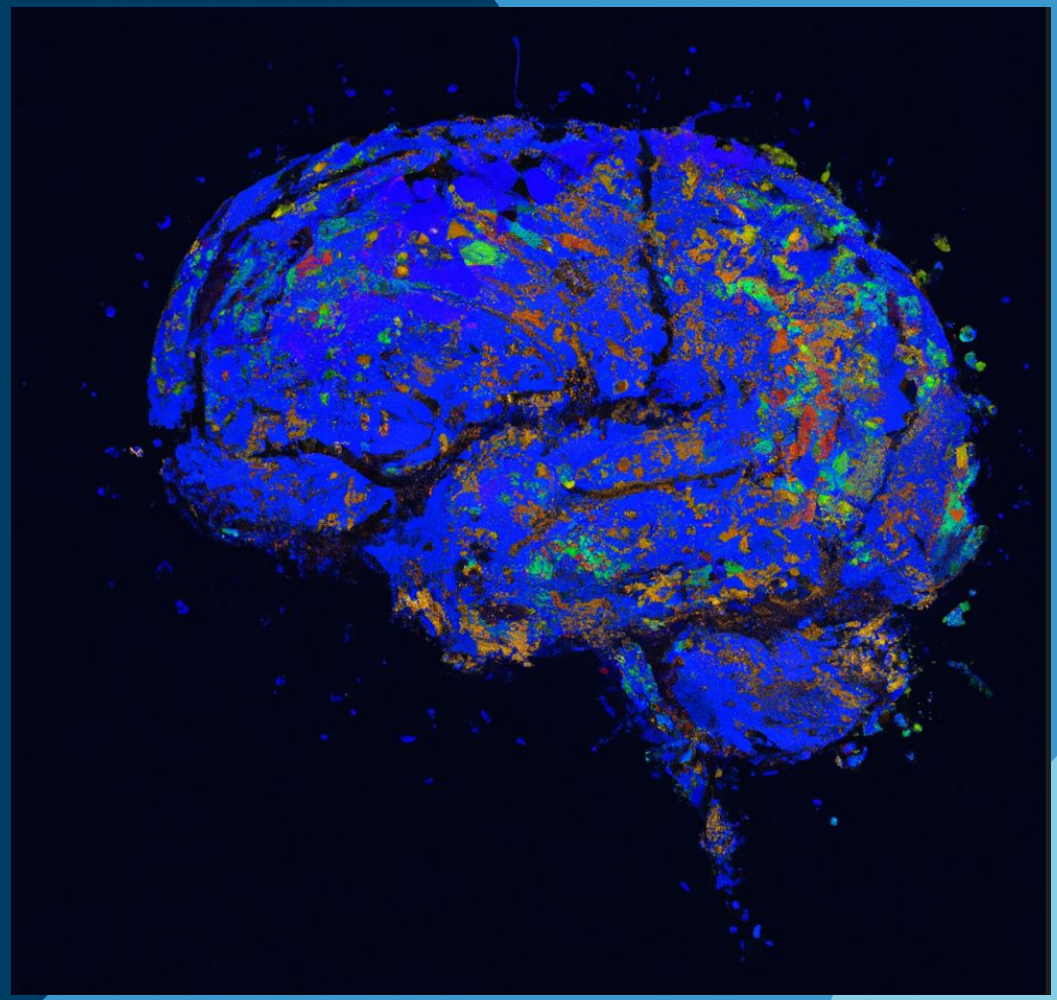
Zentralinstitut
für Seelische
Gesundheit

CNS 2023



A Guide to Reconstructing Dynamical Systems from Neural Measurements Using Recurrent Neural Networks

Max Thurm, Manuel Brenner, Florian Hess, Alena Brändle,
Lukas Eisenmann, Janik Fechtelpeter
Daniel Durstewitz

https://durstewitzlab.github.io/cns-2023-tutorial/

Who we are

Agenda

- Theory of dynamical systems reconstruction (DSR)
  - Introduction to (deep) DSR
  - Machine learning approaches for DSR
  - Analysis of trained models
- Application on real-world data
  - Practical considerations for training on imperfect data
  - Observation model for fMRI data
  - Non-stationarity during rule learning
- Hands-on tutorial

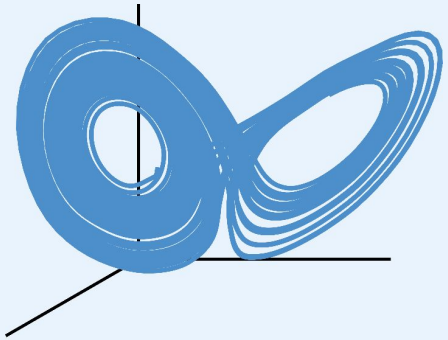
Introduction to
Deep Dynamical Systems Reconstruction

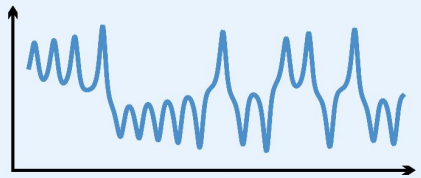
Max Ingo Thurm

Deep Dynamical Systems Reconstruction

dynamical system

$$\dot{x}_1 = a(x_2 - x_1)$$
$$\dot{x}_2 = x_1(b - x_3) - x_2$$
$$\dot{x}_3 = x_1 x_2 - c x_3$$

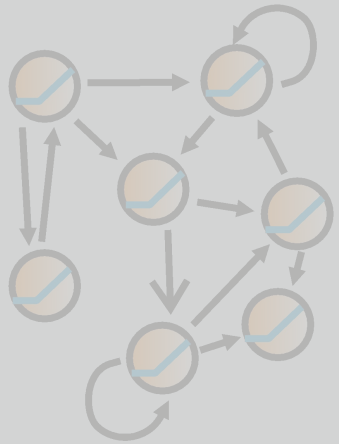




train

RNN

$$z_t = F_\theta(z_{t-1}, s_t)$$



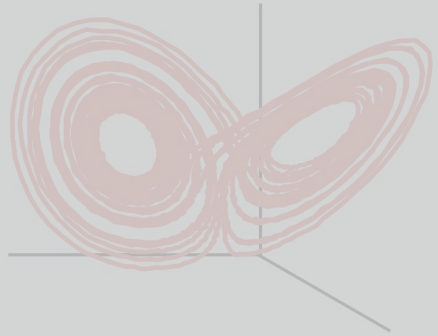
universal approximator of dynamical systems

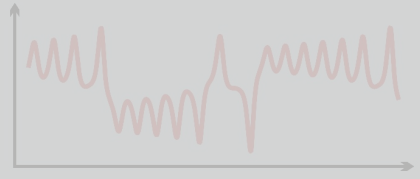
(Hanson & Raginsky, 2020)

generate

simulated trajectory

$$z_t = A z_{t-1} + W \max(z_{t-1}, 0) + \mathrm{h}$$



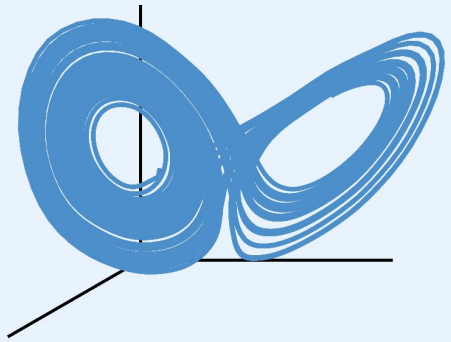


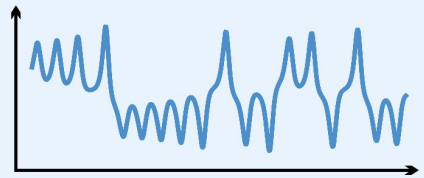
Schmidt*, Koppe* et al. 2021, *ICLR*

Deep Dynamical Systems Reconstruction

dynamical system

$$\dot{x}_1 = a(x_2 - x_1)$$
$$\dot{x}_2 = x_1(b - x_3) - x_2$$
$$\dot{x}_3 = x_1 x_2 - c x_3$$

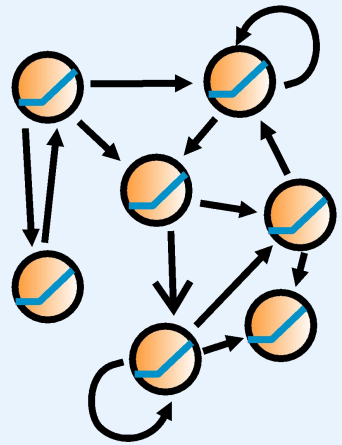




train

RNN

$$\boldsymbol{z}_t = F_\theta(\boldsymbol{z}_{t-1}, \boldsymbol{s}_t)$$



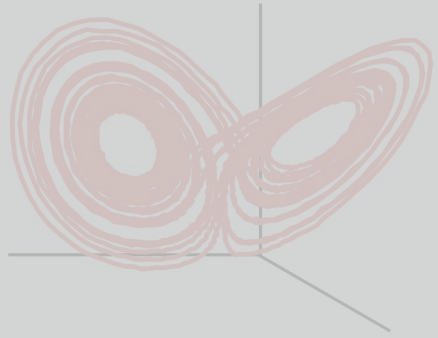
universal approximator of dynamical systems

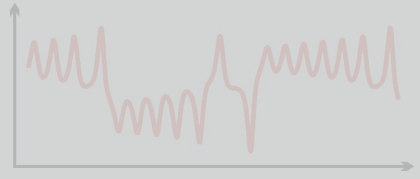
(Hanson & Raginsky, 2020)

generate

simulated trajectory

$$z_t = Az_{t-1} + W\max(z_{t-1}, 0) + \mathrm{h}$$



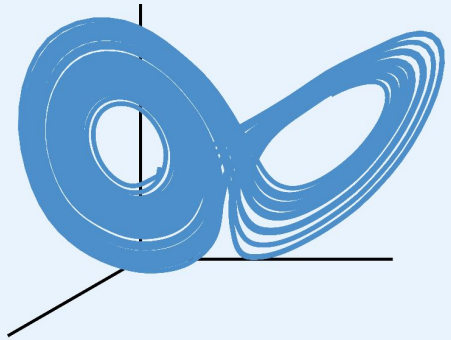


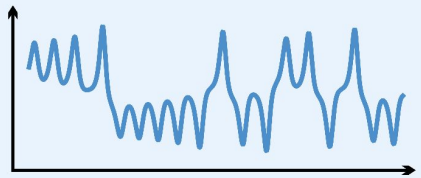
Schmidt*, Koppe* et al. 2021, *ICLR*

Deep Dynamical Systems Reconstruction

dynamical system

$$\dot{x}_1 = a(x_2 - x_1)$$
$$\dot{x}_2 = x_1(b - x_3) - x_2$$
$$\dot{x}_3 = x_1 x_2 - c x_3$$

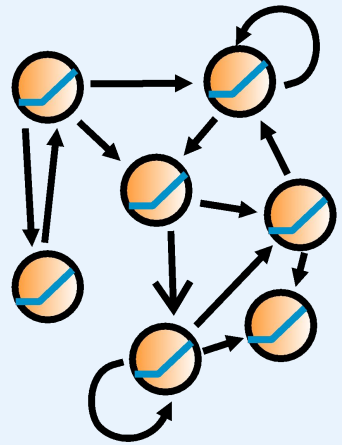




train

RNN

$$\boldsymbol{z}_t = F_\theta(\boldsymbol{z}_{t-1}, \boldsymbol{s}_t)$$



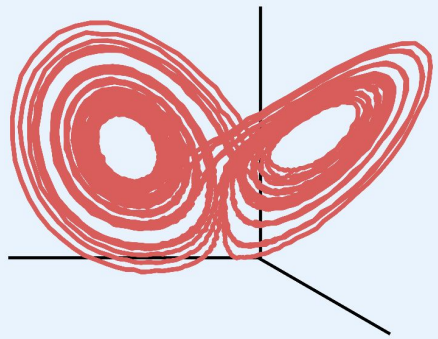
universal approximator of dynamical systems

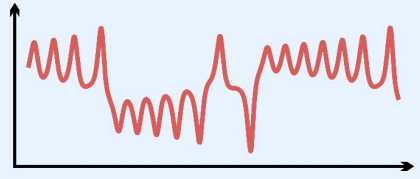
(Hanson & Raginsky, 2020)

generate

simulated trajectory

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{W}\max(\boldsymbol{z}_{t-1}, \boldsymbol{0}) + \mathbf{h}$$



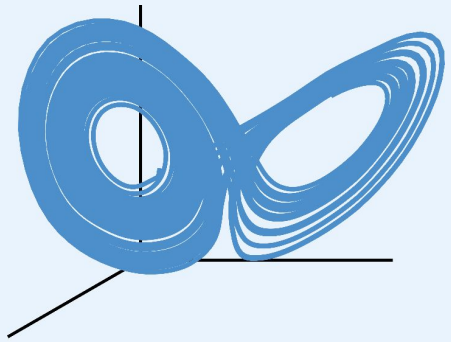


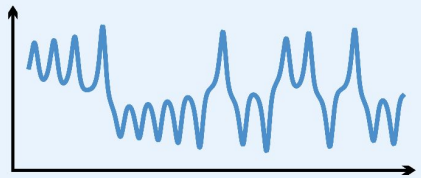
Schmidt*, Koppe* et al. 2021, *ICLR*

Deep Dynamical Systems Reconstruction

dynamical system

$$\dot{x}_1 = a(x_2 - x_1)$$
$$\dot{x}_2 = x_1(b - x_3) - x_2$$
$$\dot{x}_3 = x_1 x_2 - c x_3$$





RNN

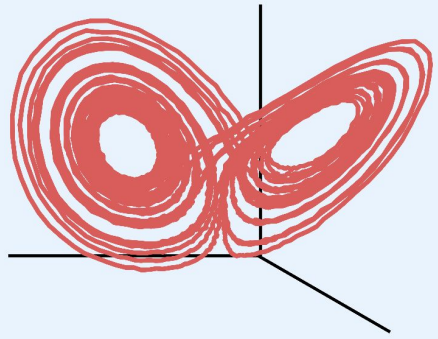
$$z_t = F_\theta(z_{t-1}, s_t)$$

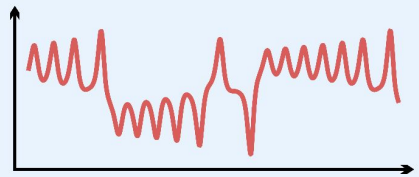
Same geometrical and temporal structure

RNN becomes formal surrogate for real system

simulated trajectory

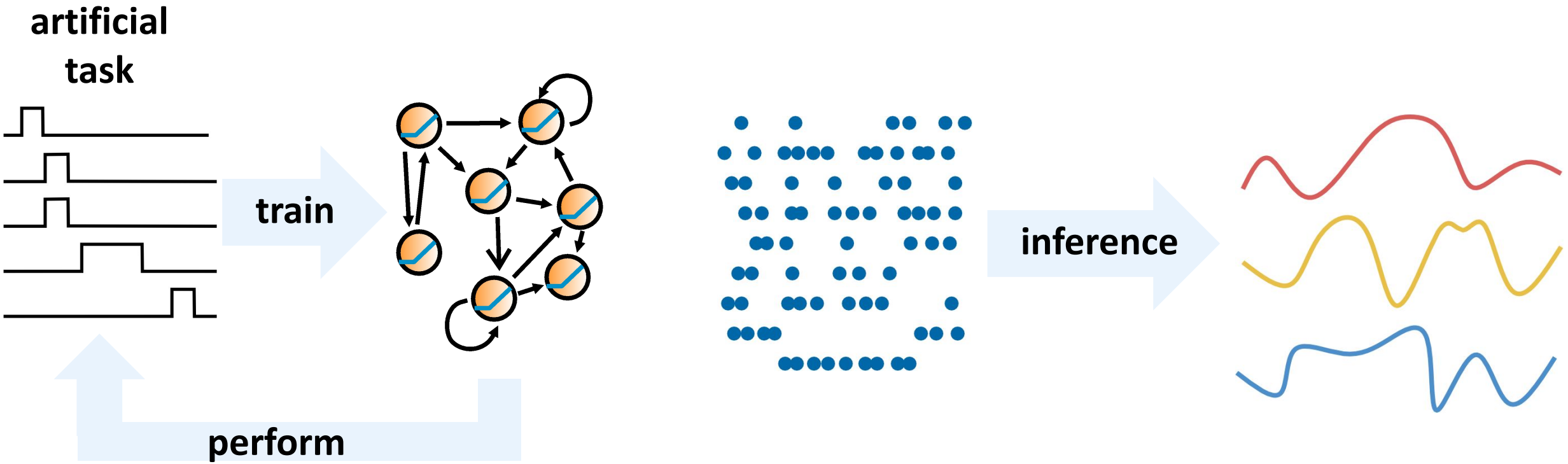
$$z_t = A z_{t-1} + W \max(z_{t-1}, 0) + h$$



