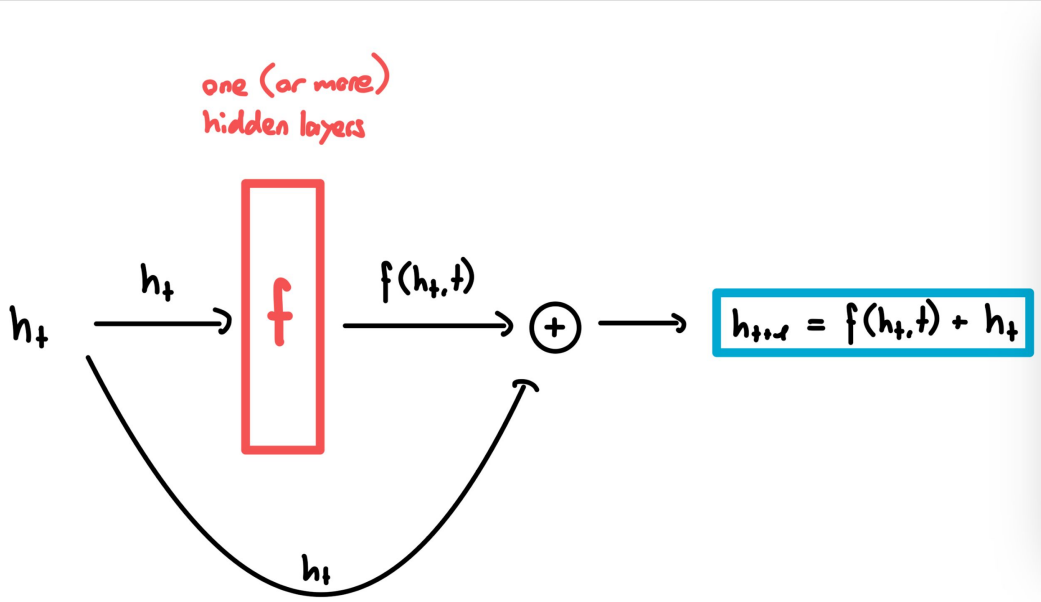
ETH zürich

Recap: ODEs



ETH zürich

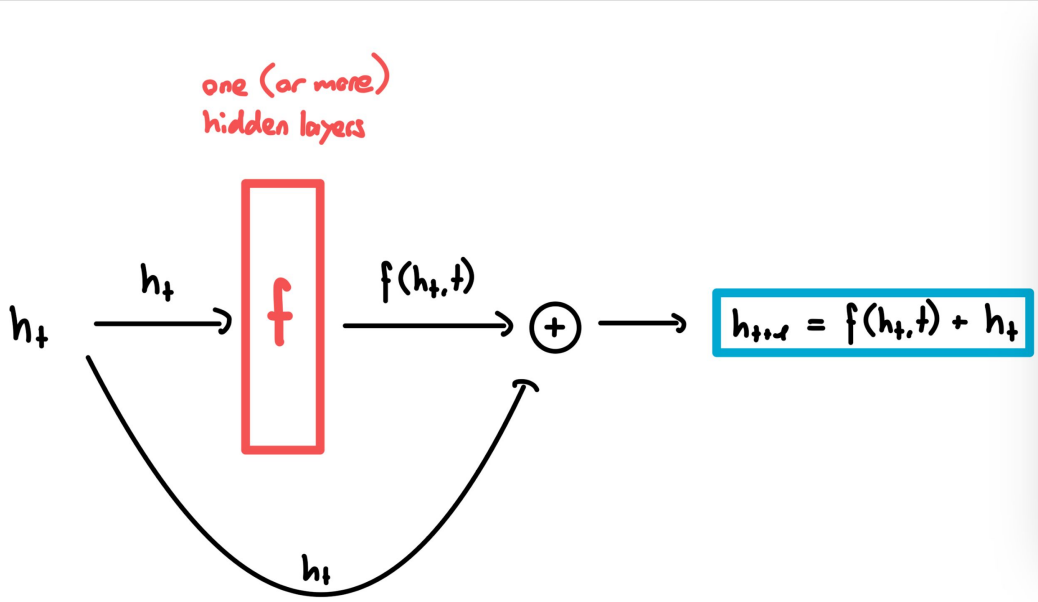
Forwardpropagation: ResNets



- addresses vanishing gradient problem
- Continues to be state-of-the-art in Computer Vision

ETH zürich

Forwardpropagation: ResNets

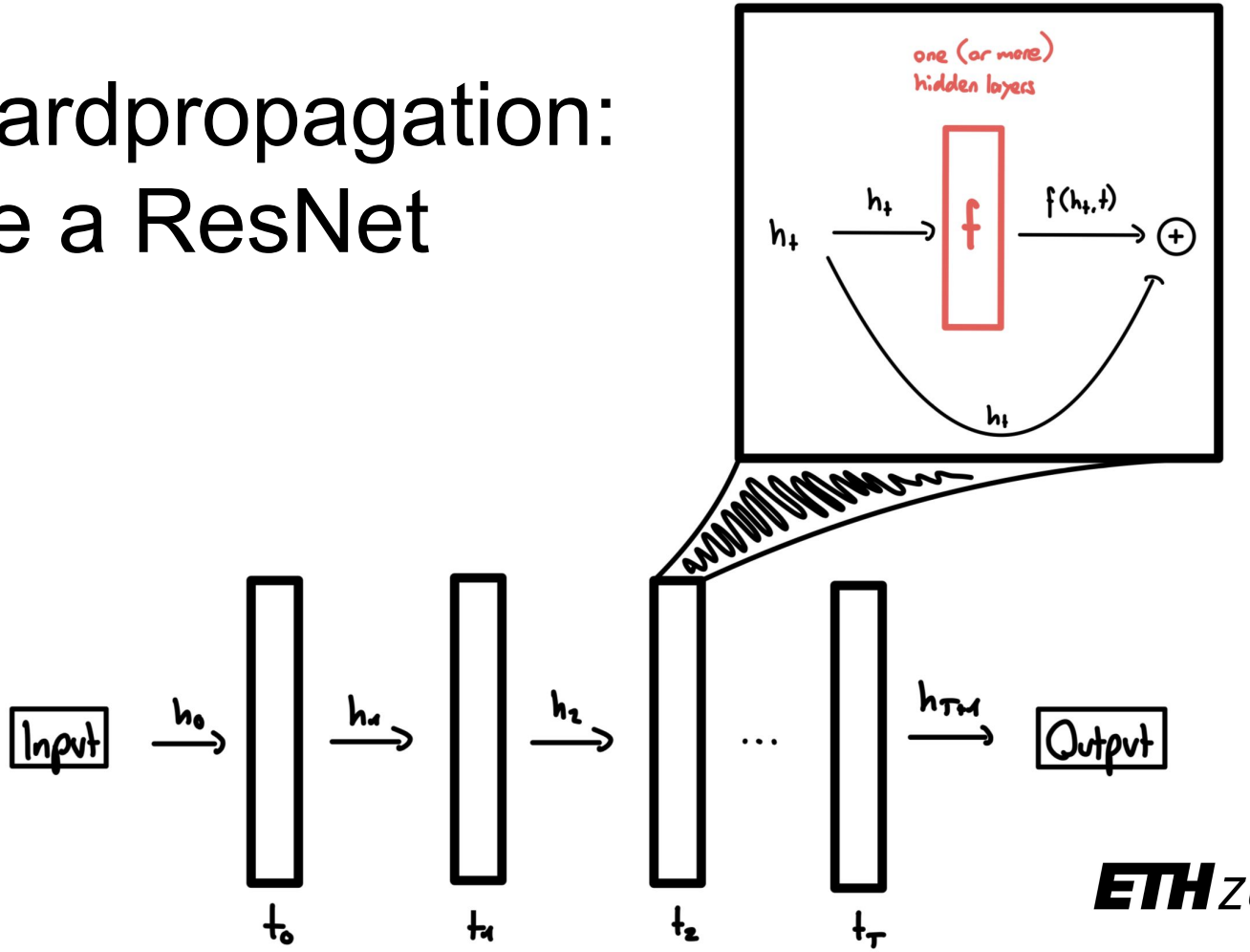


- addresses vanishing gradient problem
- Continues to be state-of-the-art in Computer Vision