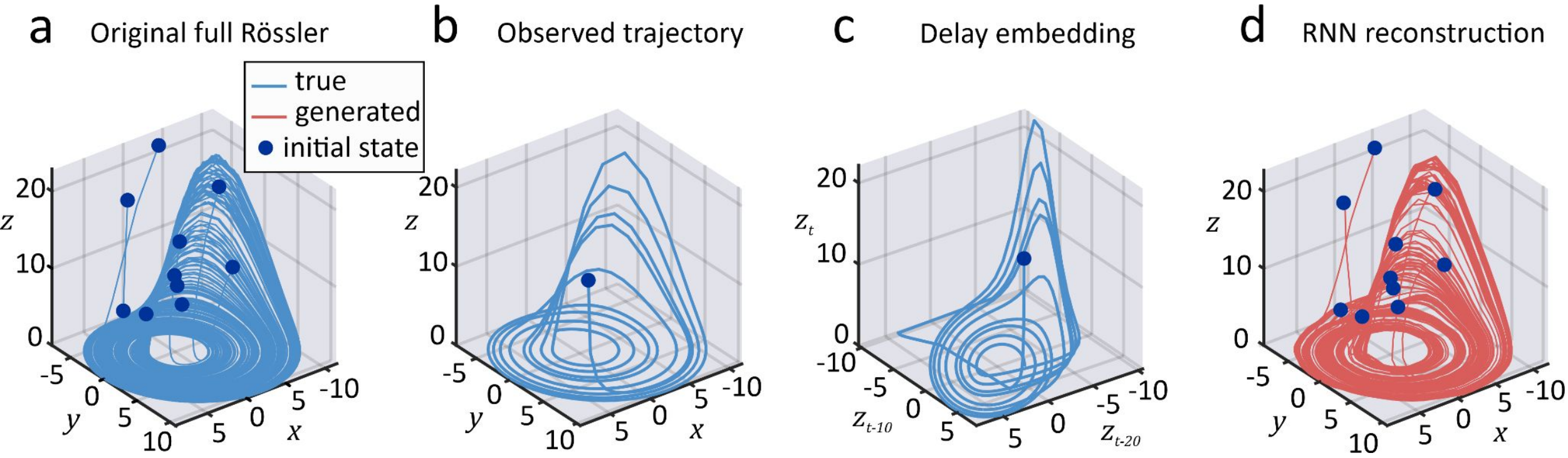


Schmidt*, Koppe* et al. 2021, *ICLR*

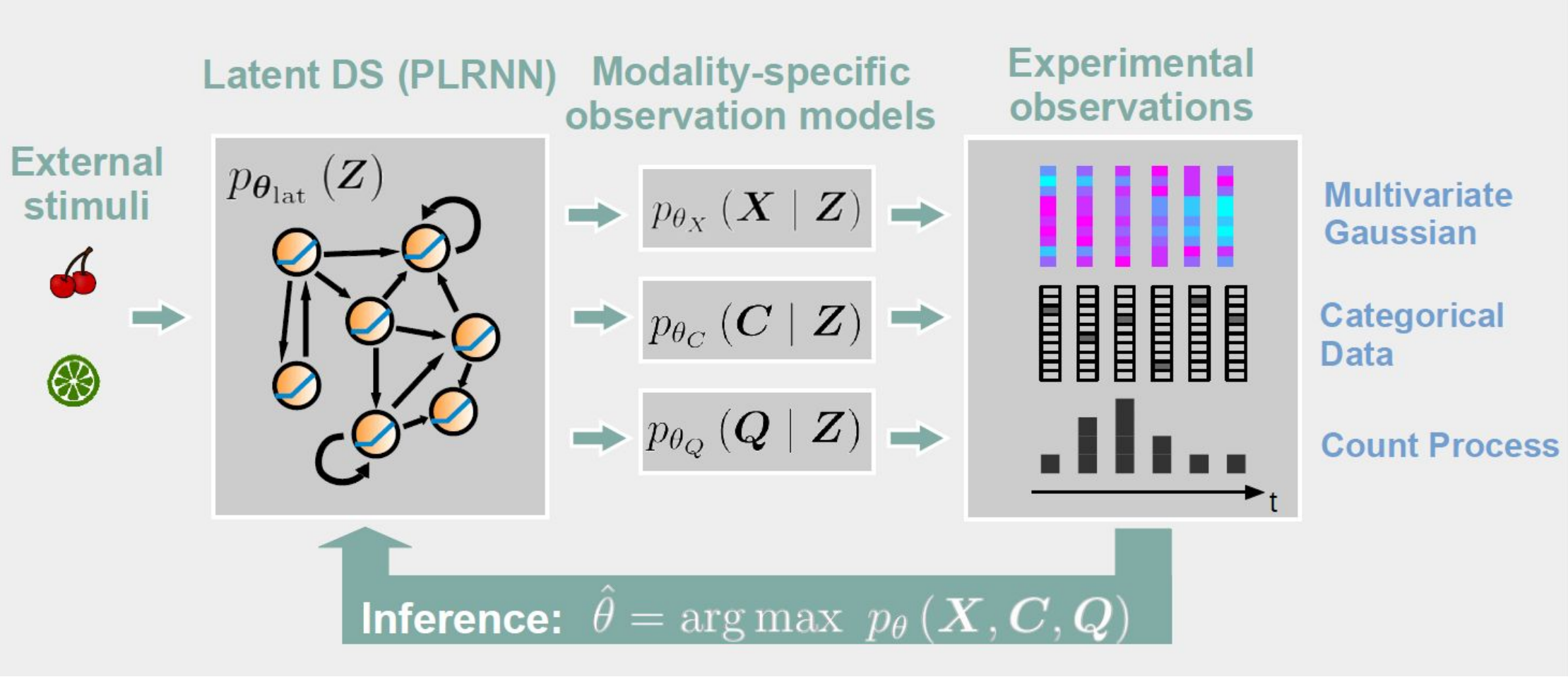
What are we *NOT* doing!!!!!!