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

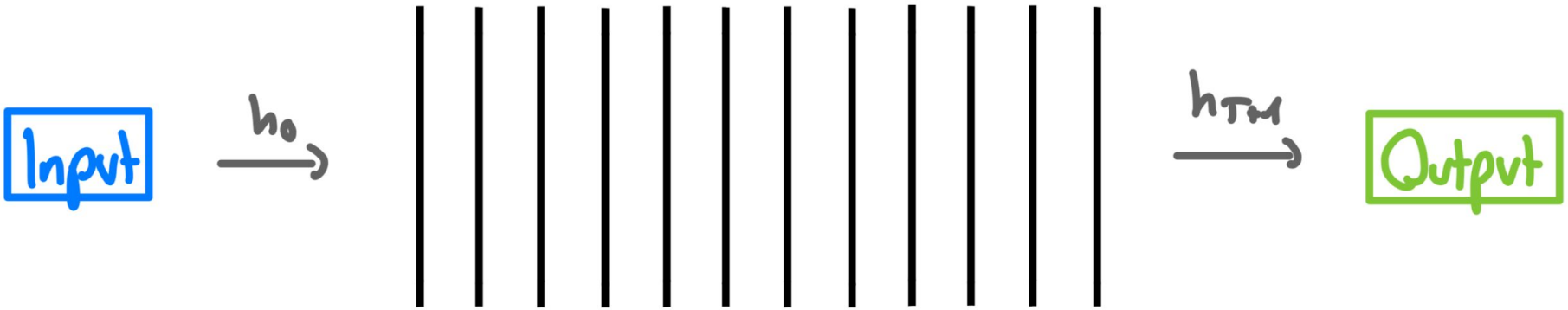
ETH zürich

Forwardpropagation: Inside a ResNet

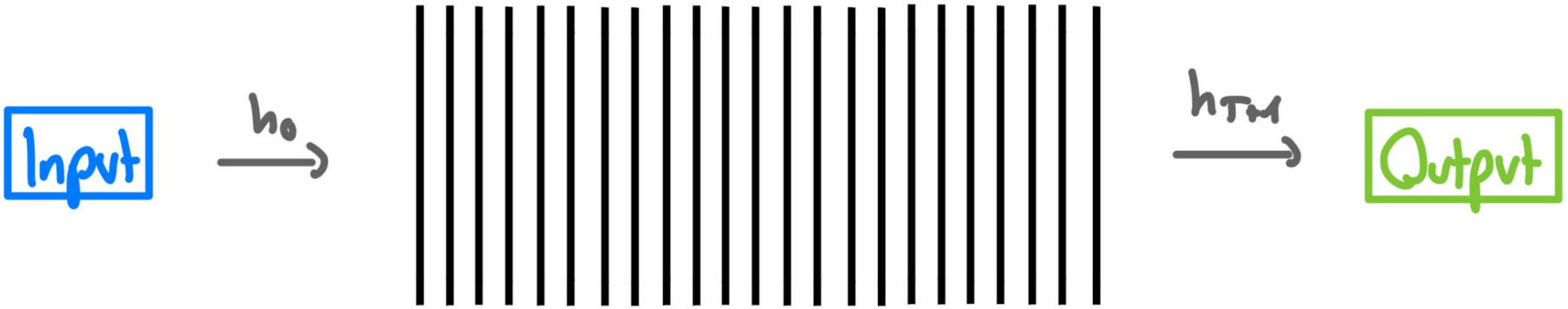


ETH zürich

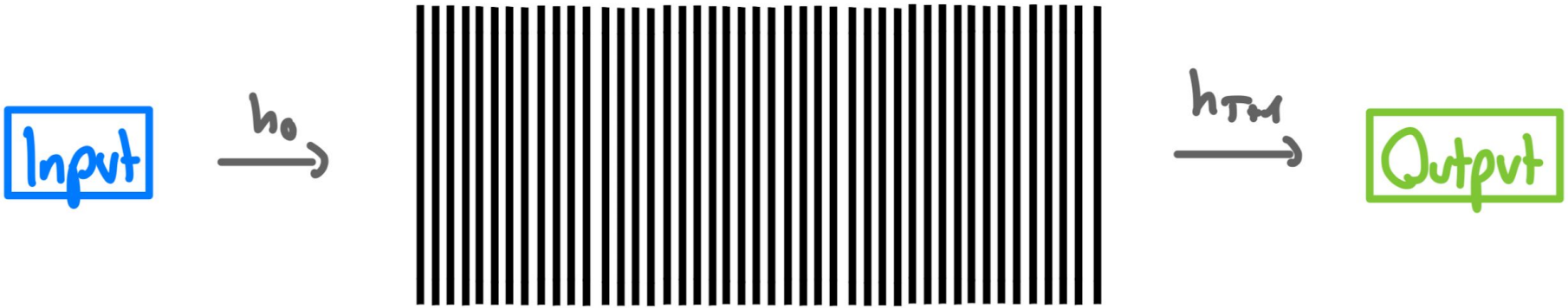
What if $T \to \infty$?



What if $T \to \infty$?



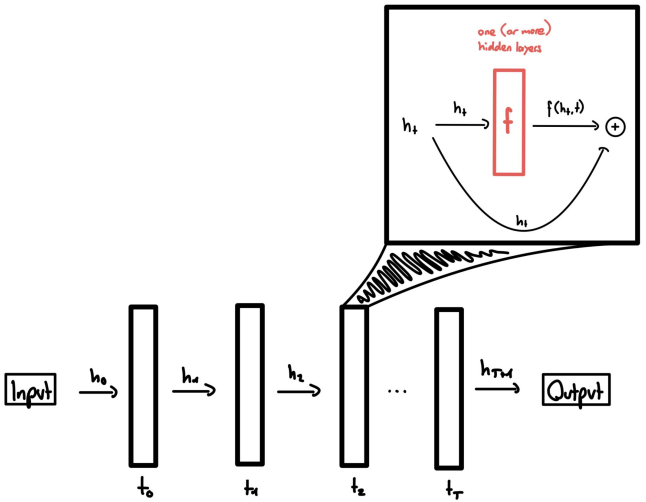
What if $T \to \infty$?



What if $T \to \infty$?

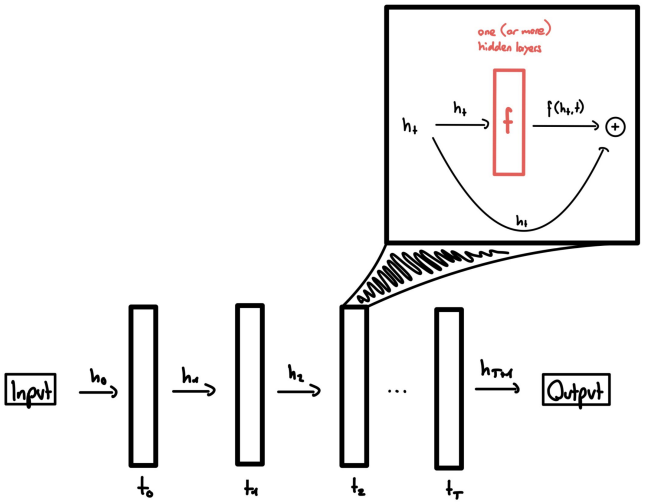
Forwardpropagation: Constructing the ODE

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



Forwardpropagation: Constructing the ODE

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

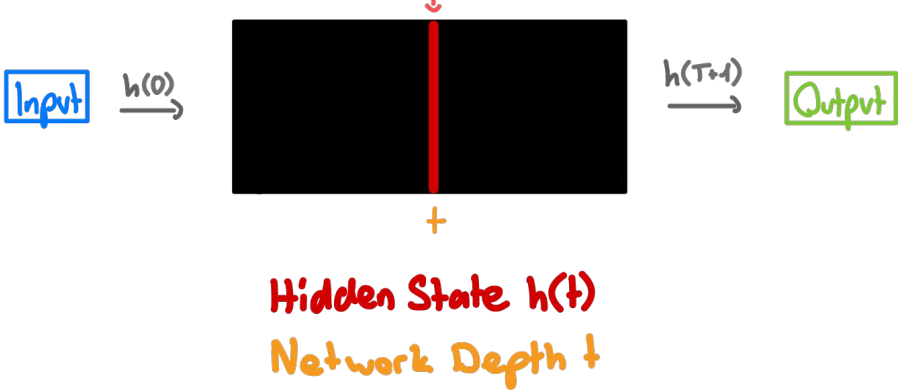


$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

ETH zürich

Forwardpropagation: Constructing the ODE

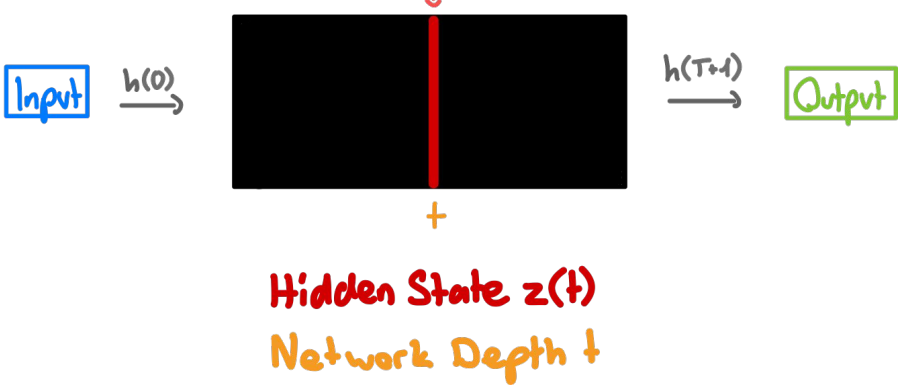
$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$



ETH zürich

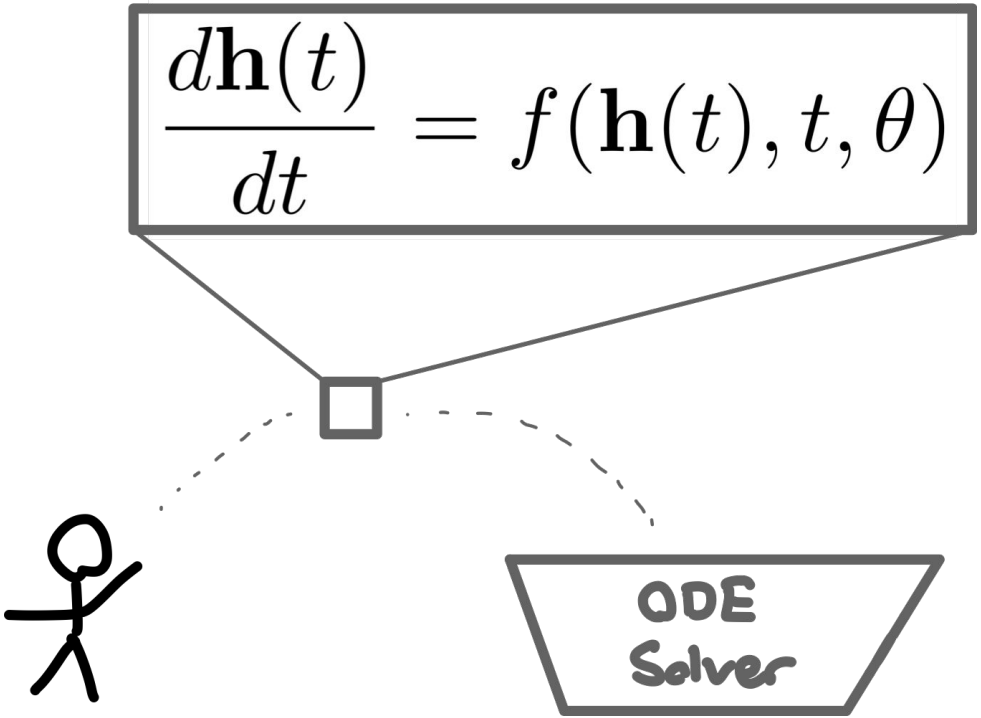
Forwardpropagation: Constructing the ODE

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



ETH zürich

Forwardpropagation: Solving the ODE



ETH zürich

Forwardpropagation: Solving the ODE

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

ETH zürich

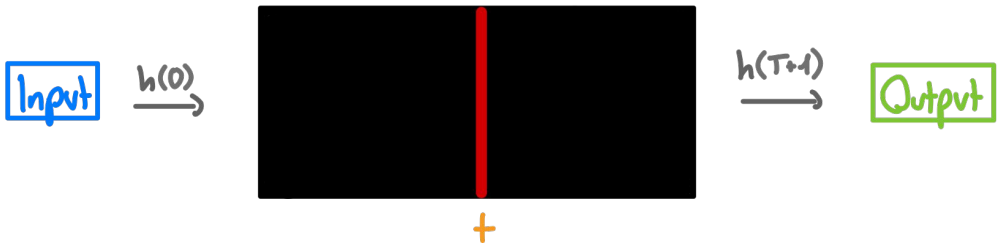
Forwardpropagation: Getting the Output

For a ODE network with depth T:

h(0): Input

h(t): Valve of the hidden "layer" at depth t

h(T+1): Output

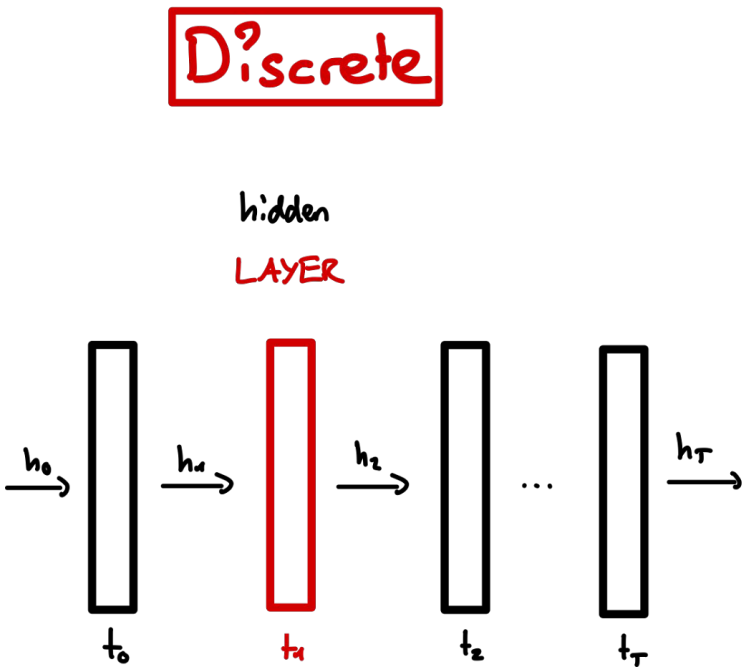


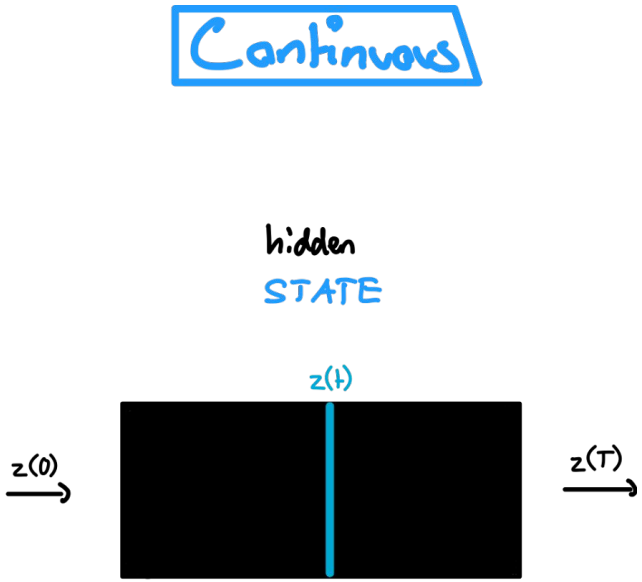
Backpropagation

Backpropagation: Notation

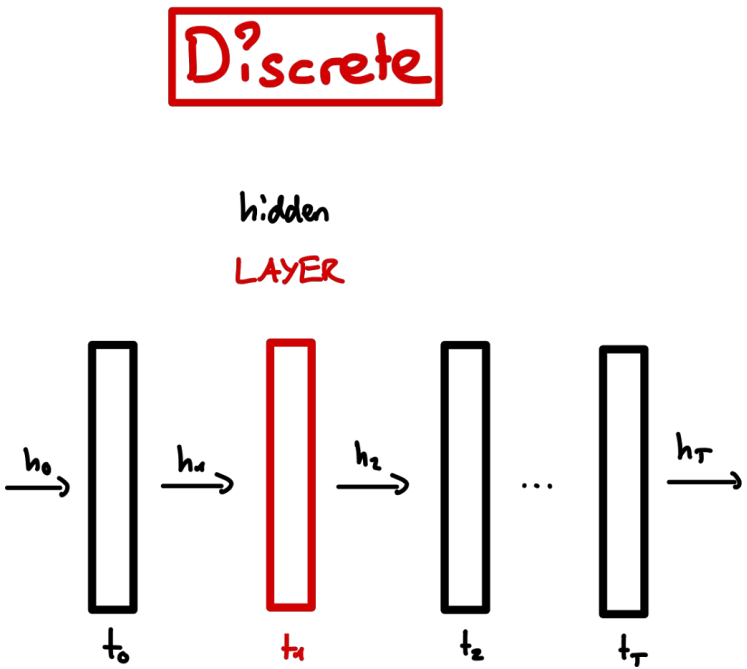
ETH zürich

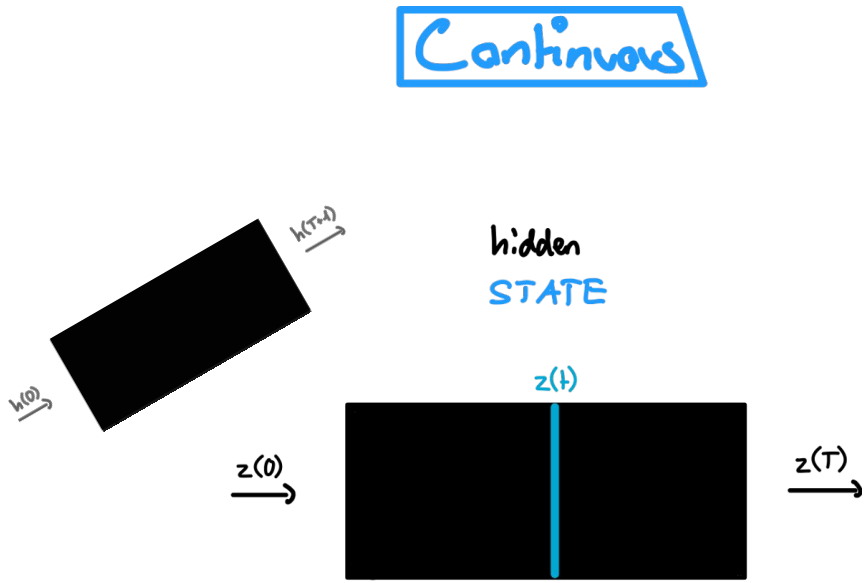
Backpropagation: Notation





Backpropagation: Notation

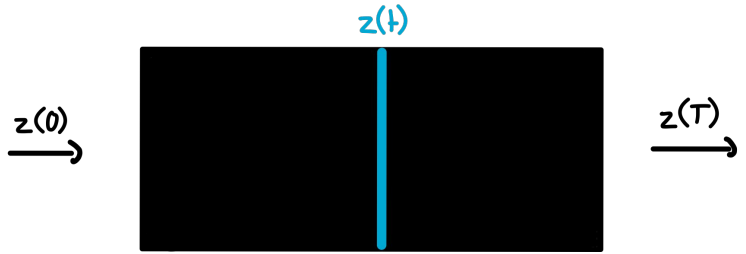




Backpropagation: The Adjoint

$$a(t) = \frac{dL}{dz(t)}$$

or how much a hidden state $z(t)$ contributes to the loss L



ETH zürich

Backpropagation: Solving for the Adjoint

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

ETH zürich

Backpropagation: Solving for the Adjoint

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

ETH zürich

Backpropagation: Weight Updates

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

Backpropagation: Weight Updates

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

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

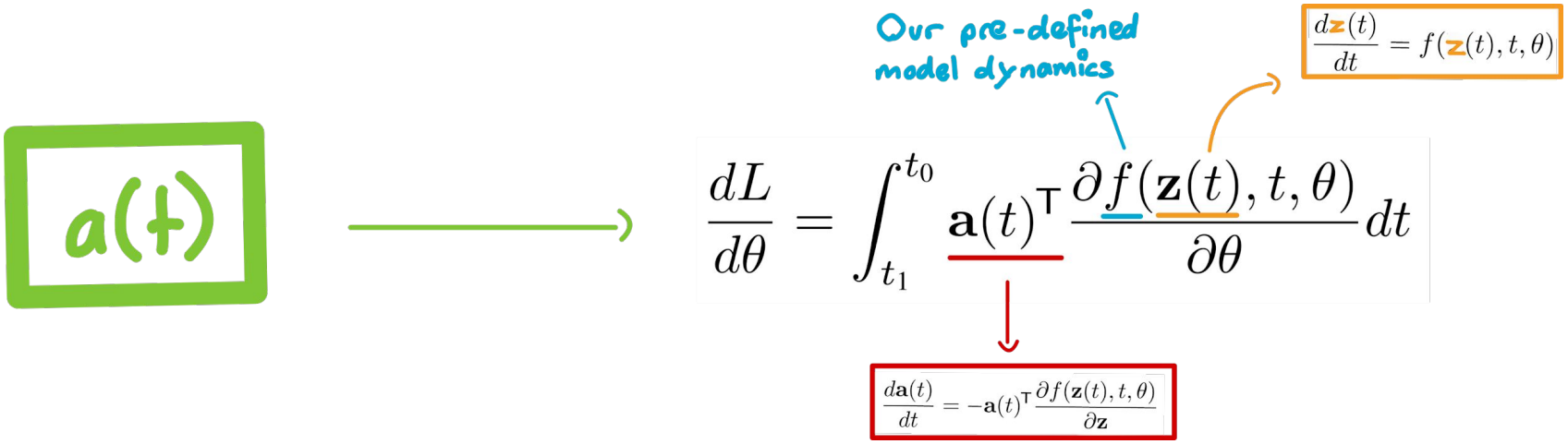
Backpropagation: Weight Updates

Our pre-defined model dynamics

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

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

Backpropagation: Weight Updates



Advantages of Continuous Propagation

Advantages of Continuous Propagation

- Forward propagation requires one call to an ODE solver

Advantages of Continuous Propagation

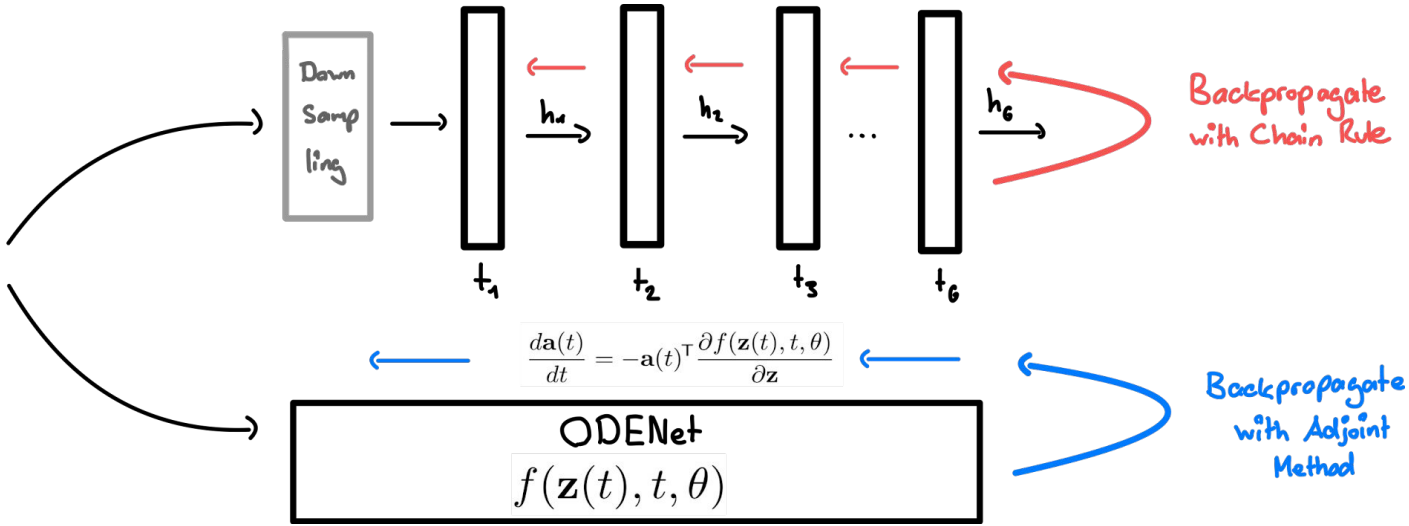
- Forward propagation requires one call to an ODE solver
- Backward propagation requires one call to an ODE solver

ETH zürich

Advantages of Continuous Propagation

- Forward propagation requires one call to an ODE solver
- Backward propagation requires one call to an ODE solver
- Runtime & Memory independent of network depth

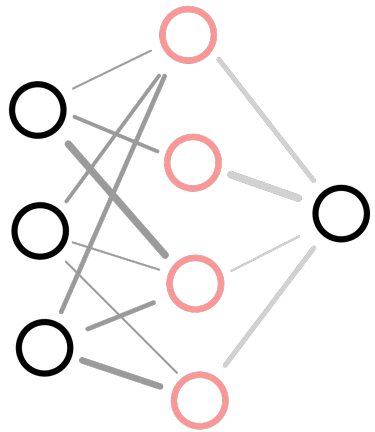
ODENets vs. ResNets on MNIST



ETH zürich

Results: 1-Layer Perceptron

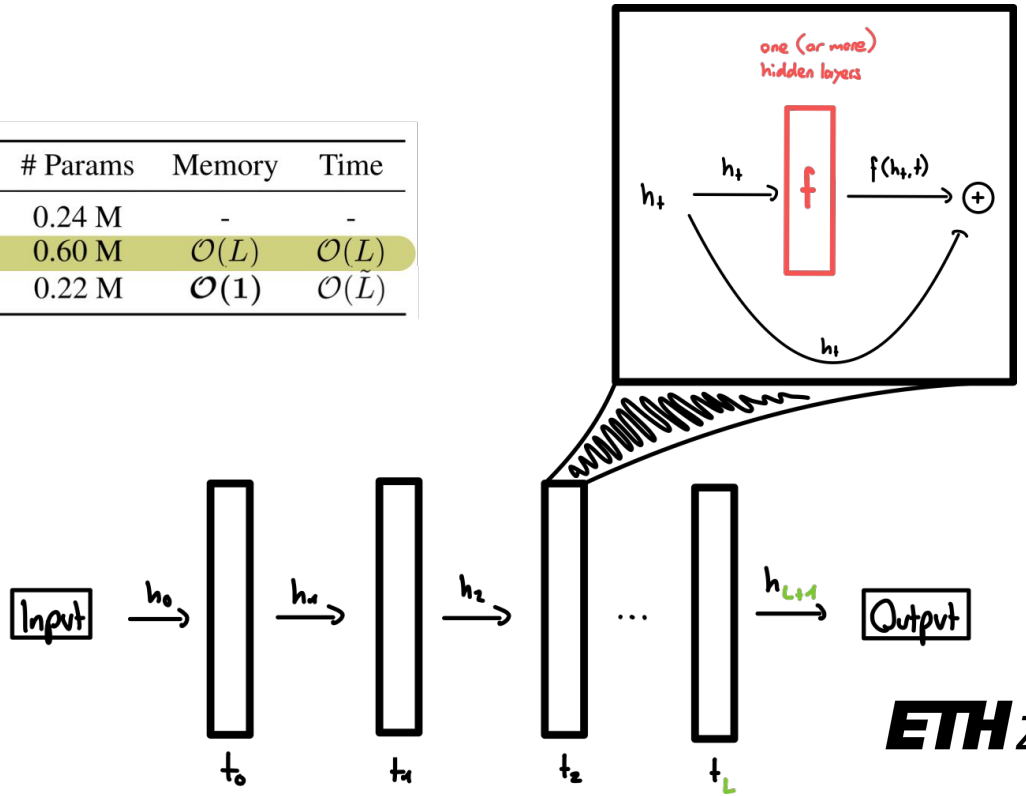
	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
ODE-Net	0.42%	0.22 M	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$



ETH zürich

Results: ResNet

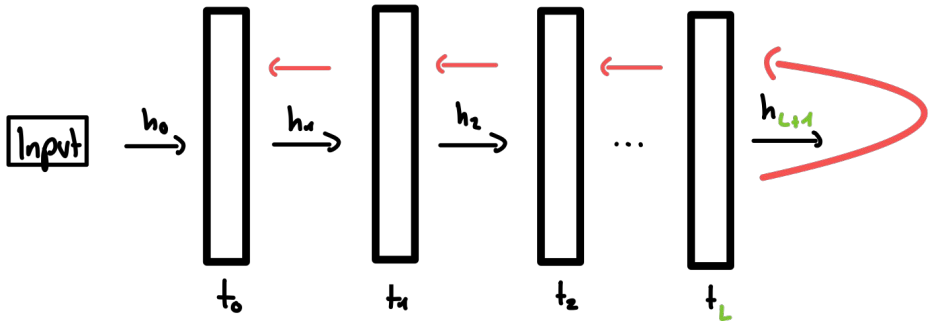
	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
ODE-Net	0.42%	0.22 M	$\mathcal{O}(1)$	$\mathcal{O}(\tilde{L})$



ETH zürich

Results: ResNet

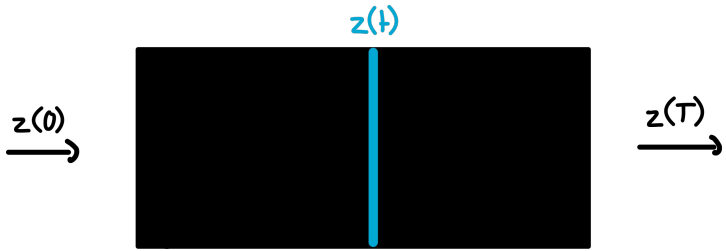
	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
ODE-Net	0.42%	0.22 M	$\mathcal{O}(1)$	$\mathcal{O}(\tilde{L})$



ETH zürich

Results: ResNet

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



ETH zürich

ODENets vs. ResNets: The Results

- constant memory with respect to number of layers
- fewer parameters
- runtime is linear with respect to the number of evaluations

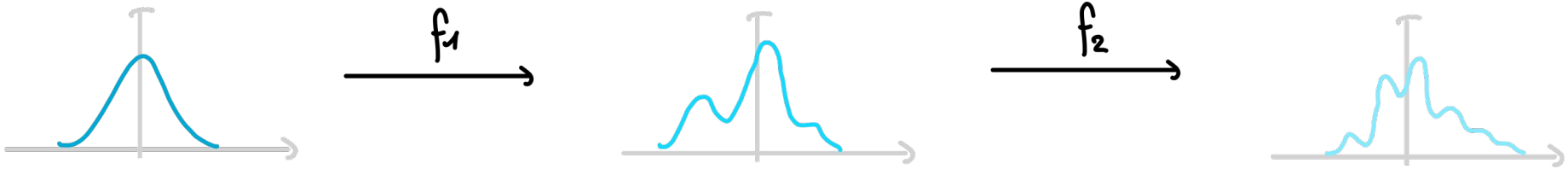
- linear memory with respect to number of layers
- more parameters
- runtime is linear with respect to the number of layers

ETH zürich

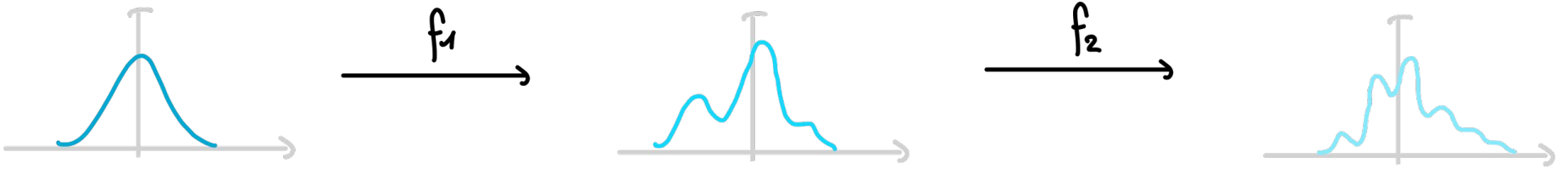
Theoretical Application: Continuous Normalizing Flows

ETH zürich

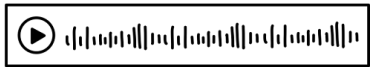
Normalizing Flows: Transforming Distributions



Normalizing Flows: Transforming Distributions

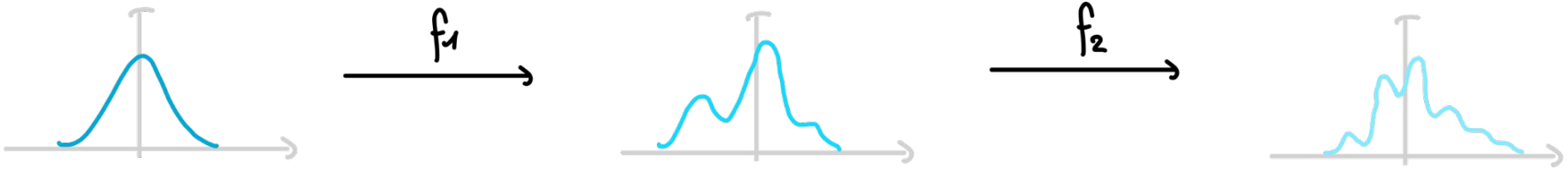


"This graph here should depict how normalizing flows create complex distributions from simple ones"

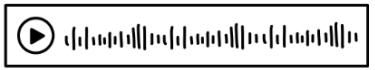


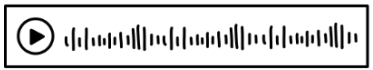
ETH zürich

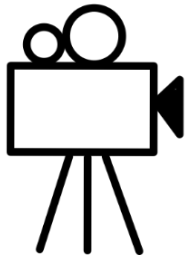
Normalizing Flows: Transforming Distributions



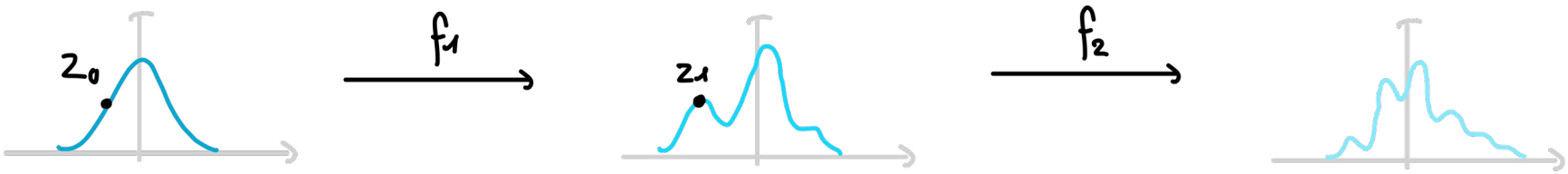
“This graph here should depict how normalizing flows create complex distributions from simple ones”