Deep Dynamical Systems Reconstruction

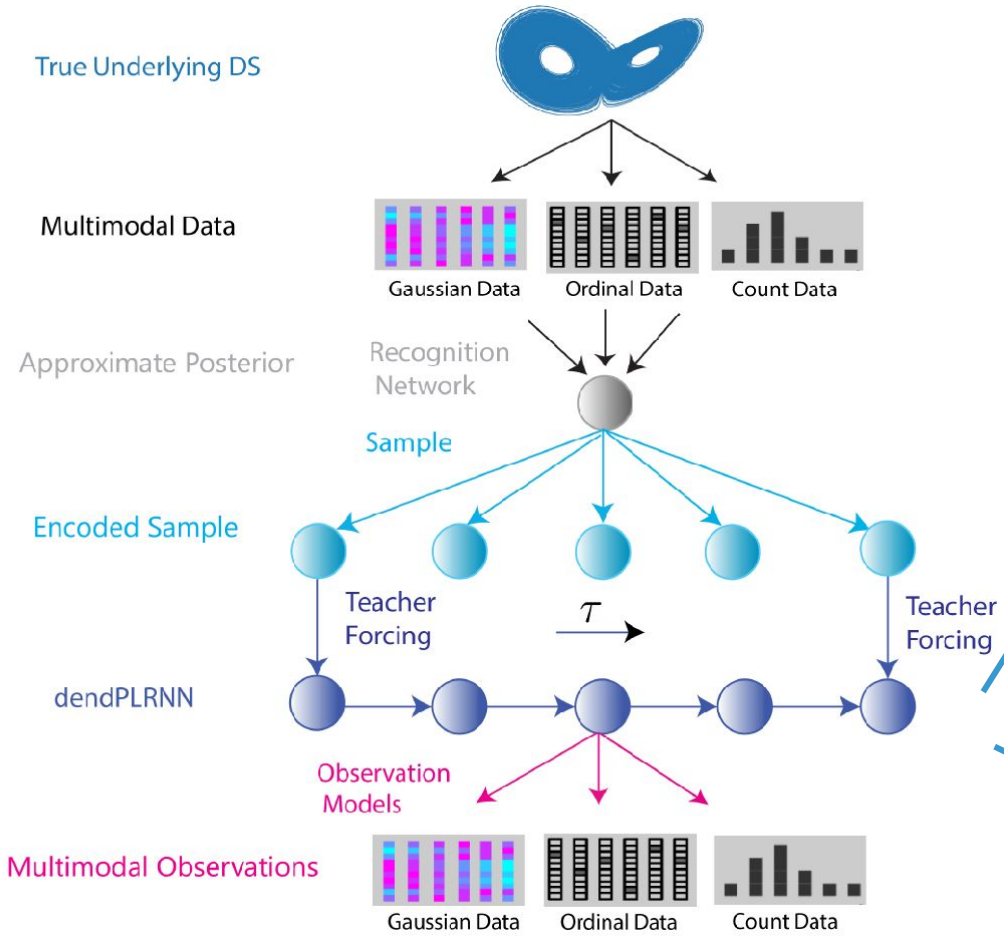


Deep Dynamical Systems Reconstruction by neural flow operators

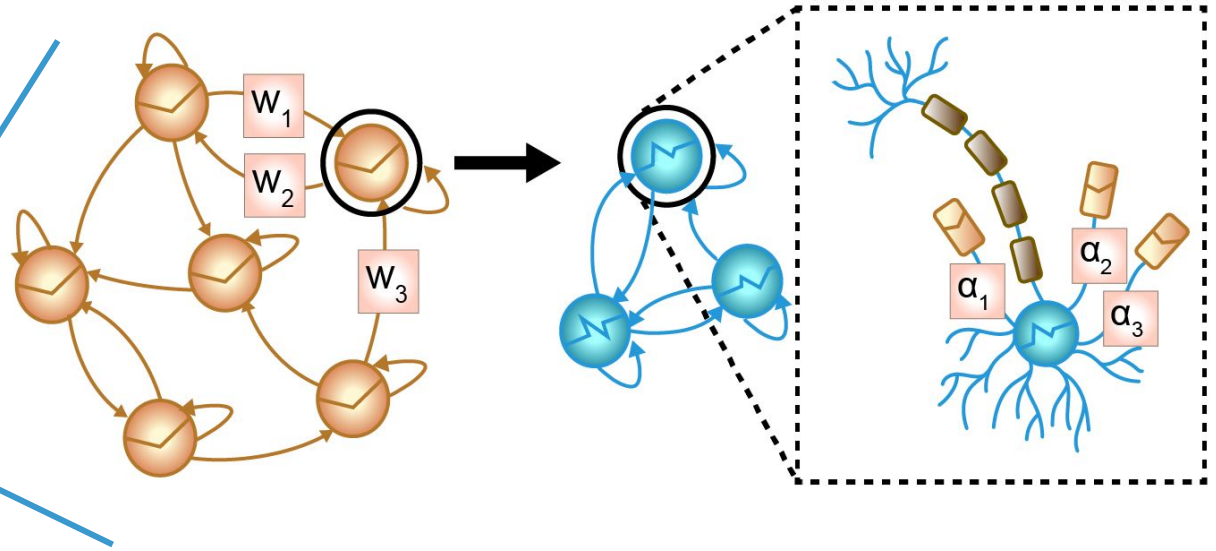


Kramer*, Bommer* et al. ICML 2022

Deep Dynamical Systems Reconstruction by neural flow operators

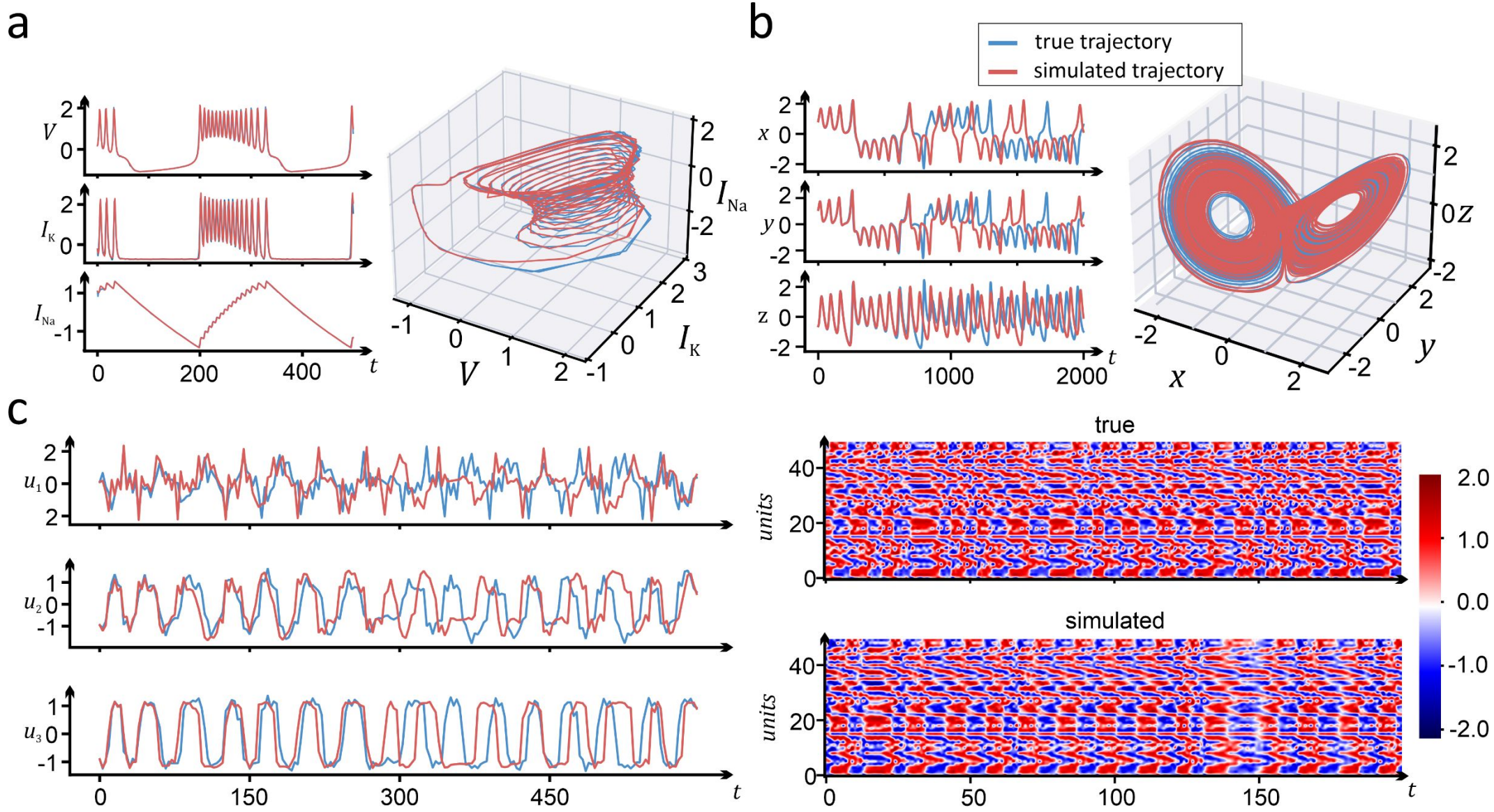


Dendritic PLRNN



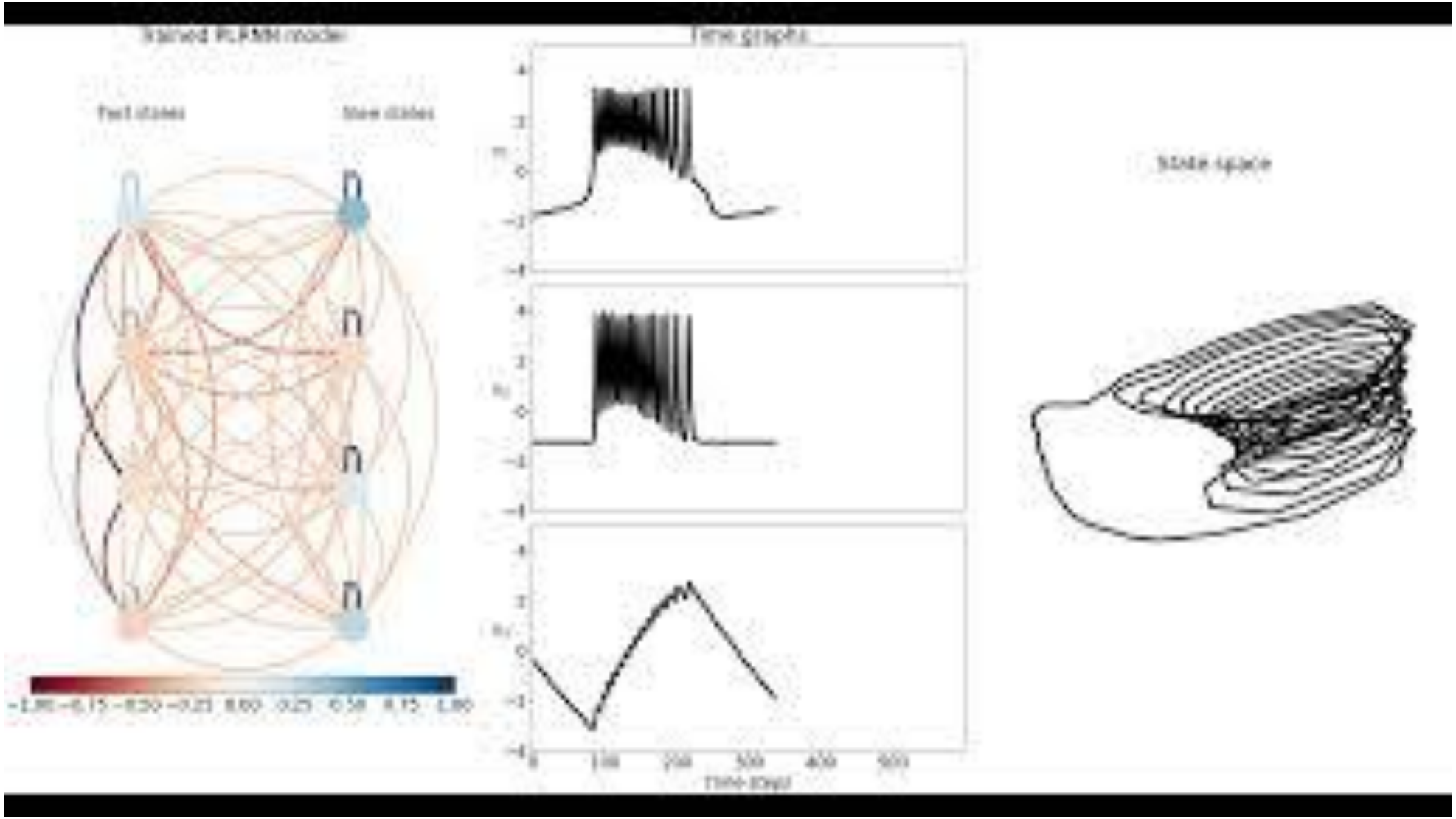
Brenner*, Hess* et al. ICML 2022; Brenner, Koppe, Durstewitz AAAI 2023

Deep Dynamical Systems Reconstruction of benchmark systems



(M. Brenner*, F. Hess*, et al. ICML 2022)

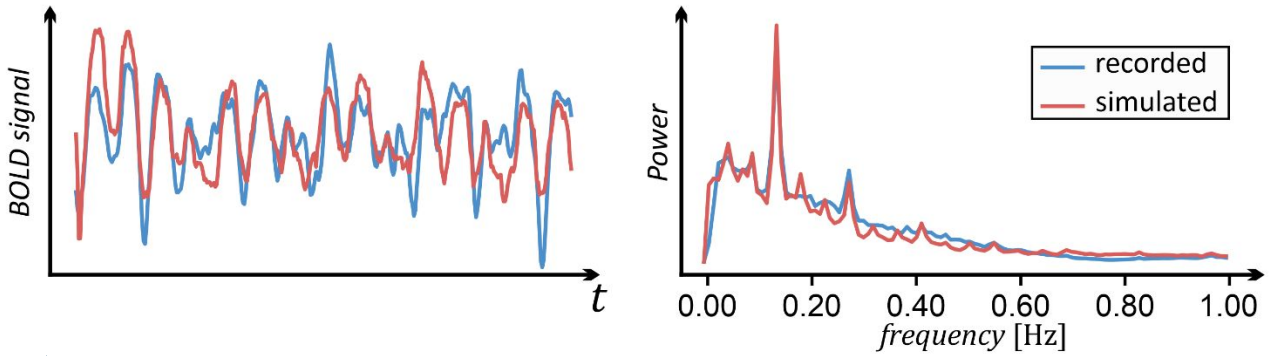
Time scale separation in state space

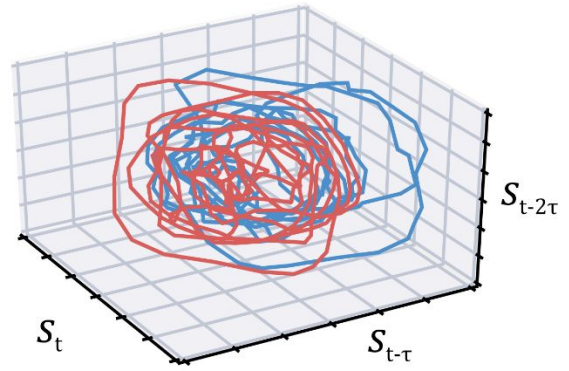


Video: Manuel Brenner & Max Thurm https://youtu.be/taO2OAsqs40

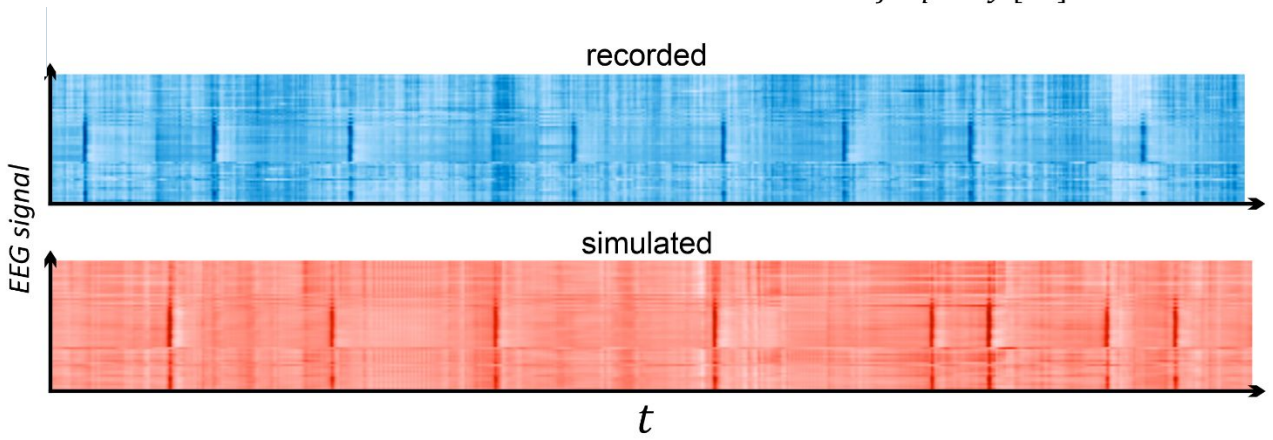
Dynamical systems reconstrcution from physiological data

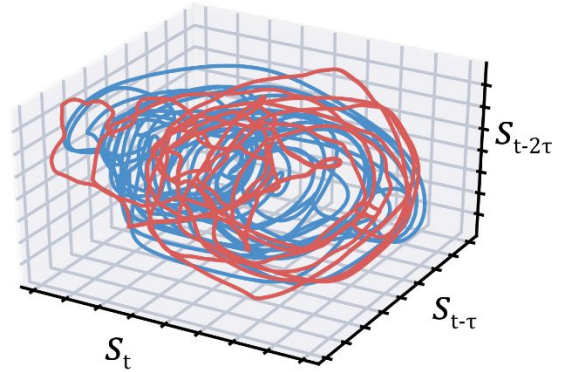
fMRI



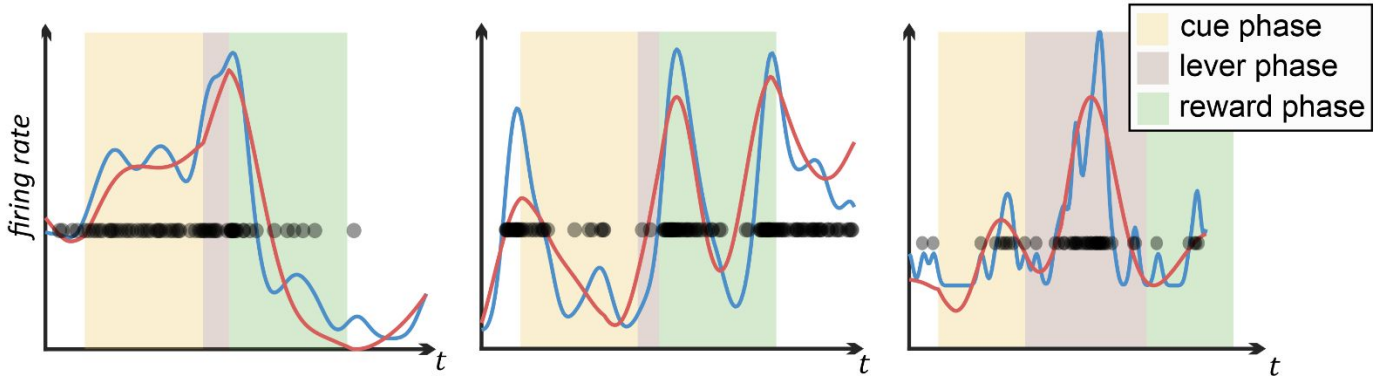


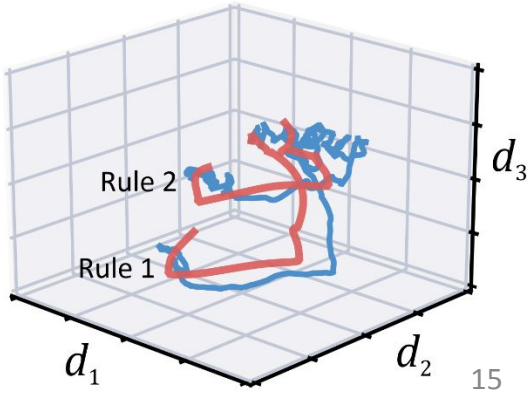
EEG





Single Units





Analysis of Trained Models

Wilson-Cowan model (neural population dynamics)

Analytical computation of fixed points and k-cycles

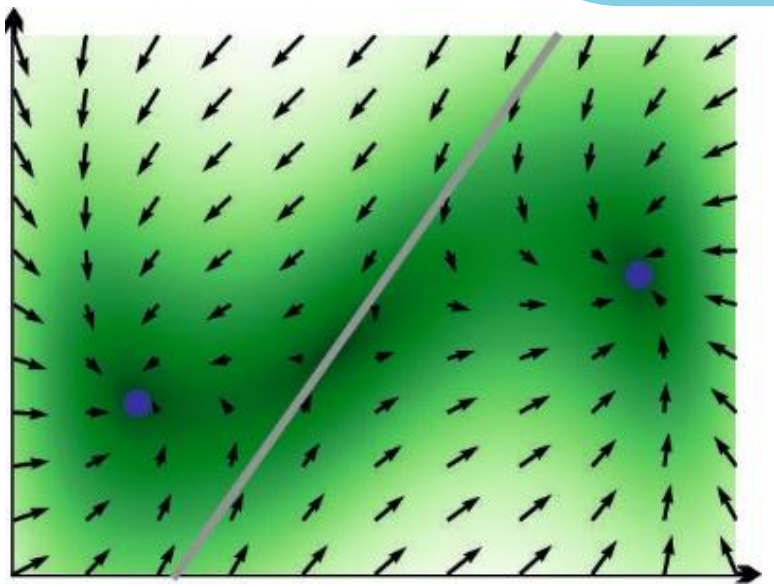
Fixed points

$$z^{*1} = \left(I - W^B_{\Omega(t^{*1})}\right)^{-1}\left[W\, h^B_{\Omega(t^{*1})} + h_0\right]$$

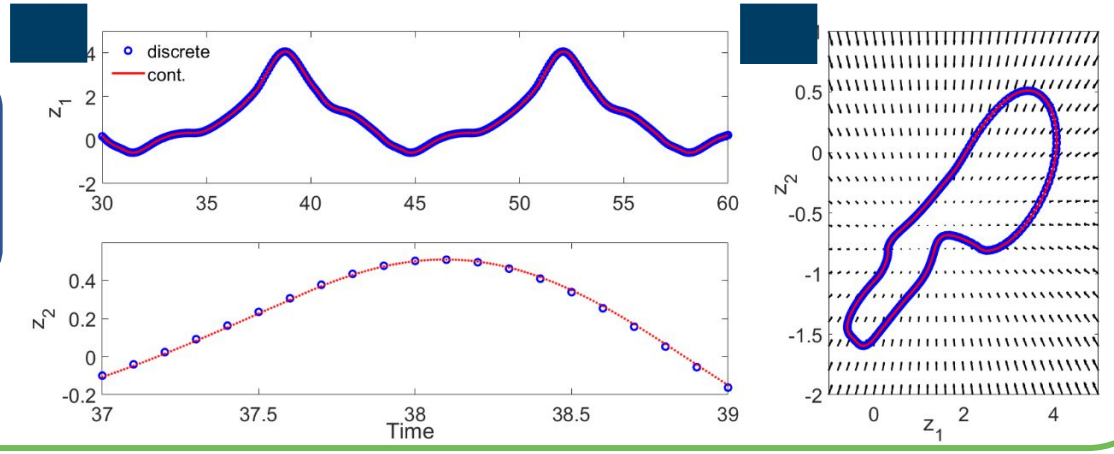
k-cycles

$$z^{*n} = \left(I - \prod_{i=1}^{n} W^B_{\Omega(t^{*n}-i)}\right)^{-1}\left(\sum_{j=2}^{n}\left[\prod_{i=1}^{n-j+1} W^B_{\Omega(t^{*n}-i)} W h^B_{\Omega(t^{*n}-n+j-2)}\right]\right.$$
$$\left. + W h^B_{\Omega(t^{*n}-1)} + \left(\sum_{j=2}^{n}\prod_{i=1}^{n-j+1} W^B_{\Omega(t^{*n}-i)} + I\right) h_0\right),$$

Tractability



Transformation into equivalent continuous time ODE systems (Monfared & Durstewitz, ICML 2020)



Machine Learning Approaches for DSR

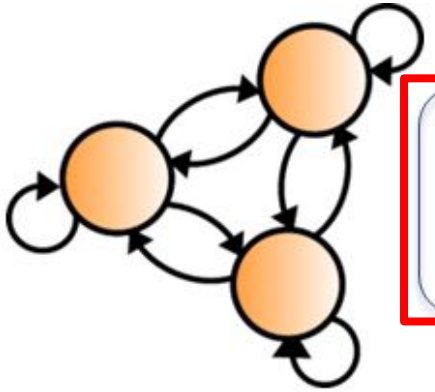
Manuel Brenner

Dynamical Systems Reconstruction

Unknown DS

$$\dot{x}_1 = s(x_2 - x_1)$$
$$\dot{x}_2 = rx_1 - x_2 - x_1x_3$$
$$\dot{x}_3 = x_1x_2 - bx_3$$

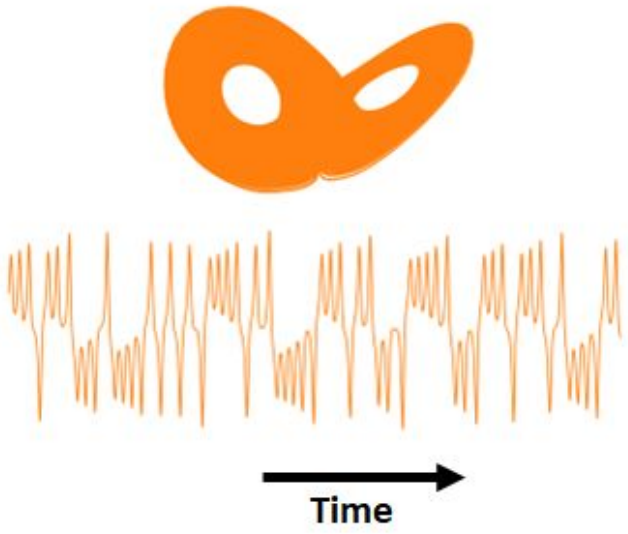
Inference



Generic reconstruction model

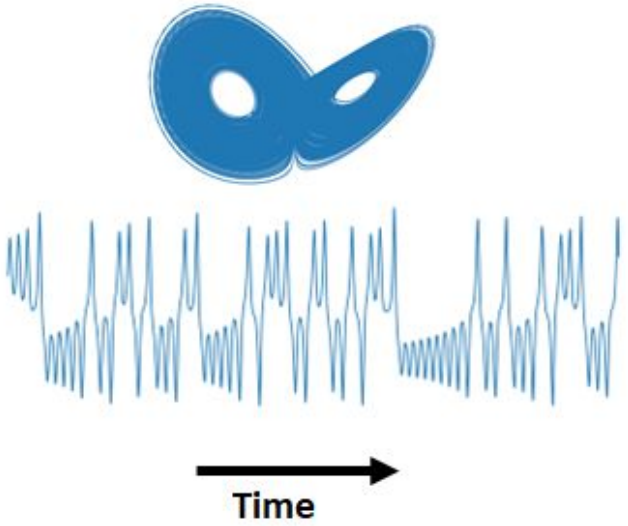
$$\mathbf{z}_t = \boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{z}_{t-1}, \boldsymbol{s}_t)$$
$$\mathbf{x}_t = g(\mathbf{z}_t)$$