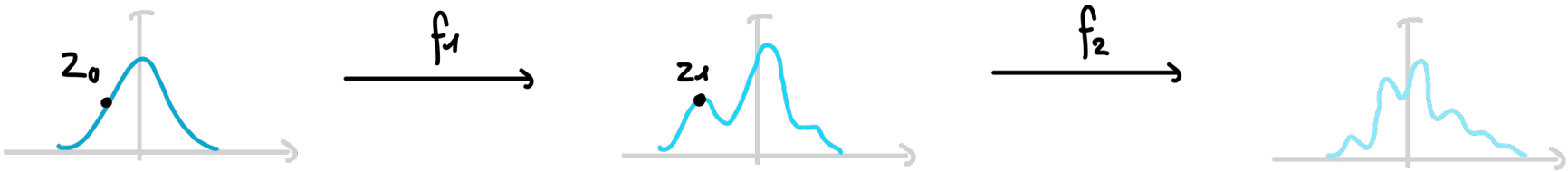
Normalizing Flows: The Main Bottleneck



$$\mathbf{z}_1 = f(\mathbf{z}_0)$$

ETH zürich

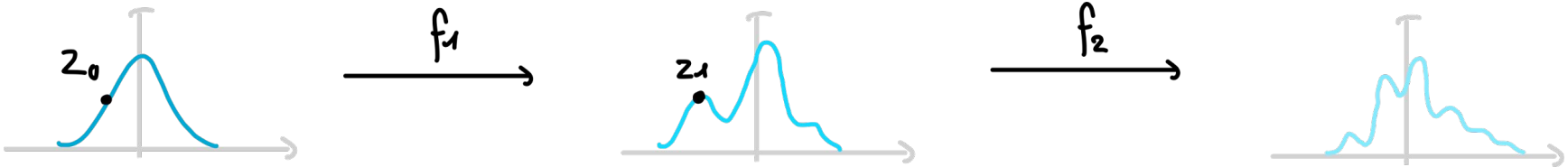
Normalizing Flows: The Main Bottleneck



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

ETH zürich

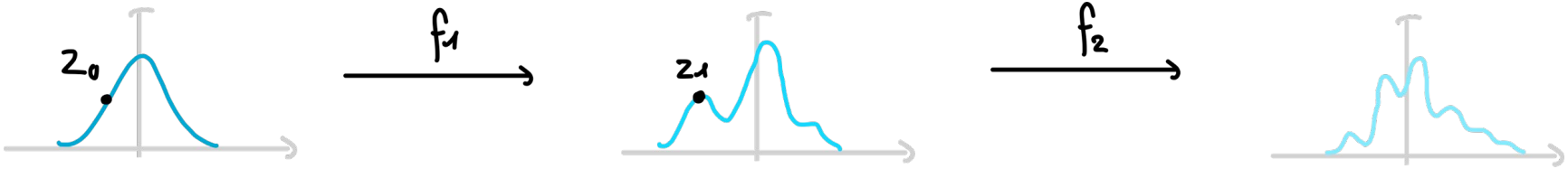
Normalizing Flows: The Main Bottleneck



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

$O(n^3)$!

Normalizing Flows: The Main Bottleneck



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

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

ETH zürich

Continuous Normalizing Flows: Making It Efficient

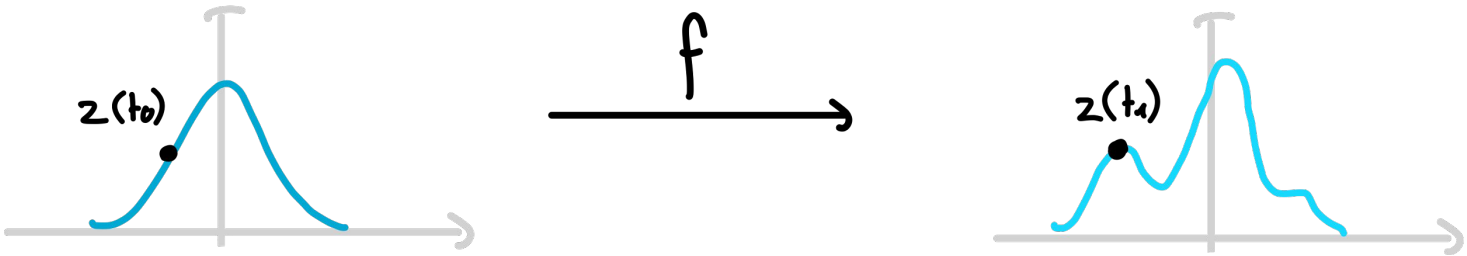
Theorem 1 (Instantaneous Change of Variables)**.** *Let* $\mathbf{z}(t)$ *be a finite continuous random variable with probability* $p(\mathbf{z}(t))$ *dependent on time. Let* $\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)$ *be a differential equation describing a continuous-in-time transformation of* $\mathbf{z}(t)$*. Assuming that* f *is uniformly Lipschitz continuous in* $\mathbf{z}$ *and continuous in* t*, then the change in log probability also follows a differential equation,*

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \tag{8}$$

ETH zürich

Continuous Normalizing Flows: Making It Efficient

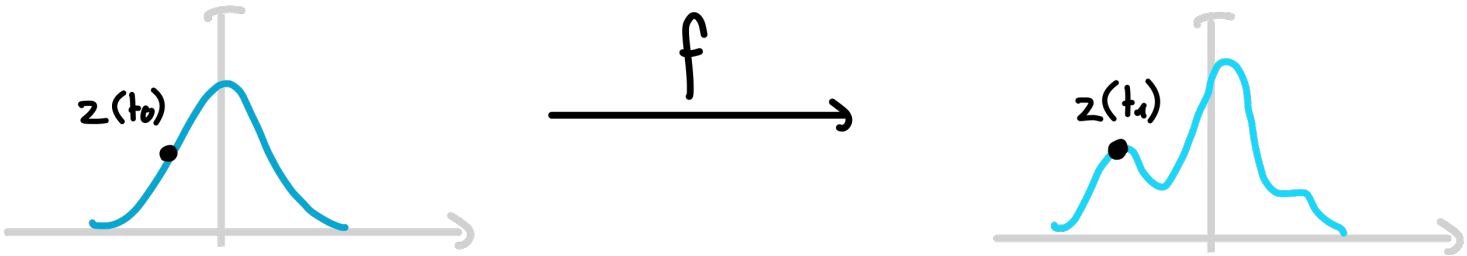
$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$$



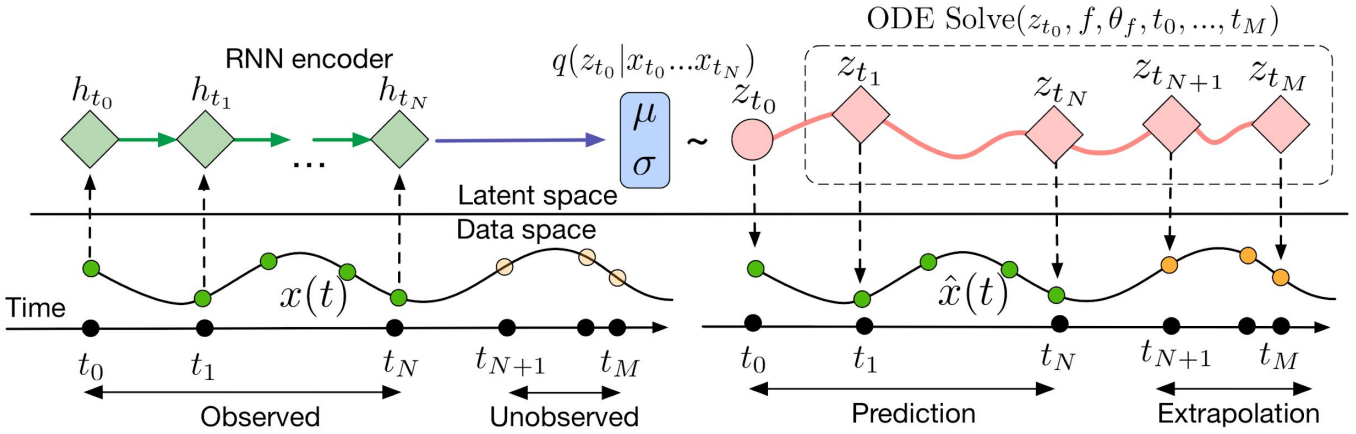
Continuous Normalizing Flows: Making It Efficient

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

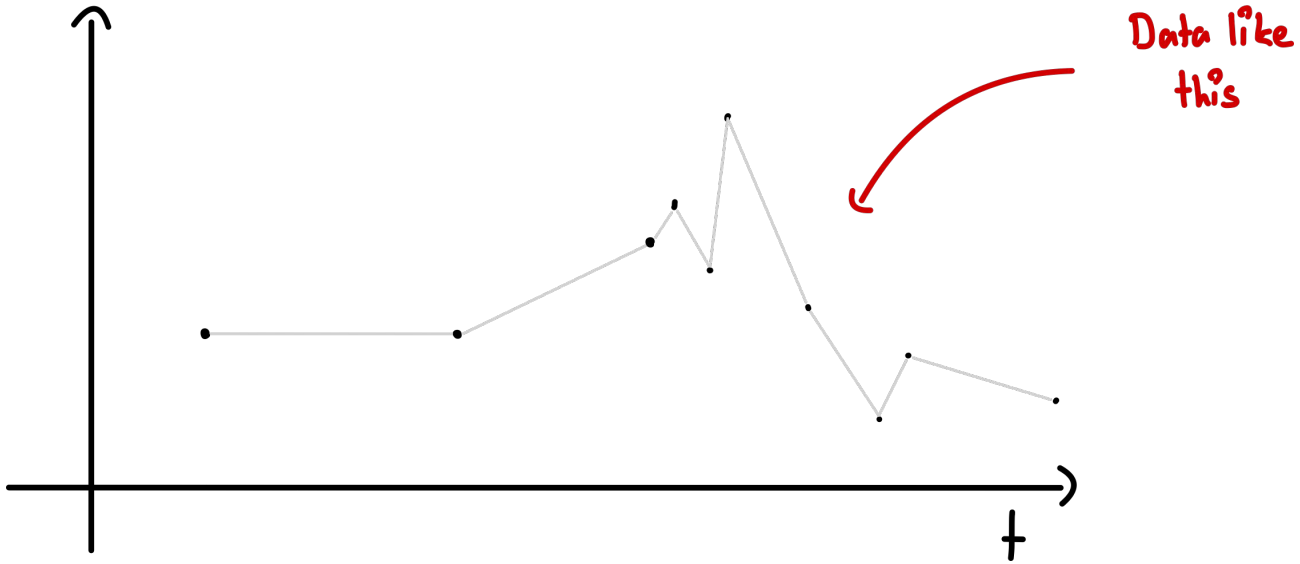
O(n)!!



Generative Latent Time-Series Model

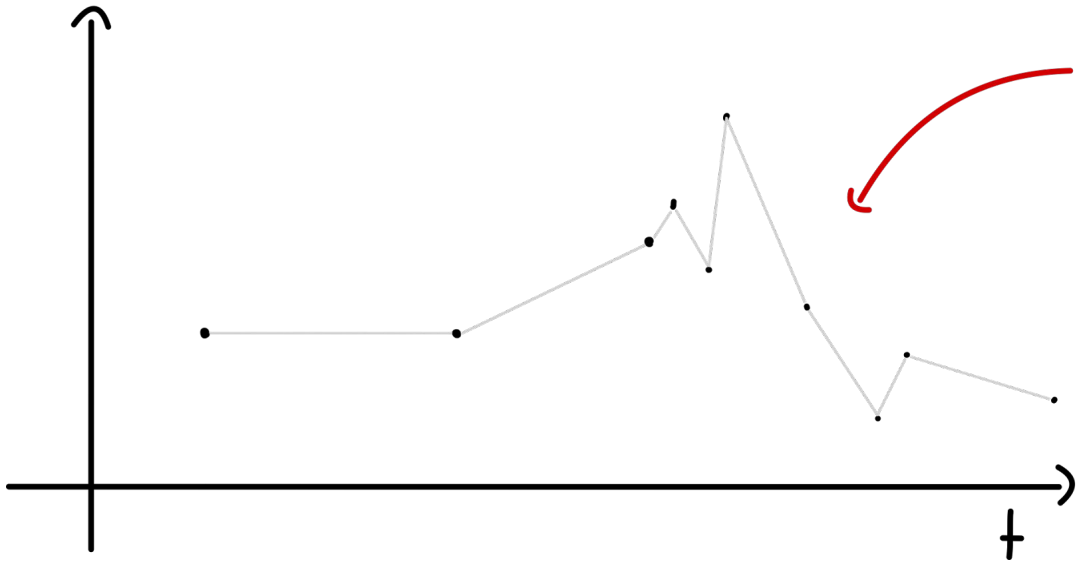


GM: The Problem



ETH zürich

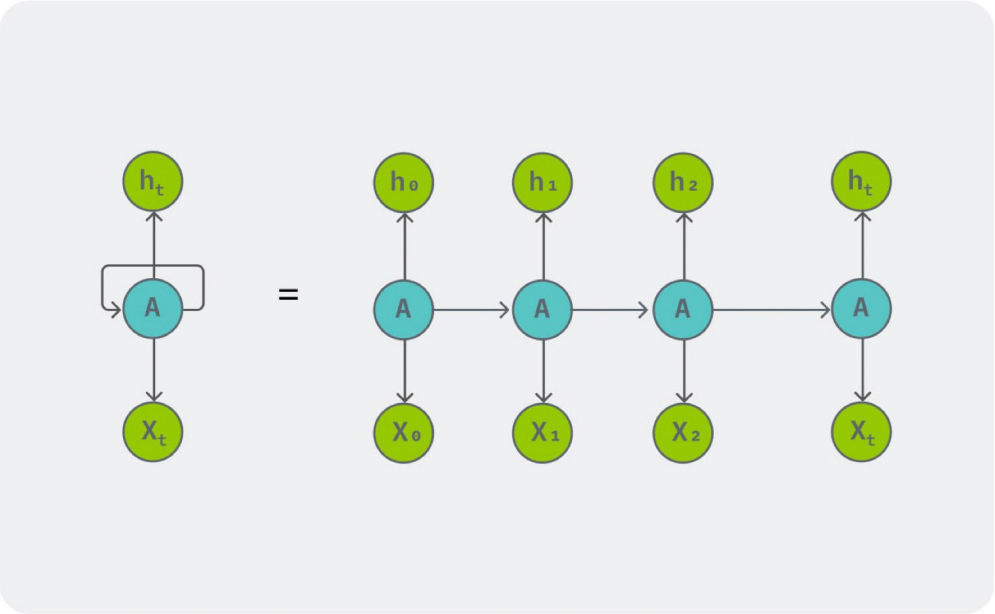
GM: The Problem



Irregular Time-Series Data

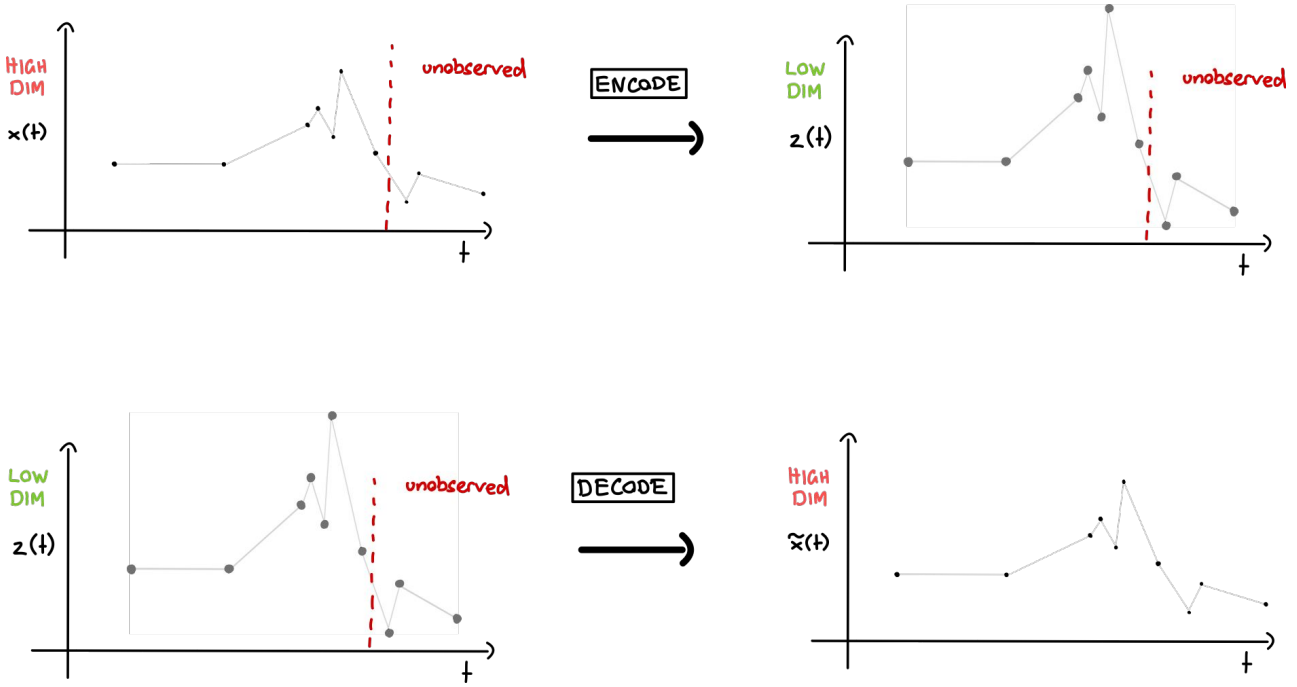
ETH zürich

GM: The Common Solution

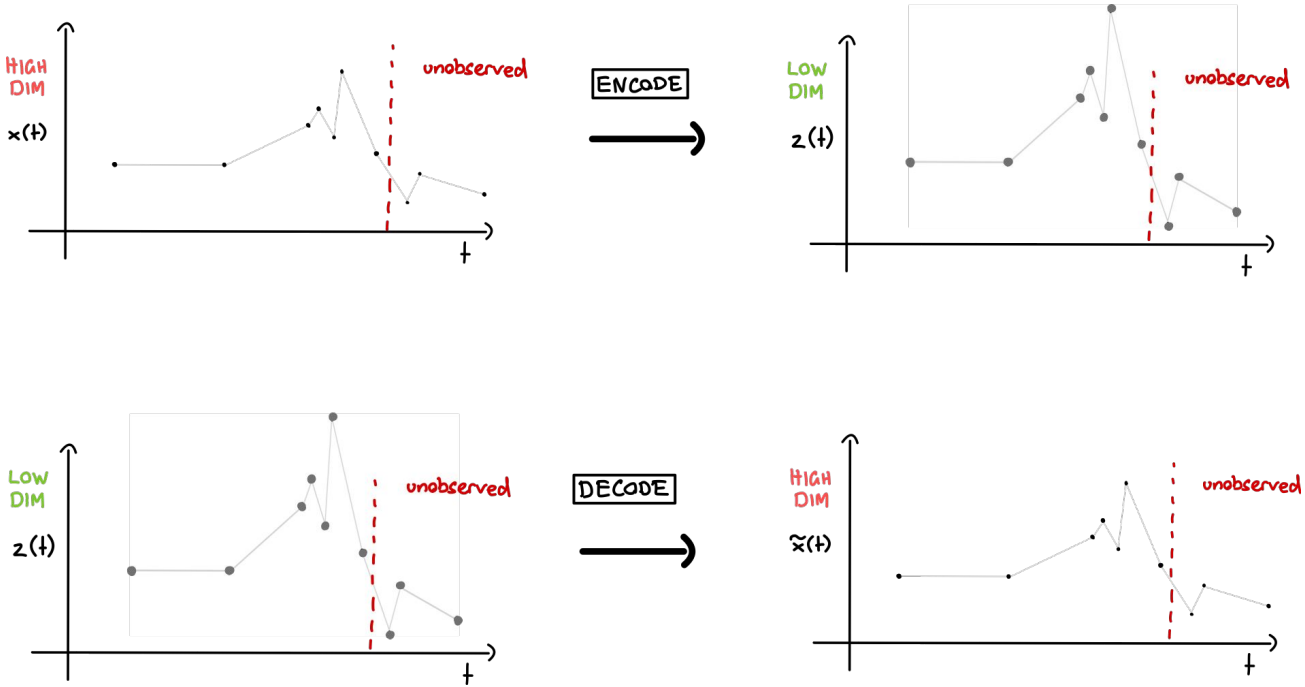


- Using a Recurrent Neural Network (RNN)