True trajectory



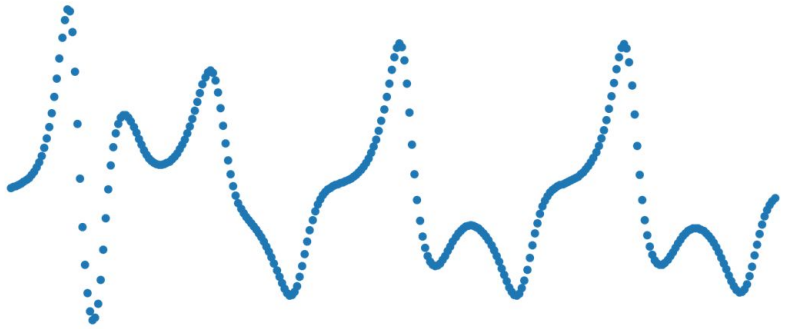
Agreement in geometrical and temporal structure

Simulated trajectory



Dynamical Systems Reconstruction

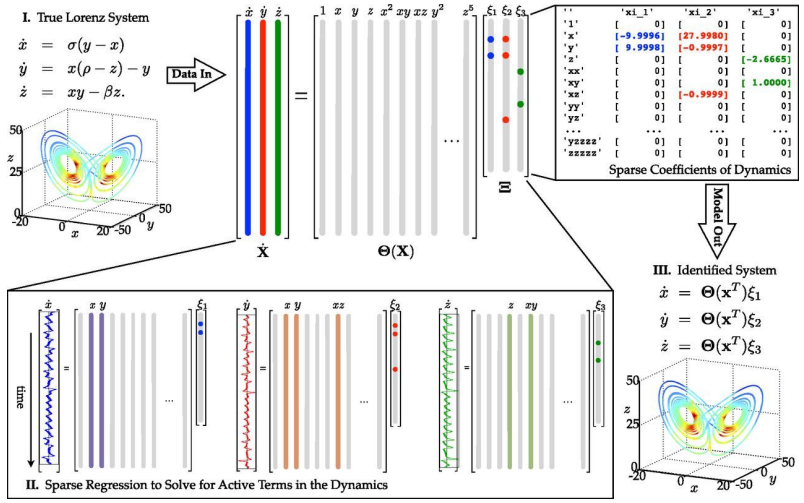
$$\dot{x}_1 = a(x_2 - x_1)$$
$$\dot{x}_2 = x_1(b - x_3) - x_2$$
$$\dot{x}_3 = x_1 x_2 - c x_3$$



Approximate flow field directly	Approximate the flow / solution curve
Symbolic Regression	Reservoir Computing
Neural ODEs	RNN models

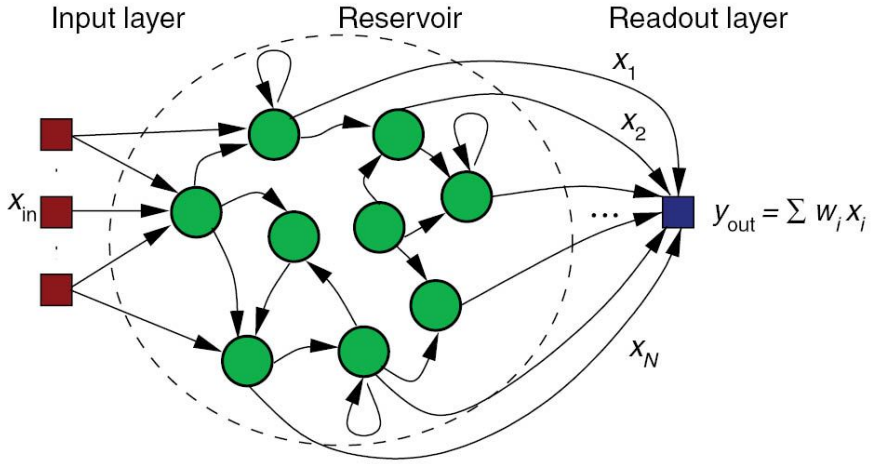
Reconstruction Models

Symbolic Regression

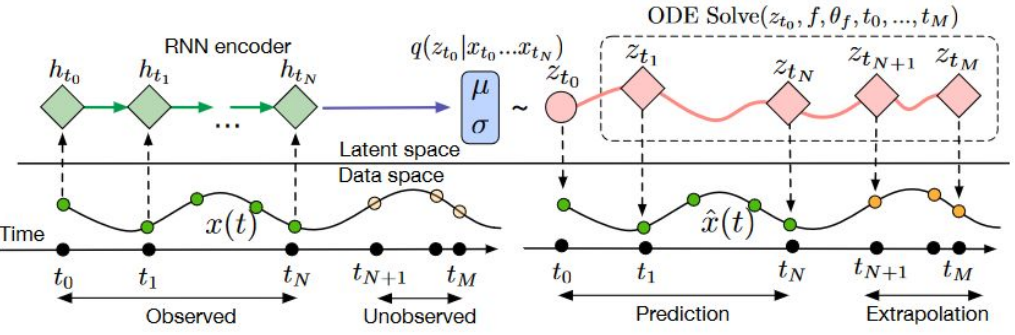


from Brunton et al.

Reservoir Computing

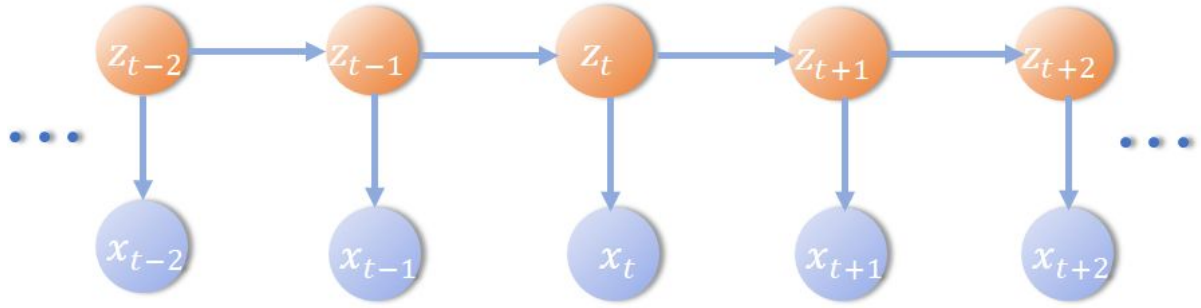


ODE formulations



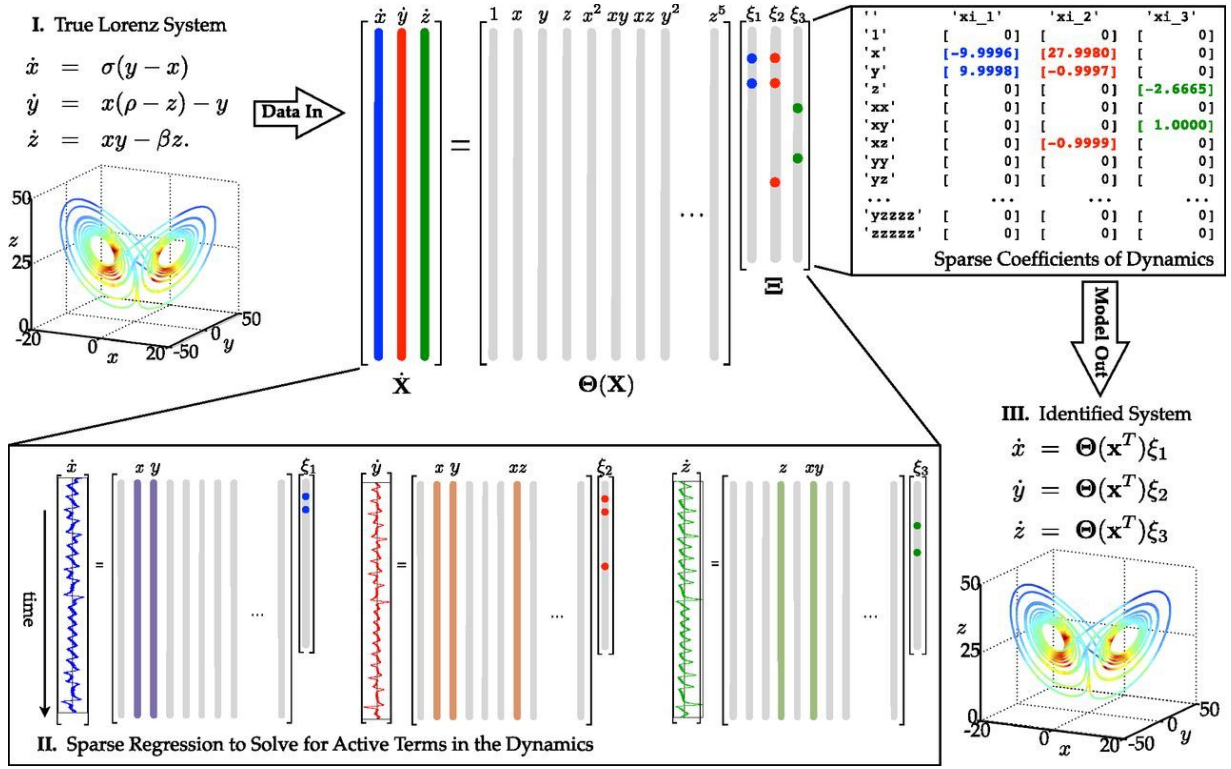
from Chen et al., 2018

RNN models



Symbolic Regression

Approaches like SINDy (Sparse Identification of Nonlinear Dynamics) (Brunton et al.)
Fit parameters from set of **library functions** (polynomials, trigonometric functions…) via **sparse** regression



from Brunton et al., 2017

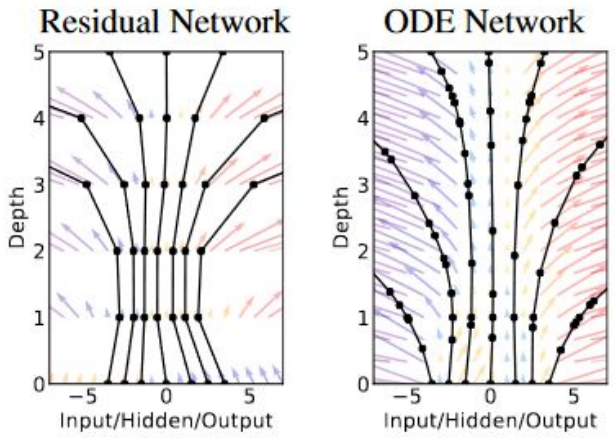
Pros

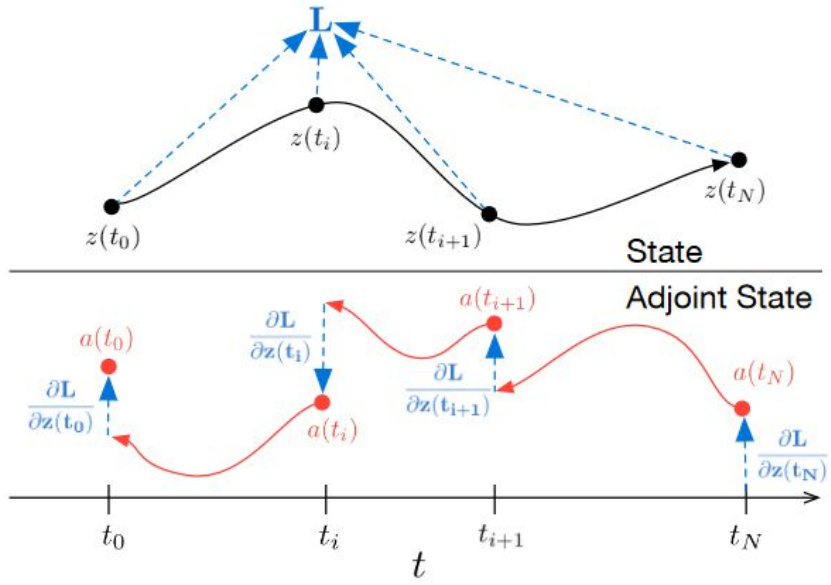
- Highly interpretable solutions
- Fast fitting

Cons

- Dependence on library function
- Numerically unstable for experimental data
- Requires numerical approximation of flow field

Neural ODEs





from Chen et al., 2018

Pros

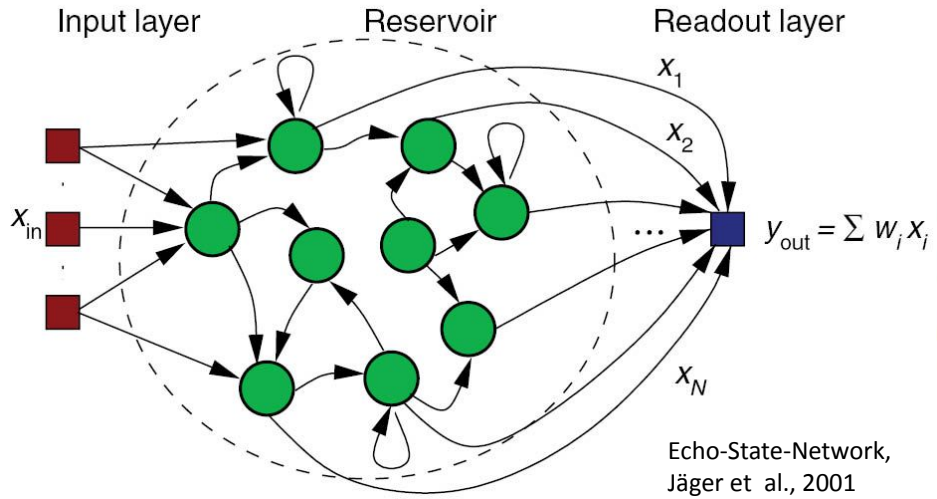
- Universal approximation
- Continuous time solutions (independent of discretization)

Cons

- more frequently numerically unstable
- challenging to train
- solutions frequently depend on solver
- rather slow training via adjoint method

Reservoir Computing

- Initialize a large dynamical reservoir with specific hyperparameters (connectivity, sparsity etc.)
- Input data at every time step
- Only fit output layer via regression



Echo-State-Network,
Jäger et al., 2001

$$\text{input:} \quad \mathbf{u}_t = \mathbf{W}^{in}\mathbf{x}_t$$

$$\text{reservoir:} \quad \mathbf{r}_t = (1-\alpha)\mathbf{r}_{t-1} + \alpha f(\mathbf{A}\mathbf{r}_{t-1} + \mathbf{u}_t + \epsilon_t)$$

$$\text{readout:} \quad \mathbf{y}_t = \mathbf{W}^{out}\mathbf{r}_t + \mathbf{c}$$

Pros

- Super fast training
- No exploding gradients/BPTT necessary

Cons

- Dynamical model hard to analyze
- Dependence on hyperparameters of reservoir initialization
- Hard to get long-term dynamics right

Recurrent Neural Networks

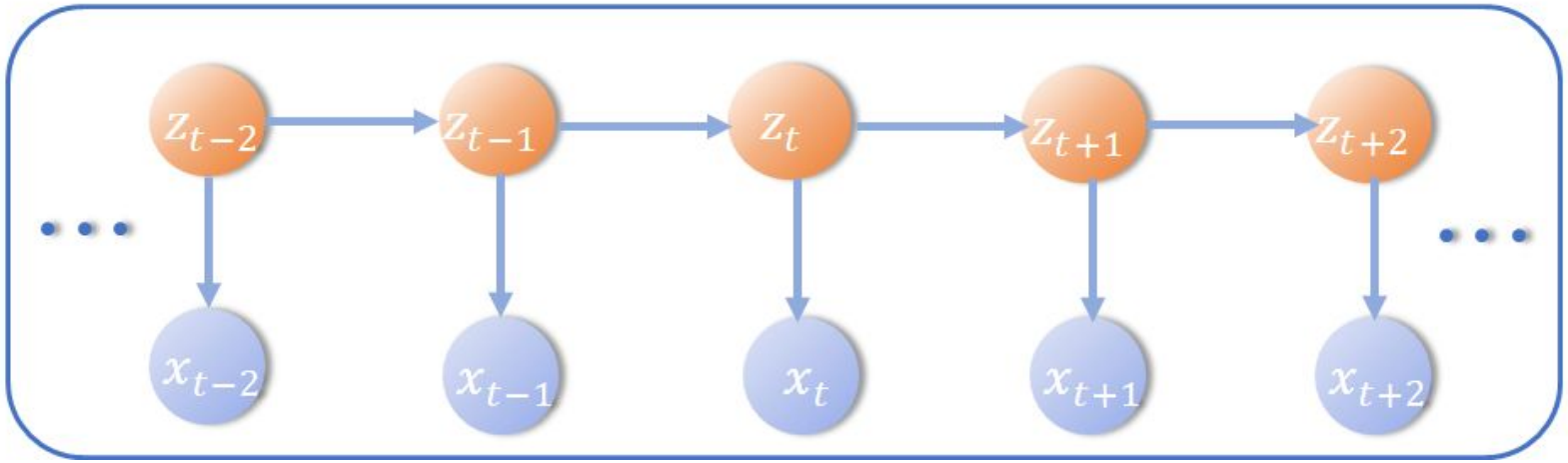
Latent Process

$$z_t = Az_{t-1} + W\phi(z_{t-1}) + Cs_t + h + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \Sigma)$$

Observations

$$x_t = Bz_t + \eta_t, \eta_t \sim \mathcal{N}(0, \Gamma)$$

$$\phi(z_{t-1}) = \max(0, z_{t-1})$$



Pros

- Universal approximation
- mathematically tractable solutions of **dynamics**

Cons

- depends on discretization
- exploding/vanishing gradients during BPTT

Mathematical Tractability & Analysis

Wilson-Cowan model (neural population dynamics)

Analytical computation of fixed points and k-cycles

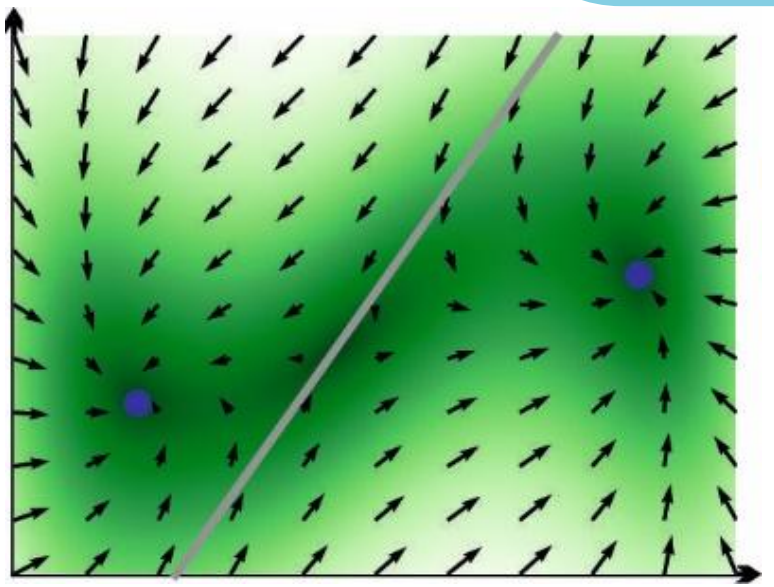
Fixed points

$$z^{*1} = \left(I - W^B_{\Omega(t^{*1})}\right)^{-1}\left[W\, h^B_{\Omega(t^{*1})} + h_0\right]$$

k-cycles

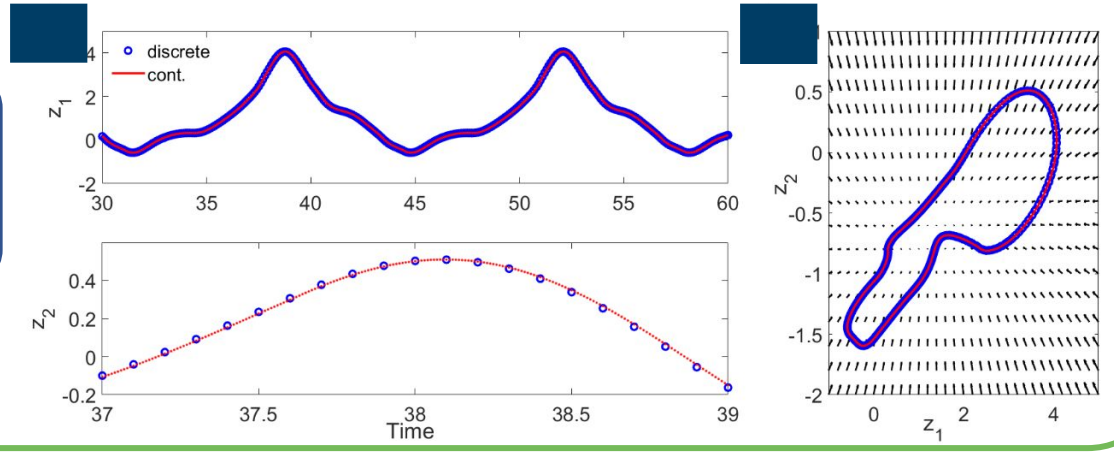
$$z^{*n} = \left(I - \prod_{i=1}^{n} W^B_{\Omega(t^{*n}-i)}\right)^{-1}\left(\sum_{j=2}^{n}\left[\prod_{i=1}^{n-j+1} W^B_{\Omega(t^{*n}-i)} W h^B_{\Omega(t^{*n}-n+j-2)}\right]\right.$$
$$\left. + W h^B_{\Omega(t^{*n}-1)} + \left(\sum_{j=2}^{n}\prod_{i=1}^{n-j+1} W^B_{\Omega(t^{*n}-i)} + I\right) h_0\right),$$

Tractability



Brenner, Hess et al., ICML 2022

Transformation into equivalent continuous time ODE systems (Monfared & Durstewitz, ICML 2020)



shallow PLRNN

A PLRNN reformulation that enables low dimensional reconstructions while retaining mathematical tractability

$$\boldsymbol{z}_t = \boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}) = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{W}_1 ReLU(\boldsymbol{W}_2 \boldsymbol{z}_{t-1} + \mathbf{h}_2) + \mathbf{h}_1$$

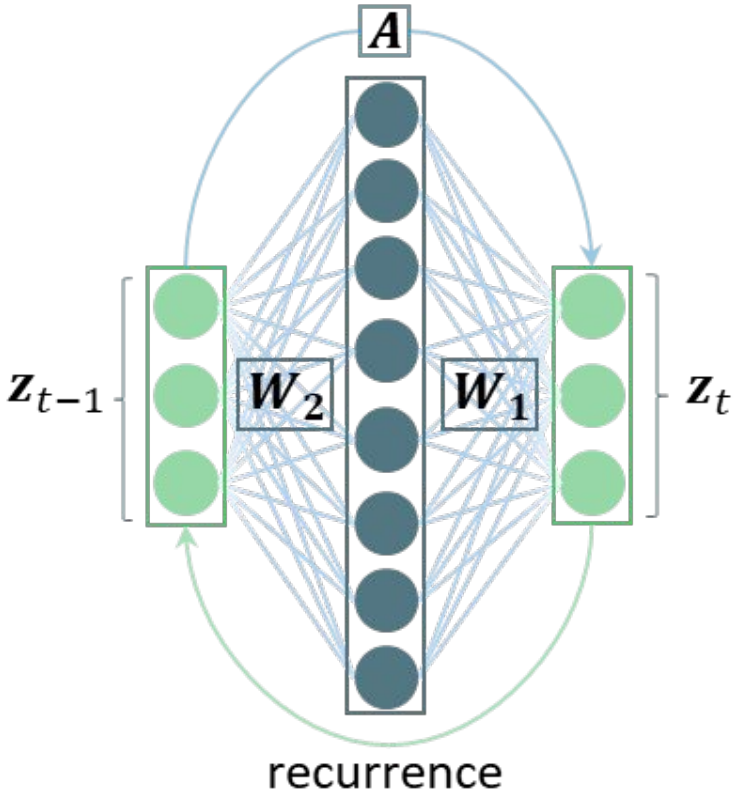
$\boldsymbol{A} \in \mathbb{R}^{M \times M}$ diagonal

$\boldsymbol{W_1} \in \mathbb{R}^{M \times L}, \boldsymbol{W_2} \in \mathbb{R}^{L \times M}$

$\boldsymbol{h_1} \in \mathbb{R}^{M}, \boldsymbol{h_2} \in \mathbb{R}^{L}$

M: model's state space dimensionality

L: hidden layer size



→ will be used in the practical of the tutorial

Hess et al. (2023, ICML)

RNN training for DSR

Florian Hess

Training Techniques and Challenges

Training RNNs via BPTT on chaotic data is ill-posed:

Generic RNN:

$$\boldsymbol{z}_t = \boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{s}_t)$$

Jacobian:

$$\boldsymbol{J}_t := \frac{\partial \boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}, \boldsymbol{s}_t)}{\partial \boldsymbol{z}_{t-1}} = \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_{t-1}}$$

Maximum Lyapunov exponent of an RNN orbit $\boldsymbol{Z} = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_T, \dots\}$:

$$\lambda_{max} := \lim_{T\to\infty} \frac{1}{T} \log \left\| \prod_{r=0}^{T-2} \boldsymbol{J}_{T-r} \right\|$$

$\lambda_{max} > 0$ necessary condition for chaos!

Backpropagation through time (BPTT) with loss $L = \sum_{t=1}^{T} L_t$

$$\frac{\partial L_t}{\partial \theta_i} = \sum_{r=1}^{t} \frac{\partial L_t}{\partial \boldsymbol{z}_t} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_r} \frac{\partial^+ \boldsymbol{z}_r}{\partial \theta_i}$$

with

$$\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_r} = \prod_{k=0}^{t-r-1} \boldsymbol{J}_{t-k}$$

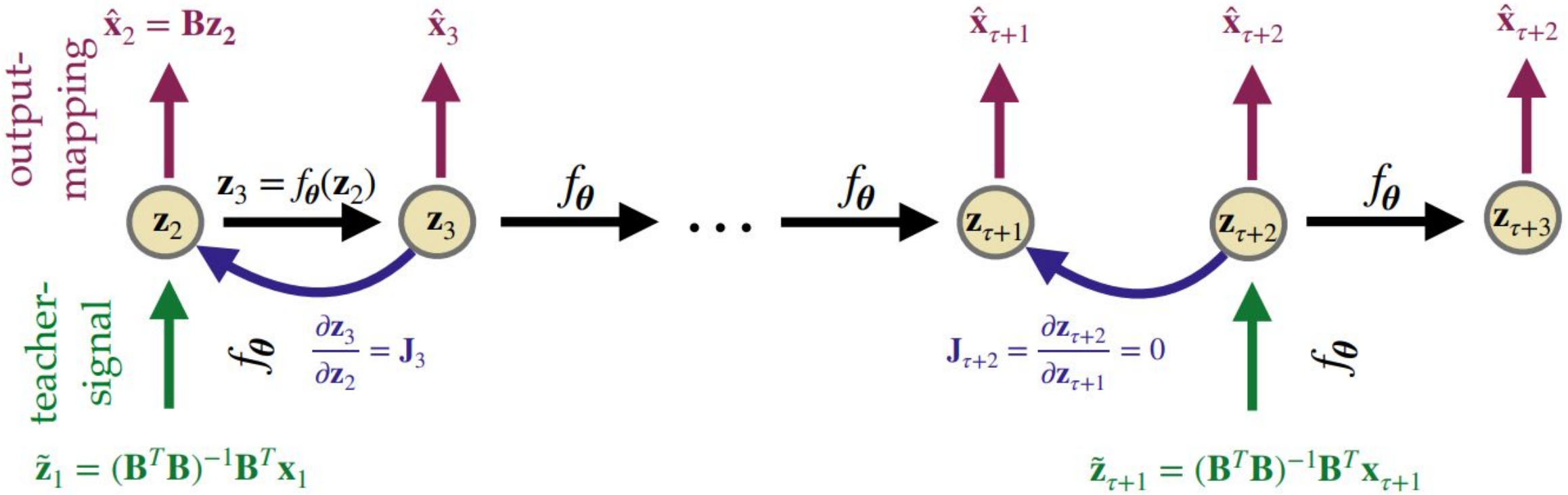
→ Loss gradients during training on chaotic data will **inevitably explode** for $T \to \infty$.

finding of Mikhaeil et al. (2022, NIPS)

Sparse Teacher Forcing (STF)

Proposed solution: Force system in intervals determined by the max. Lyapunov exponent, thereby truncating gradients before they explode.

$$\tau = \frac{\ln 2}{\lambda_{max}}$$



Mikhaeil et al. (2022, NIPS)

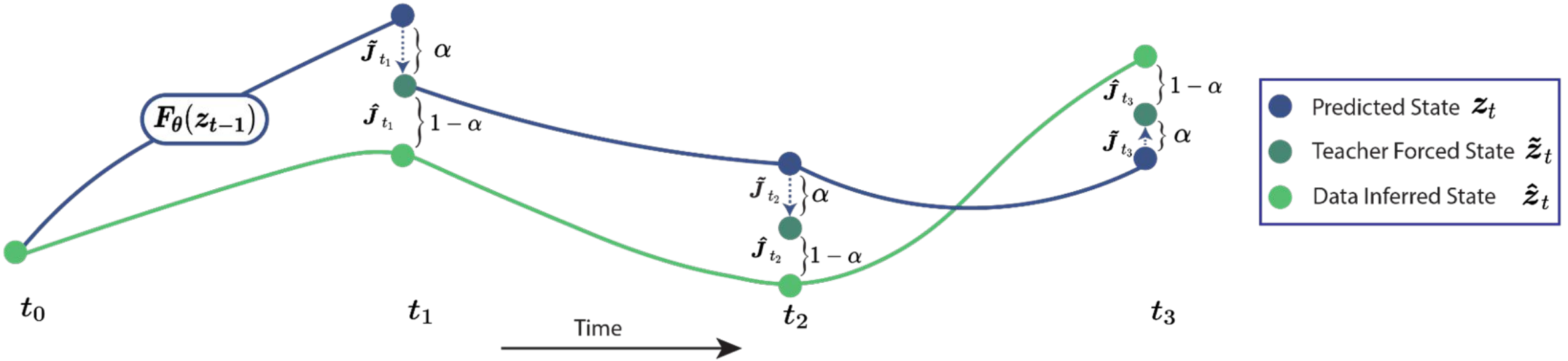
Generalized Teacher Forcing (GTF)

GTF linearly interpolates between RNN state $\boldsymbol{z}_t$ and data-inferred state $\hat{\boldsymbol{z}}_t$ with parameter $0 \leq \alpha \leq 1$

$$\boldsymbol{z}_t = \boldsymbol{F}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_{t-1})$$
$$\tilde{\boldsymbol{z}}_{t-1} = (1-\alpha)\boldsymbol{z}_{t-1} + \alpha\hat{\boldsymbol{z}}_{t-1}$$

Jacobian factorizes

$$\boldsymbol{J}_t = \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_{t-1}} = \frac{\partial \boldsymbol{F}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_{t-1})}{\partial \tilde{\boldsymbol{z}}_{t-1}} \frac{\partial \tilde{\boldsymbol{z}}_{t-1}}{\partial \boldsymbol{z}_{t-1}} = \tilde{\boldsymbol{J}}_t(1-\alpha)$$