ETH zürich

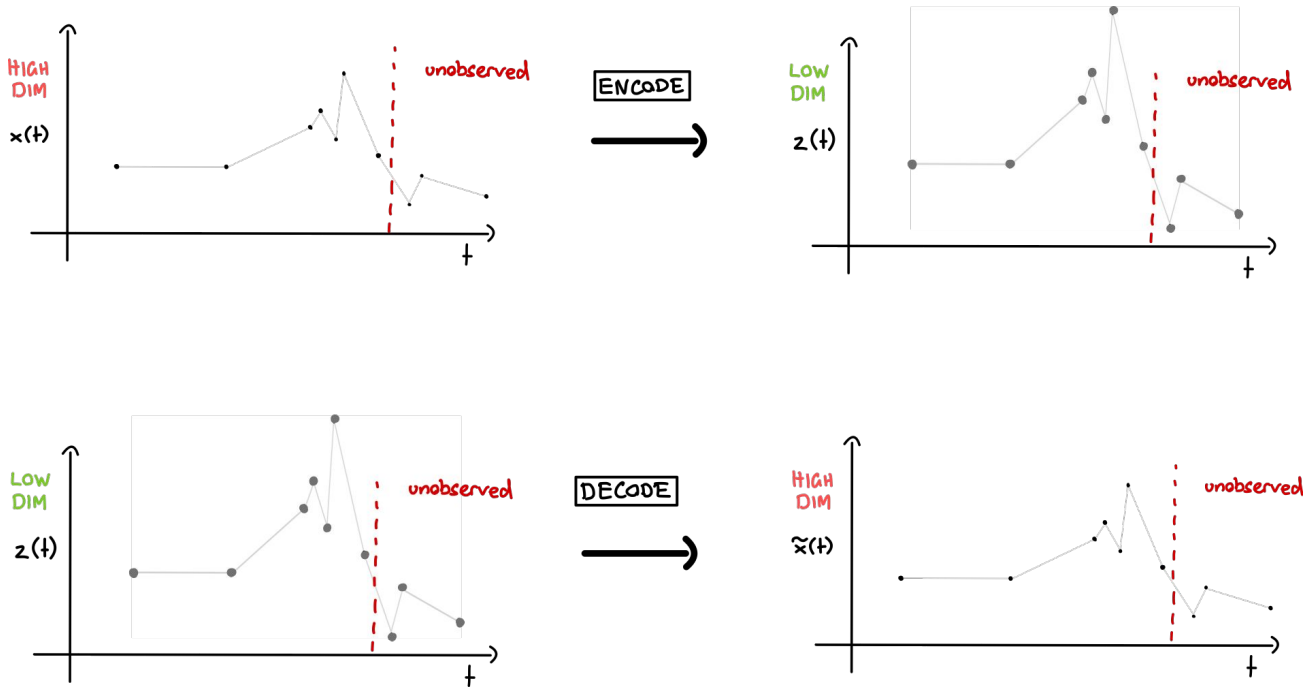

HIGH DIM
x(t)
t
unobserved
ENCODE
LOW DIM
z(t)
t
unobserved
LOW DIM
z(t)
t
unobserved
DECODE
HIGH DIM
x̃(t)
t

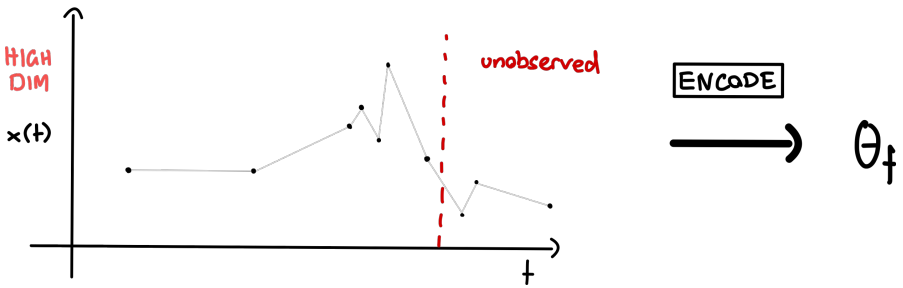
ETH zürich

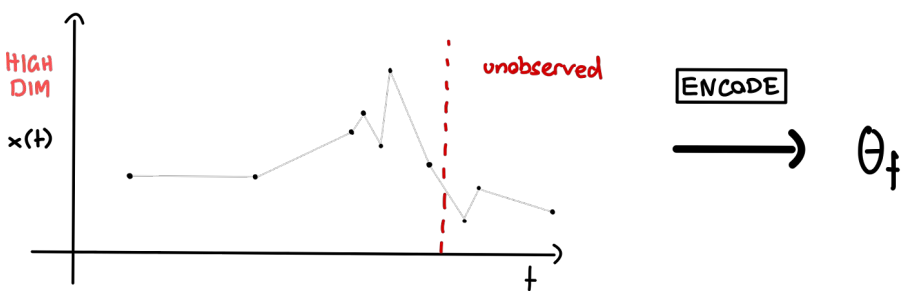

HIGH DIM
x(t)
t
unobserved
ENCODE
LOW DIM
z(t)
t
unobserved
LOW DIM
z(t)
t
unobserved
DECODE
HIGH DIM
x̃(t)
t
unobserved

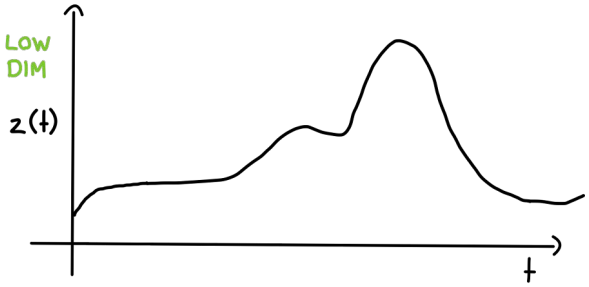
ETH zürich

GM: Autoencoders

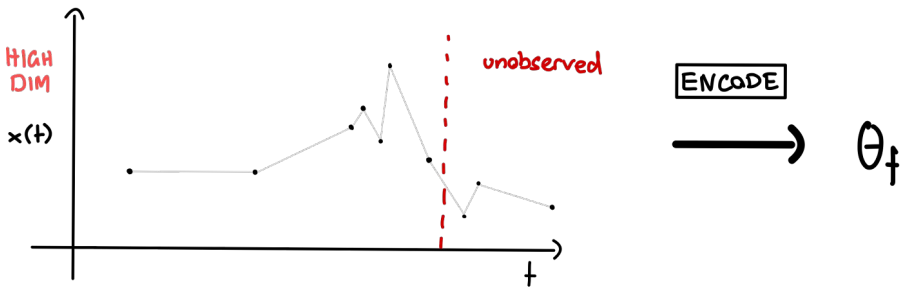



HIGH DIM
x(t)
t
unobserved
ENCODE
θ_t



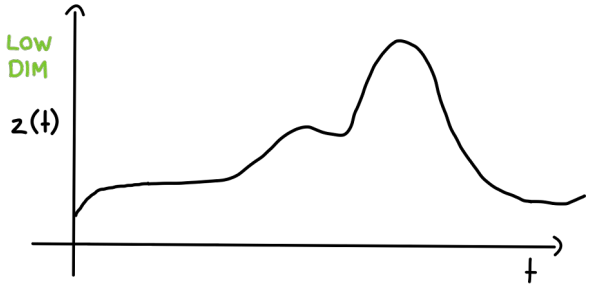


ETH zürich

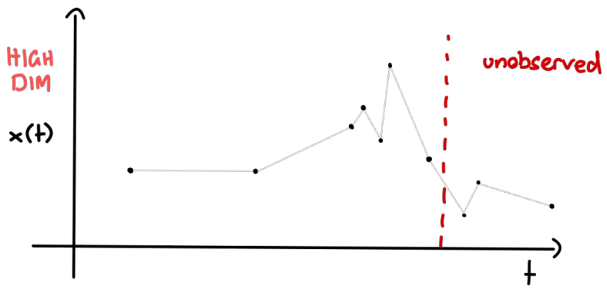


ENCODE

θ_f

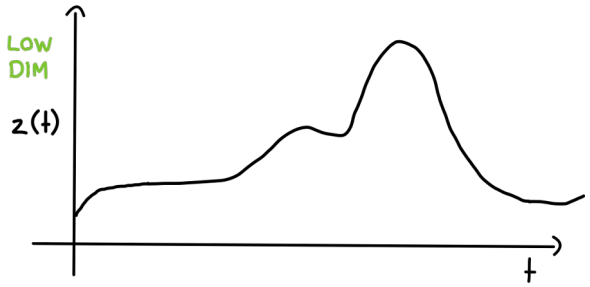


$\Rightarrow$ described by f



ENCODE

θ_f

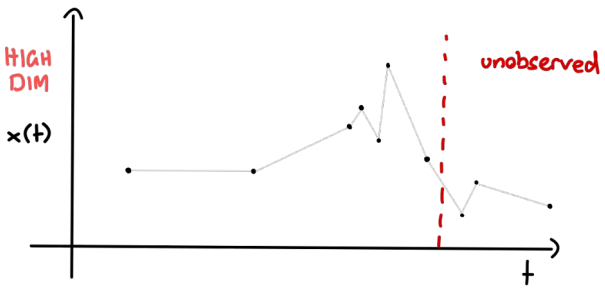


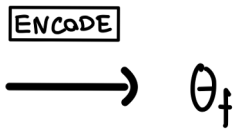
$\Longrightarrow$ described by f

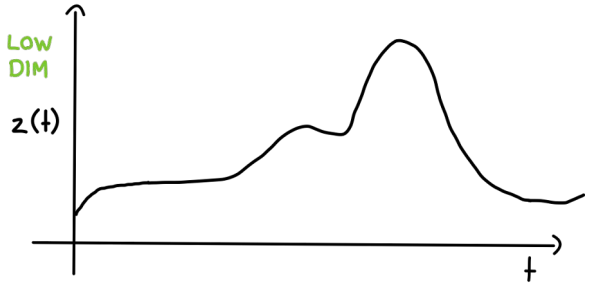
$$\frac{dz(t)}{dt} = f(z(t), t, \theta)$$