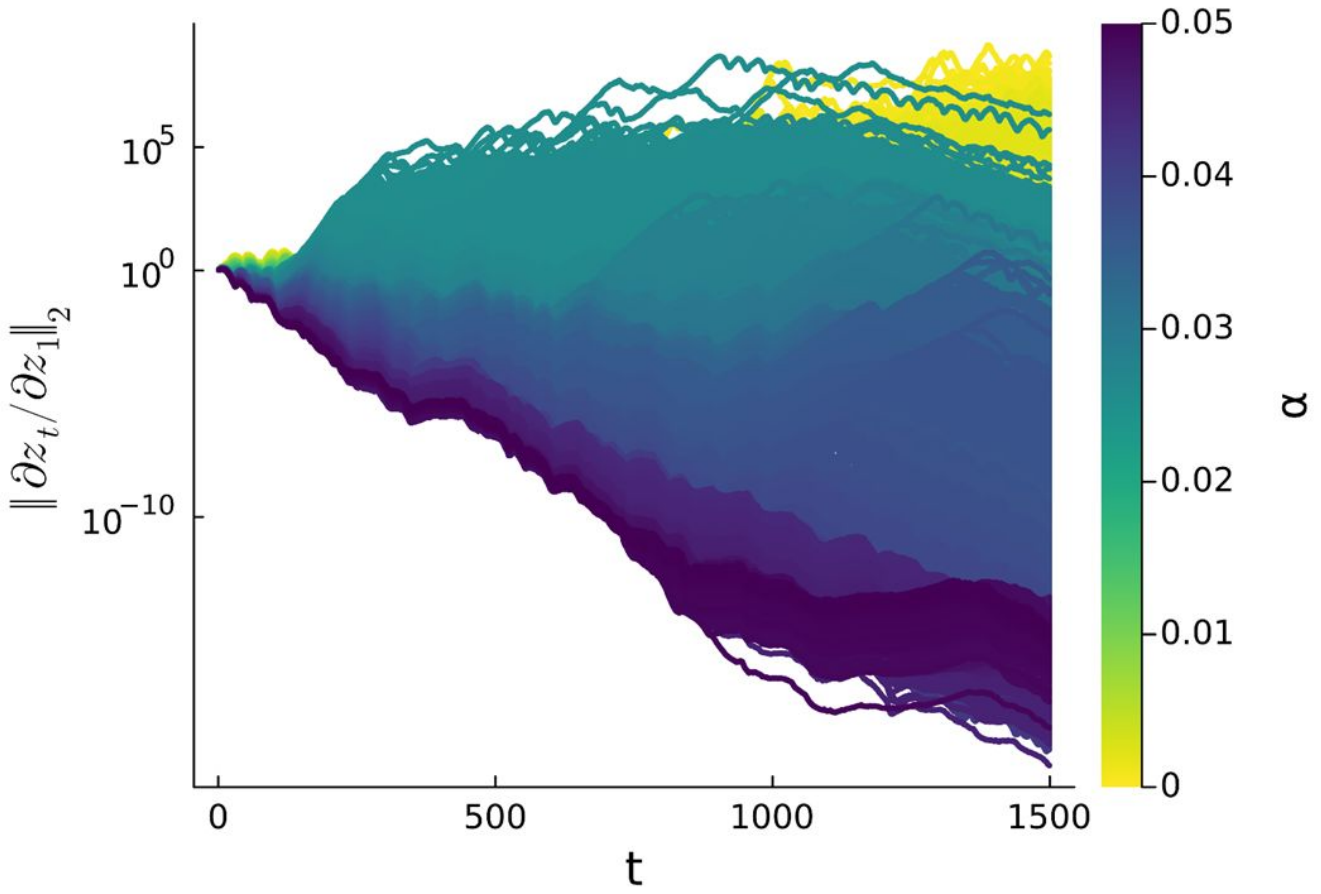
Hess et al. (2023, ICML)

Generalized Teacher Forcing (GTF)

With the right choice of α, GTF leads to all-time bounded gradients during BPTT training

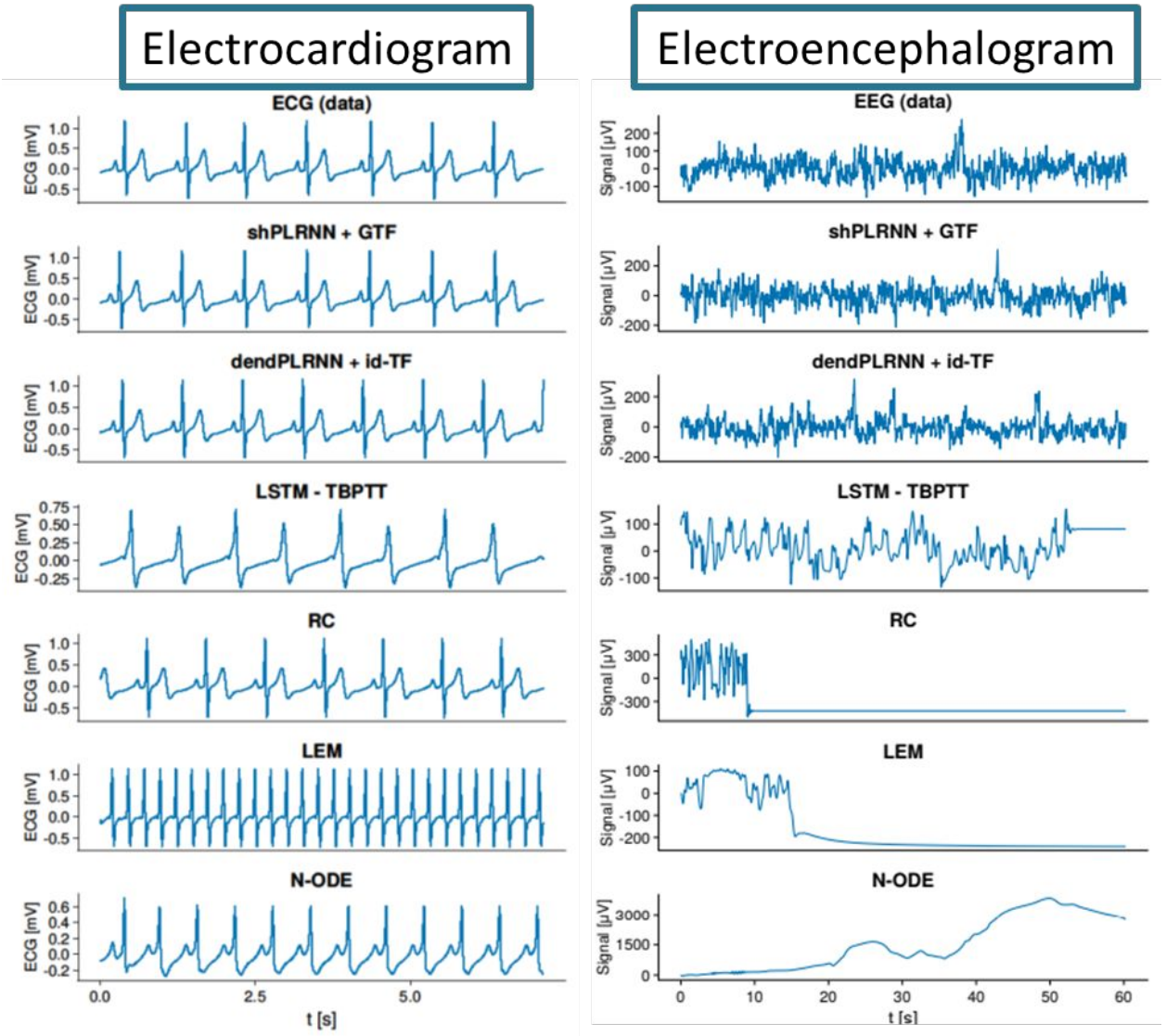
$$\left\| \frac{\partial \mathbf{z}_t}{\partial \mathbf{z}_r} \right\| = (1-\alpha)^{t-r} \left\| \prod_{k=0}^{t-r-1} \tilde{\boldsymbol{J}}_{t-r} \right\|$$

Combined with shallow PLRNN
→ SOTA reconstruction performance!



Hess et al. (2023, ICML)

Generalized Teacher Forcing (GTF)



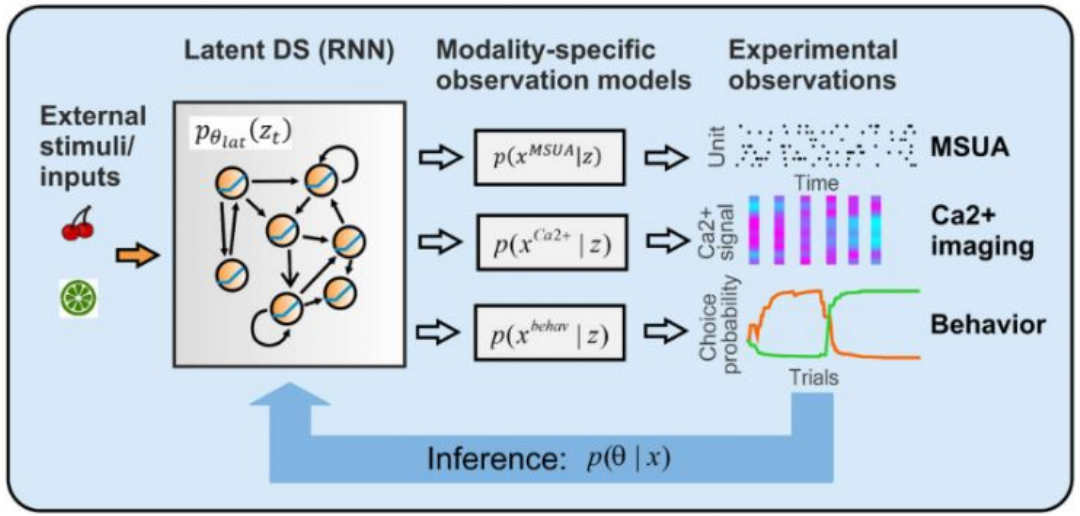
Competitive performance of shPLRNN + GTF compared to 4 major classes of DSR algorithms

- Gated RNN architectures (LSTMs)
- Reservoir Computers (RC)
- Library-based methods (SINDy)
- ODE-based RNNs like Long Expressive Memory (LEM) and Neural ODEs (N-ODE)

Especially on empirical data!

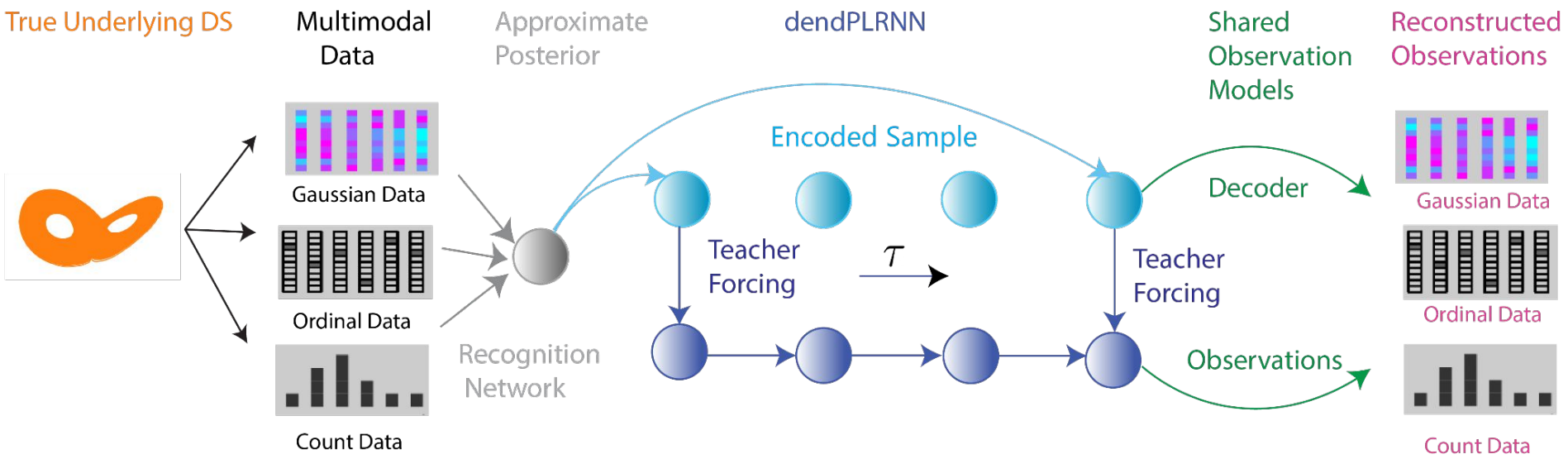
Hess et al. (2023, ICML)

Multimodal Extensions



Kramer, Bommer et al., ICML 2022

Our solution: Multimodal Variational Autoencoder+Teacher Forcing



Brenner et al., AAAI 2023

Validation

Geometrical agreement

Kullback Leibler divergence in state space

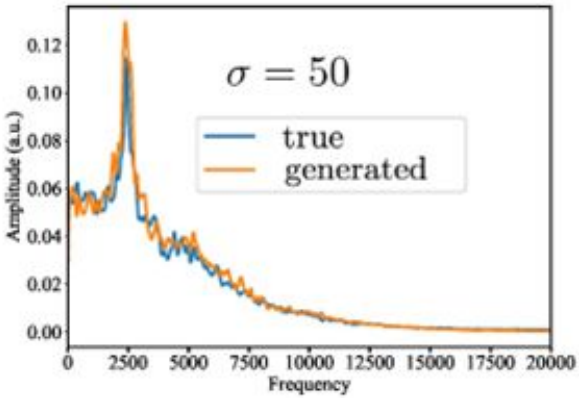
Low KL

Medium KL

High KL

Agreement in temporal structure

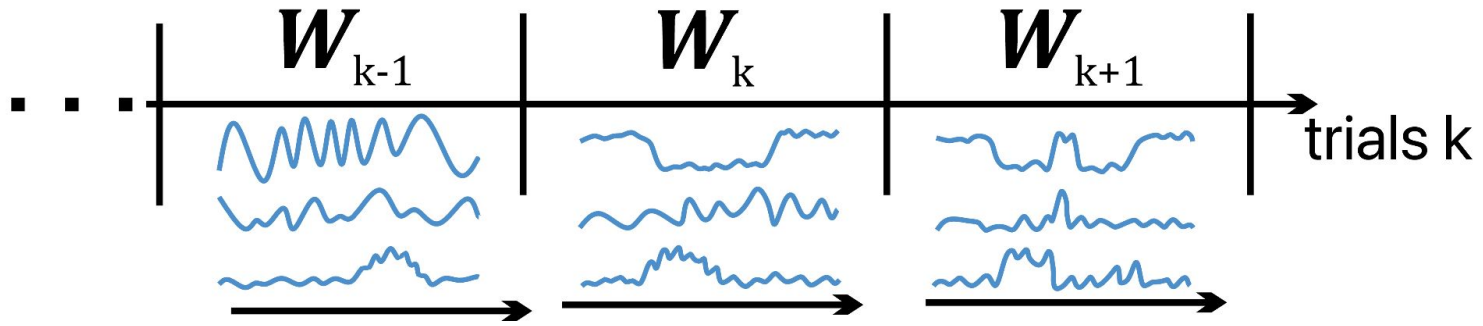
Power spectrum Hellinger Distance

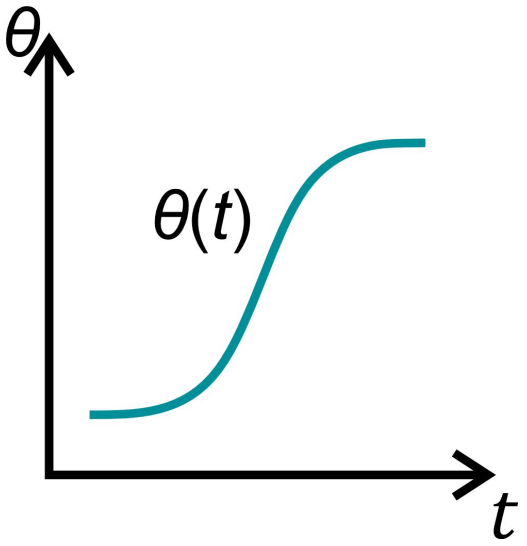


$$D_{\text{stsp}}\left(p_{\text{true}}(\boldsymbol{x}), p_{\text{gen}}(\boldsymbol{x} \mid \boldsymbol{z})\right) \approx \sum_{k=1}^{K} \hat{p}_{\text{true}}^{(k)}(\boldsymbol{x}) \log\left(\frac{\hat{p}_{\text{true}}^{(k)}(\boldsymbol{x})}{\hat{p}_{\text{gen}}^{(k)}(\boldsymbol{x} \mid \boldsymbol{z})}\right)$$

$$H(F(\omega), G(\omega)) = \sqrt{1 - \int_{-\infty}^{\infty} \sqrt{F(\omega)G(\omega)}\, d\omega} \in [0, 1]$$

Nonstationary RNNs





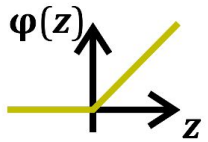
PLRNN: Trial-dependent connectivity matrix

$$z_t = Az_{t-1} + W_k\varphi(z_{t-1}) + h + Cs_t + \varepsilon_t,\ \varepsilon_t \sim N(0,\Sigma)$$ Latent model

External Inputs

$$x_t = B\varphi(z_{t-1}) + \eta_t,\ \eta_t \sim N(0,\Gamma)$$ Observation model

$$\varphi(z) = max(0, z)$$ *ReLU*



Regularization:

$$Loss(Z, \Theta) = -ELBO(Z, \Theta) + \frac{\lambda_0}{2n}\sum_{k=1}^{N}\|W_k\|_2^2 + \frac{\lambda_1}{2n}\sum_{k=1}^{N-1}\|W_{k+1} - W_k\|_2^2 + \frac{\lambda_2}{2n}\sum_{k=1}^{N-2}\|W_{k+2} - 2W_{k+1} + W_k\|_2^2$$

Weight regularization

Continuity prior (1^{st}-order)

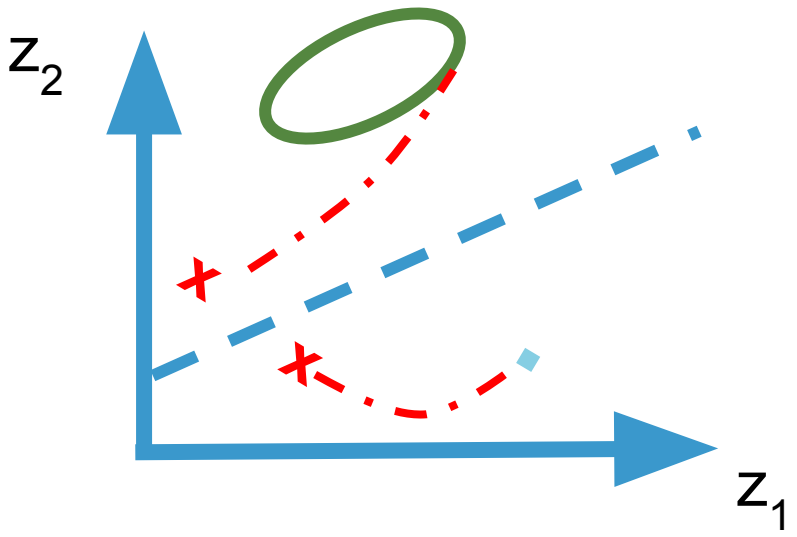
Smoothness prior (2^{nd}-order)

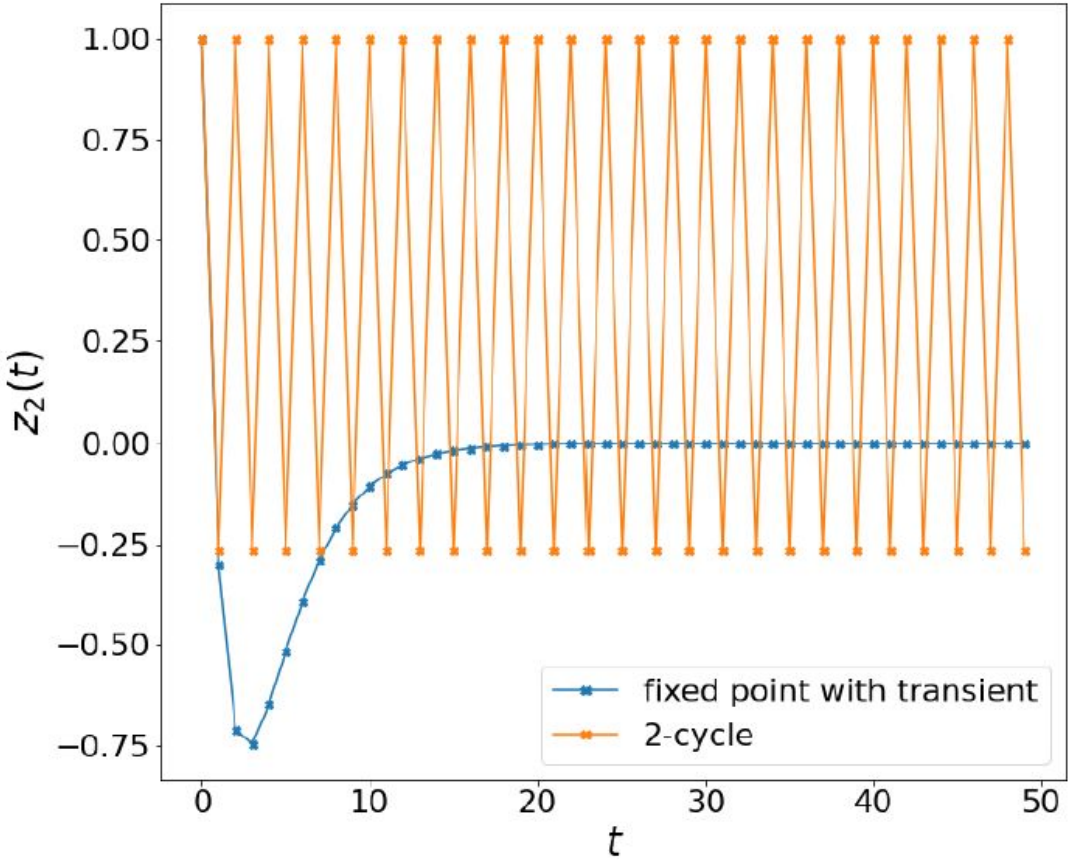
Analysis of Trained Models

Lukas Eisenmann

Analysis of trained models

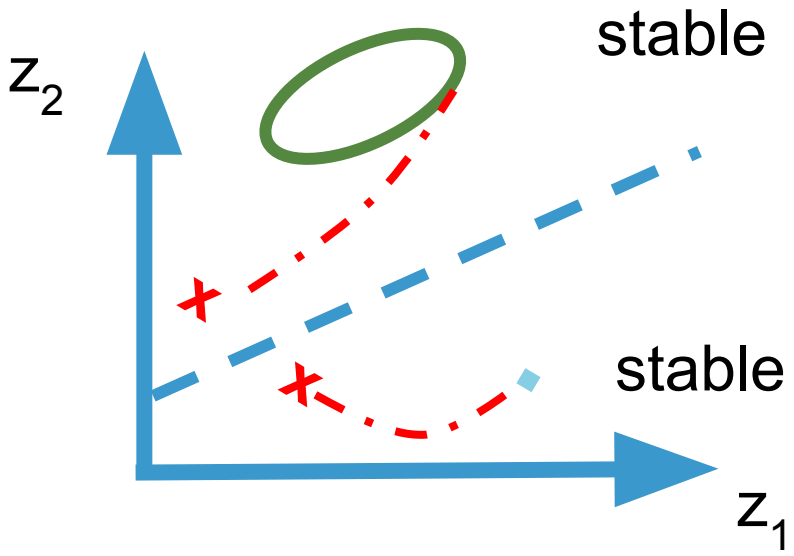
- Trained models can produce different types of time series
- One model can exhibit different behaviour depending on the initialization (multistability)

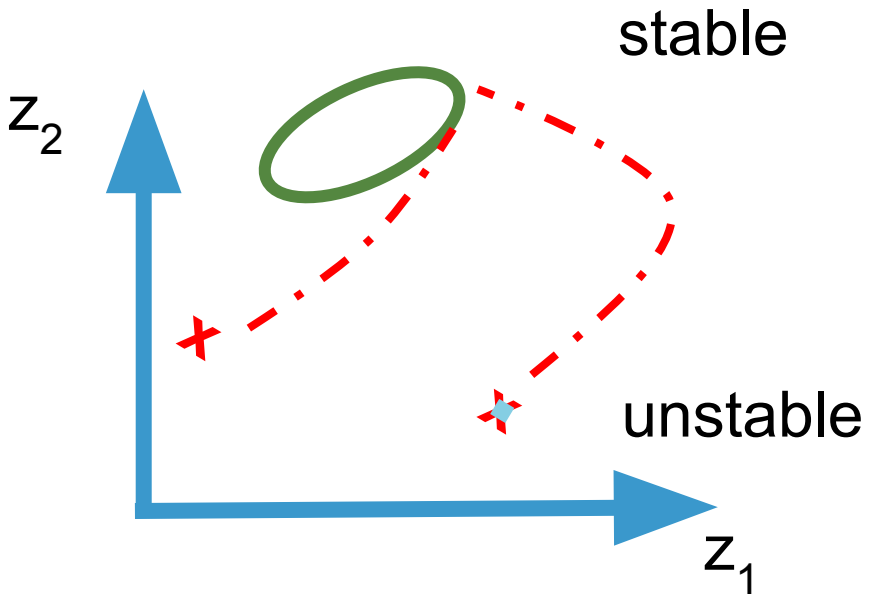




Analysis of trained models

- One PLRNN can have several dynamical objects like fixed points and cycles
- They can be attractors (stable), repellers (unstable) or with saddle properties (stable +unstable dimensions)

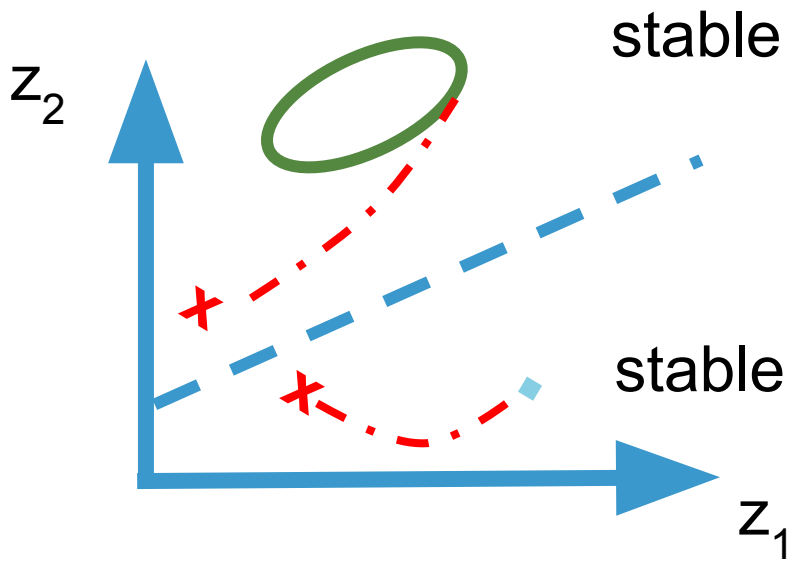


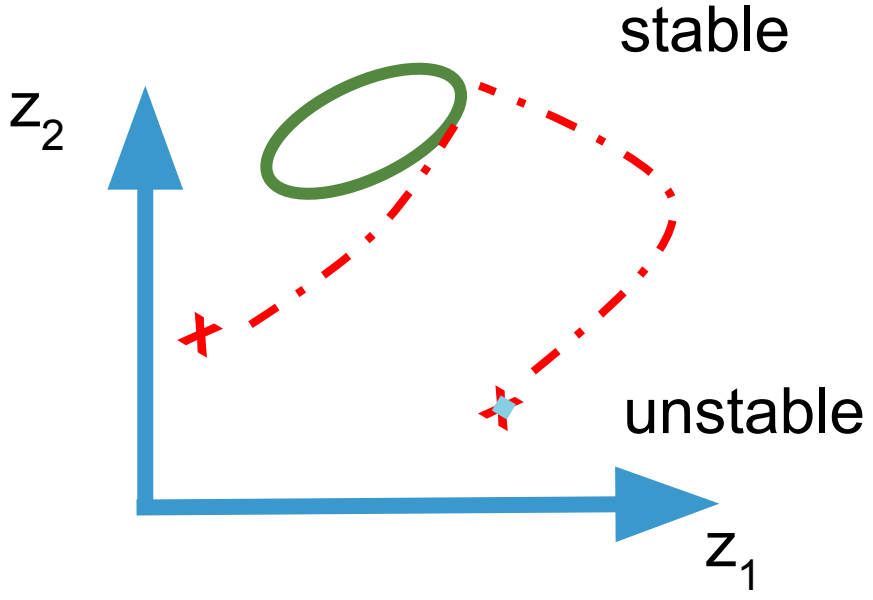


- We want to find all dynamical objects, understand their properties and visualize them

Analysis of trained models

- Properties we are interested in:
  - Stability
  - Cycle order
  - Lyapunov exponents (chaotic attractors)



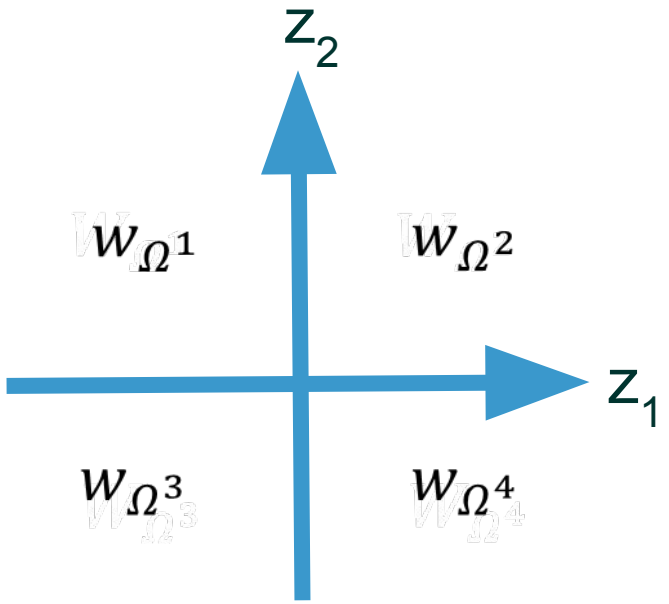


Analysis of trained models: Compute dynamical objects

- Analytically tractable, equation for FP/k-cycle:

$$z_k^* = \left(\mathbb{1} - \prod_{r=0}^{k-1} W_{\Omega(k-r)}\right)^{-1} \left[\sum_{j=2}^{k-1} \prod_{r=0}^{k-j} W_{\Omega(k-r)} + \mathbb{1}\right] h,$$

- Stability is given by the eigenvalues of the Jacobians
- Problem: solve eq. in every linear region