ETH zürich

GM: Variational Autoencoders





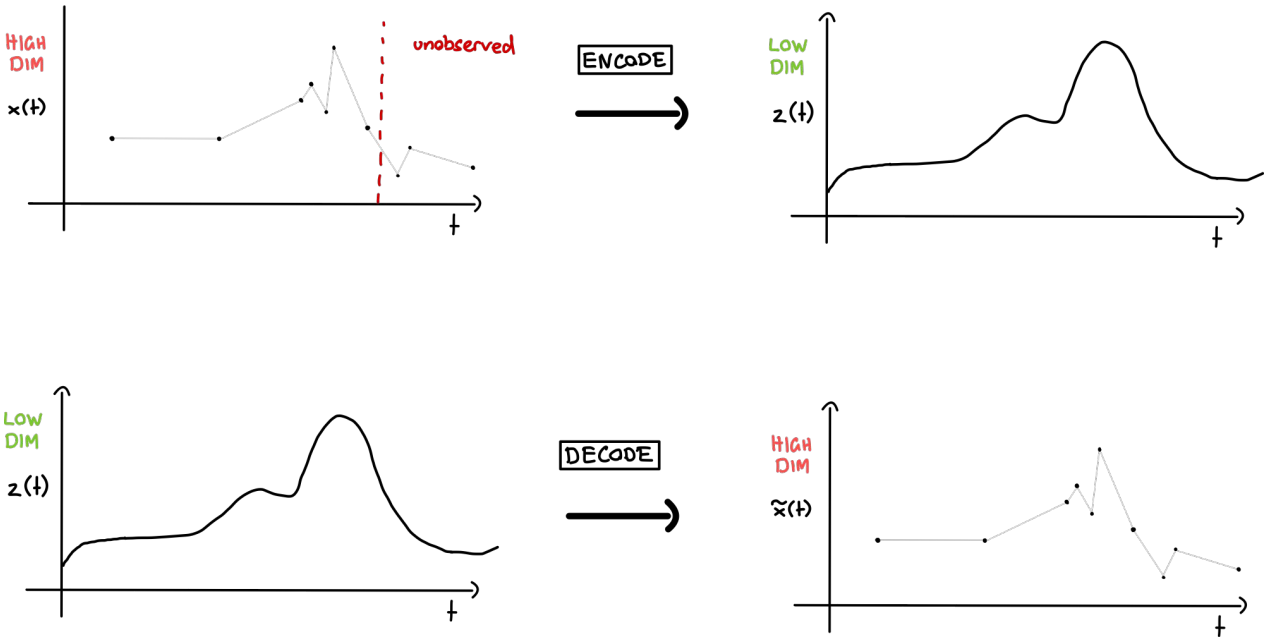


$\Rightarrow$ described by f

$$\frac{dz(t)}{dt} = f(z(t), t, \theta)$$

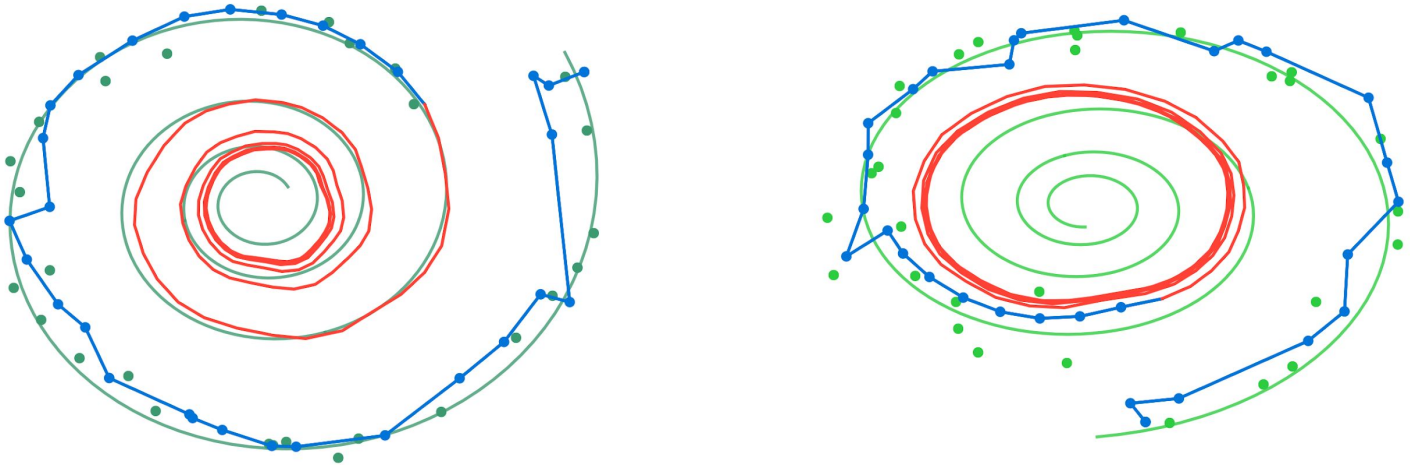
ETH zürich

GM: Variational Autoencoders



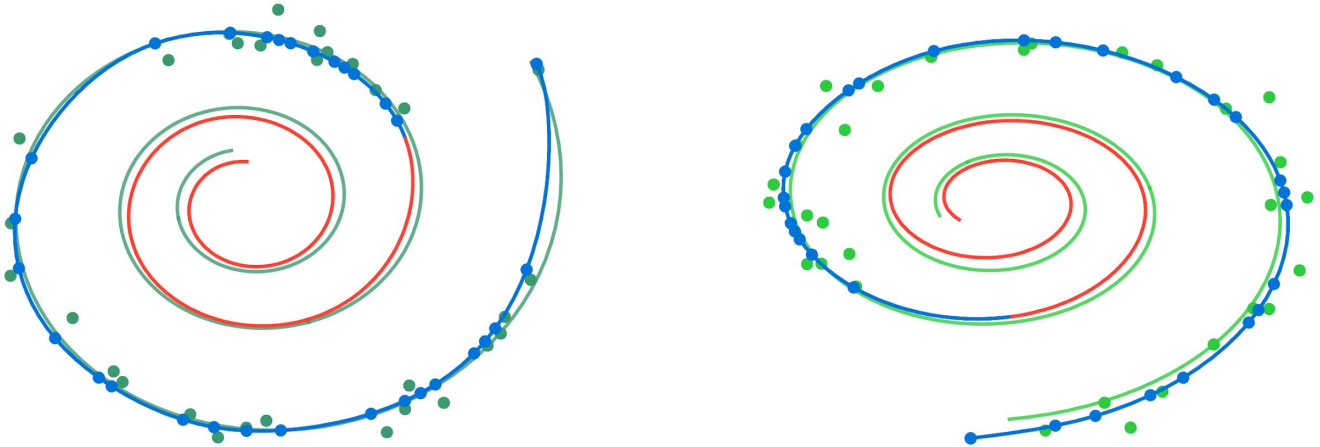
ETH zürich

GM: RNN Results



ETH zürich

GM: Results with our Generative Model



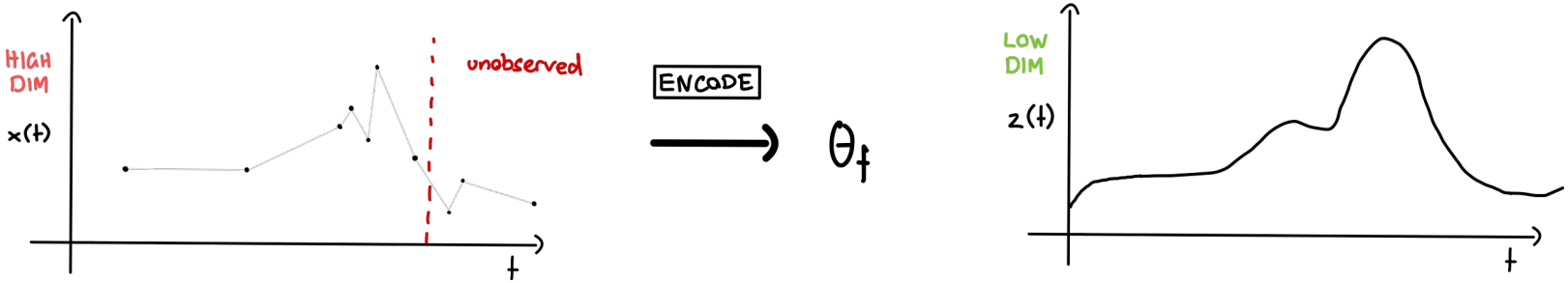
Criticism

Criticism

- Experiments are only performed on toy-ish datasets (MNIST)

Criticism

- Experiments are only performed on toy-ish datasets (MNIST)
- Using an RNN for the Encoding part of the Generative Model



Open Questions: Robustness

ON ROBUSTNESS OF NEURAL ORDINARY DIFFERENTIAL EQUATIONS

Hanshu Yan*, Jiawei Du*, Vincent Y. F. Tan & Jiashi Feng
Department of Electrical and Computer Engineering
National University of Singapore
{hanshu.yan, dujiawei}@u.nus.edu, {vtan, elefjia}@nus.edu.sg

EFFICIENT CERTIFIED TRAINING AND ROBUSTNESS VERIFICATION OF NEURAL ODES

Mustafa Zeqiri, Mark Niklas Müller, Marc Fischer & Martin Vechev
Department of Computer Science
ETH Zurich, Switzerland
mzeqiri@ethz.ch, {mark.mueller,mark.fischer,martin.vechev}@inf.ethz.ch

- Shown to be more robust than CNNs
- Still unclear for high-dimensional datasets

ETH zürich

Further Research: Neural ODE Processes

Problem: Neural ODE's don't adapt well to new incoming data without the entire model being retrained

Solution:

NEURAL ODE PROCESSES

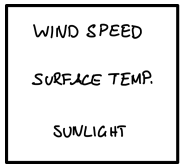
Alexander Norcliffe* †
Department of Computer Science
University College London
London, United Kingdom
ucabino@ucl.ac.uk

Cristian Bodnar*, Ben Day*, Jacob Moss* & Pietro Liò
Department of Computer Science
University of Cambridge
Cambridge, United Kingdom
{cb2015, bjd39, jm2311, pl219}@cam.ac.uk

ETH zürich

Further Research: Identification and Interventions

They introduced a method that can recover the governing ODEs from data alone



$$\frac{dS}{dt} = Q - \lambda S$$

Beyond Predictions in Neural ODEs: Identification and Interventions

Hananeh Aliee
Helmholtz Center, Munich

Fabian J. Theis
TUM, Helmholtz Center, Munich

Niki Kilbertus
Helmholtz Center, Munich

{hananeh.aliee,fabian.theis,niki.kilbertus}@helmholtz-muenchen.de

ETH zürich

Future Applications

Forecasting virus outbreaks with social media data via neural ordinary differential equations

Matías Núñez, Nadia L. Barreiro, Rafael A. Barrio, Christopher Rackauckas

doi: https://doi.org/10.1101/2021.01.27.21250642

July 3, 2020

#machine learning | #julia | #ode | #time series

Forecasting the weather with neural ODEs

Weather forecasting is a tricky problem. Traditionally, it has been done by manually modelling weather dynamics using differential equations, but this approach is highly dependent on us getting the equations right. To avoid this problem, we can use machine learning to directly predict the weather, which let's us make predictions without modelling the dynamics. However, this approach requires huge amounts of data to reach good performance. Fortunately, there is a middle ground: What if we instead use machine learning to model the *dynamics* of the weather?

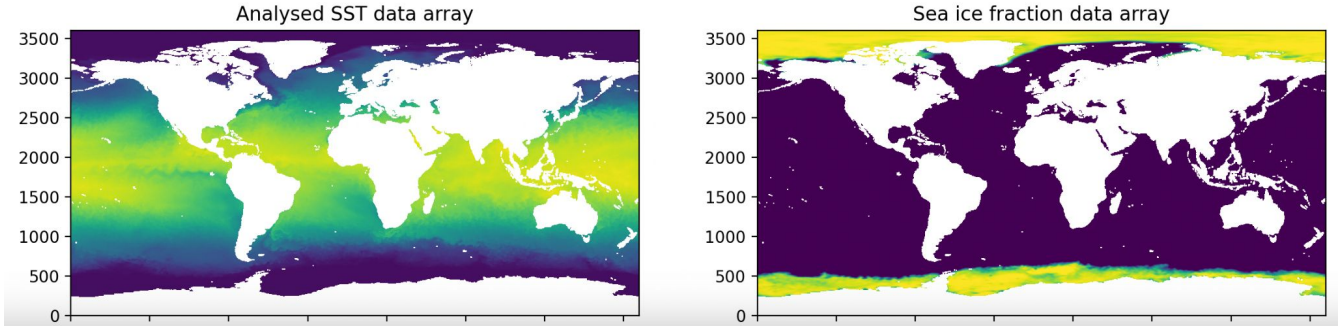
Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems

Alec J. Linot [a], Joshua W. Burby [b], Qi Tang [b], Prasanna Balaprakash [c], Michael D. Graham [a], Romit Maulik [c]

ETH zürich

Future Applications: An Idea

- Use Sea Surface Temperature data to feed into a generative latent time-series model
- ESA dataset



ETH zürich

Thank you for your attention!

ETH zürich