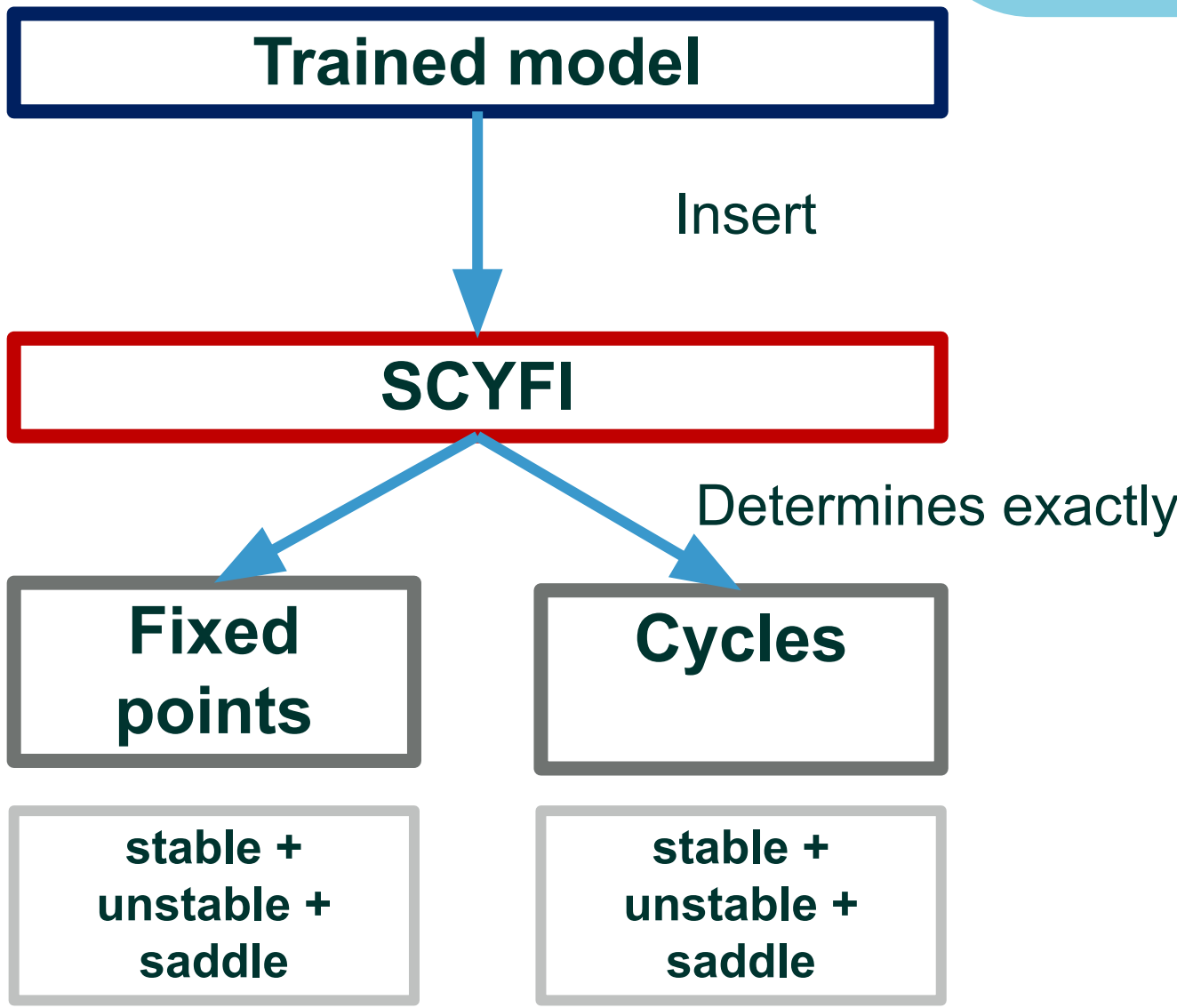


→ Number of combinations: 2^{dk}

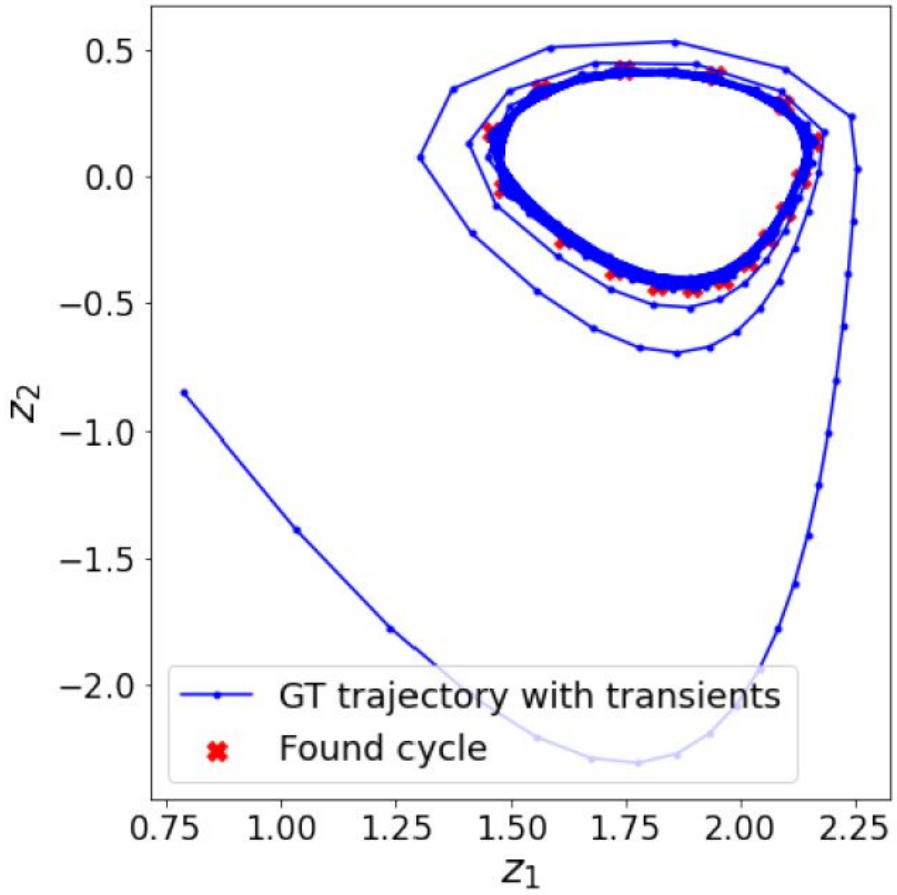
Analysis of trained models: Compute dynamical objects

- We developed an algorithm “**S**earcher for **Cy**cles and **Fi**xed points” (**SCYFI)**
- SCYFI determines the fixed points and cycles of a PLRNN *exactly*
- Deals with the combinatorial issue

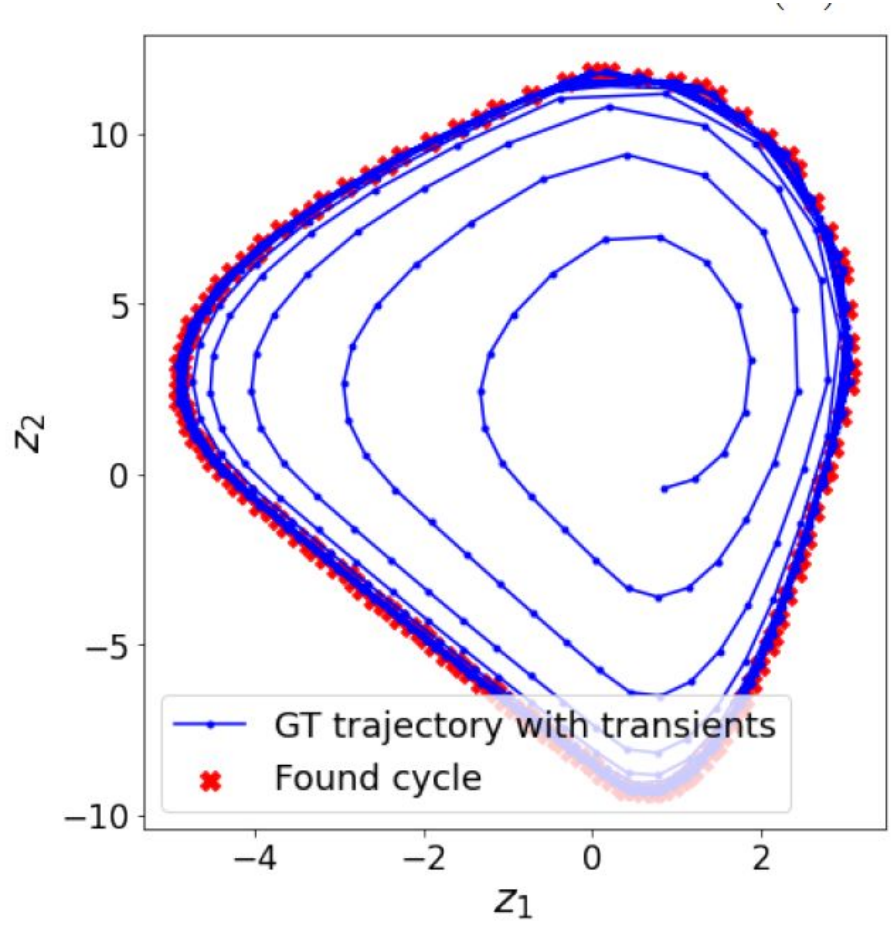
$$z_k^* = \left(\mathbb{1} - \prod_{r=0}^{k-1} W_{\Omega(k-r)} \right)^{-1} \left[\sum_{j=2}^{k-1} \prod_{r=0}^{k-j} W_{\Omega(k-r)} + \mathbb{1} \right] h,$$



Analysis of trained models: synthetic examples



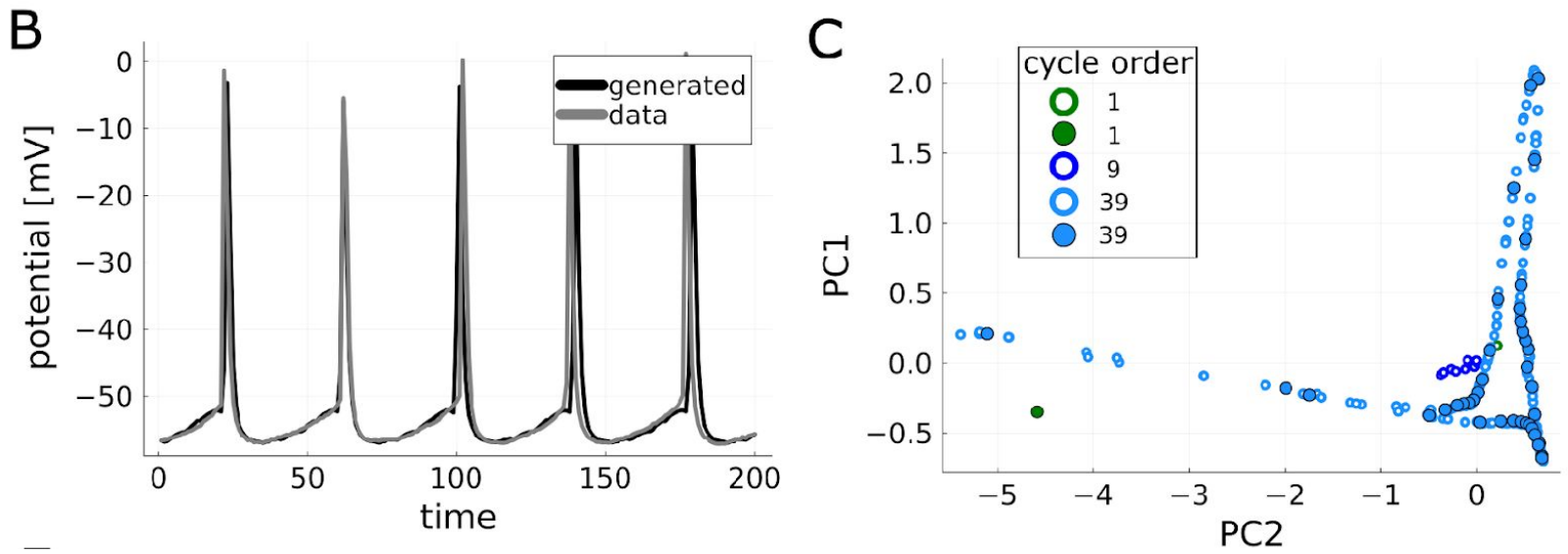
16-cycle



133-cycle

Analysis of trained models: real example

- Shallow PLRNN (M=6,H=20) trained on electrophysiological recordings from a cortical neuron.

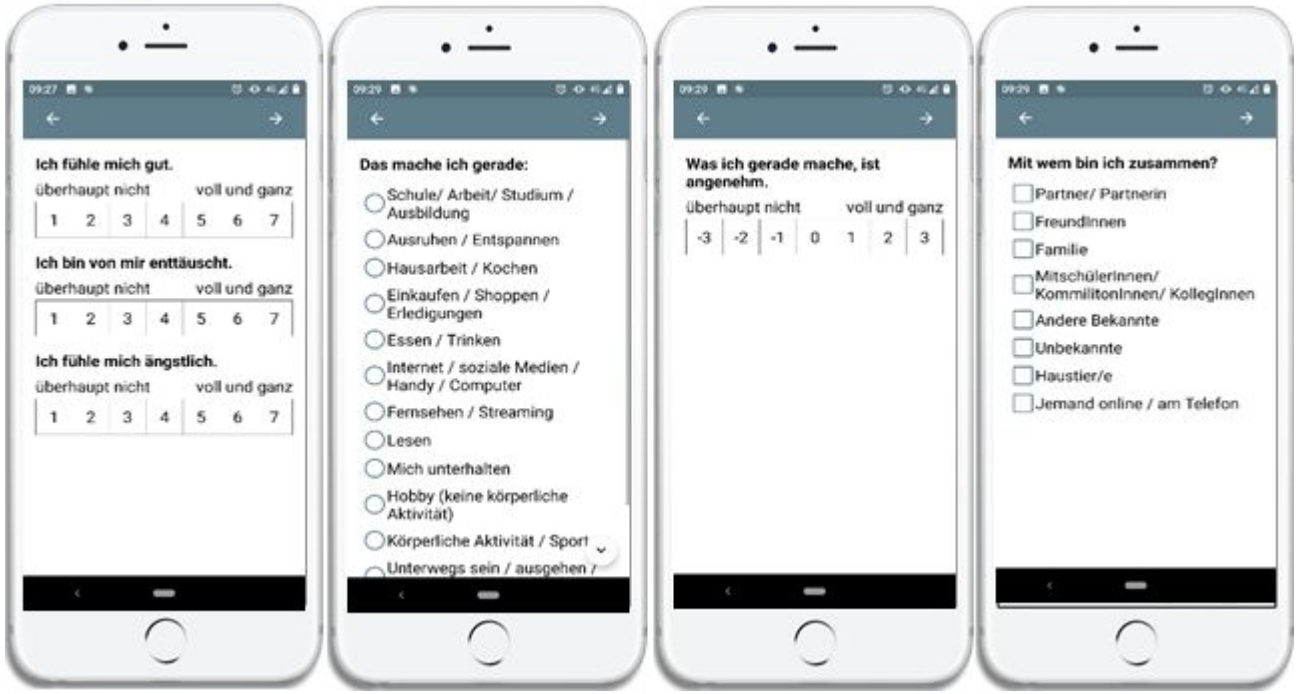


Practical considerations for training on imperfect data

Janik Fechtelpeter

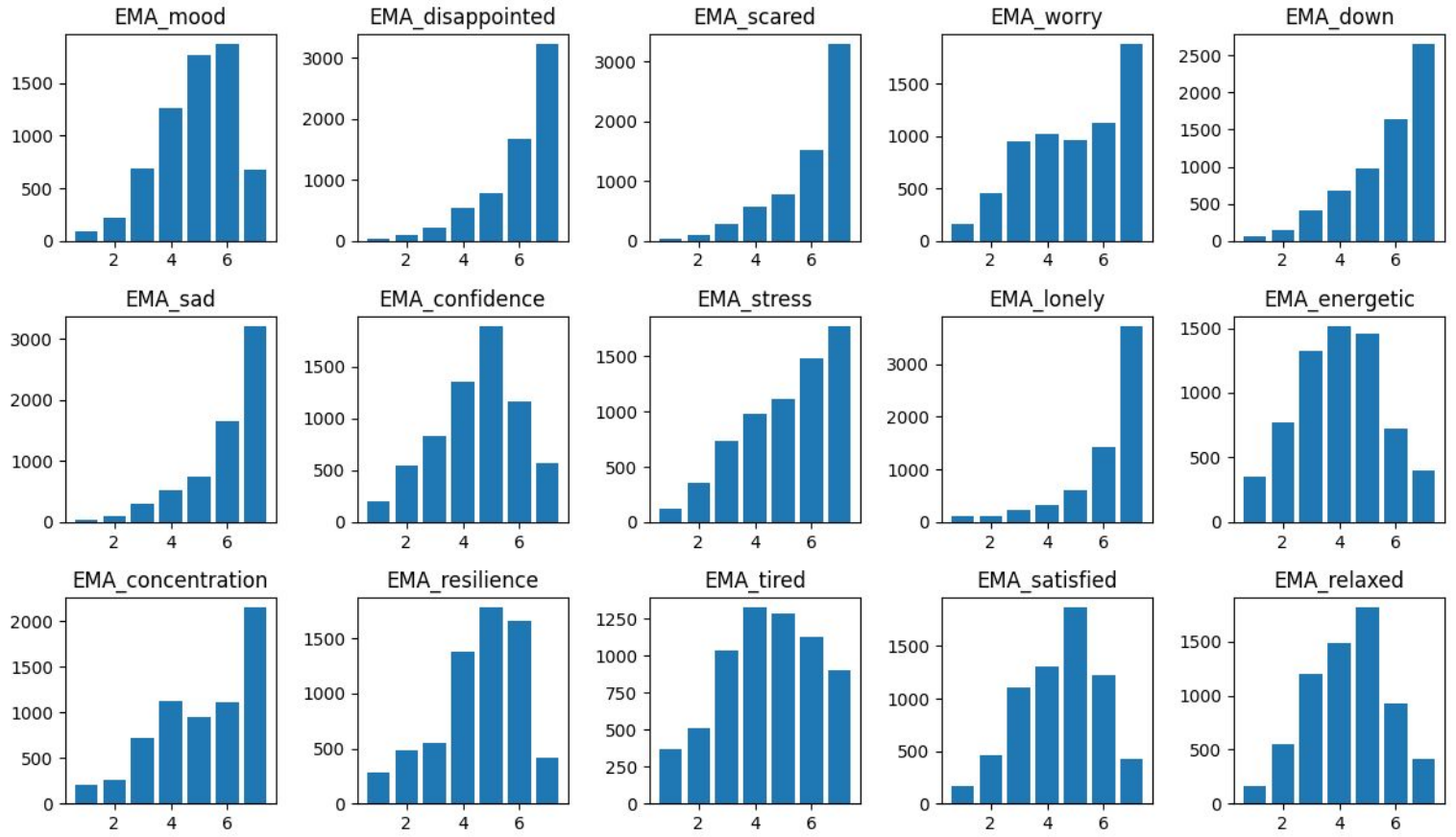
Training on Imperfect Data

- ordinal data
- missing values
- irregular sampling
- e.g. from event sampling methods



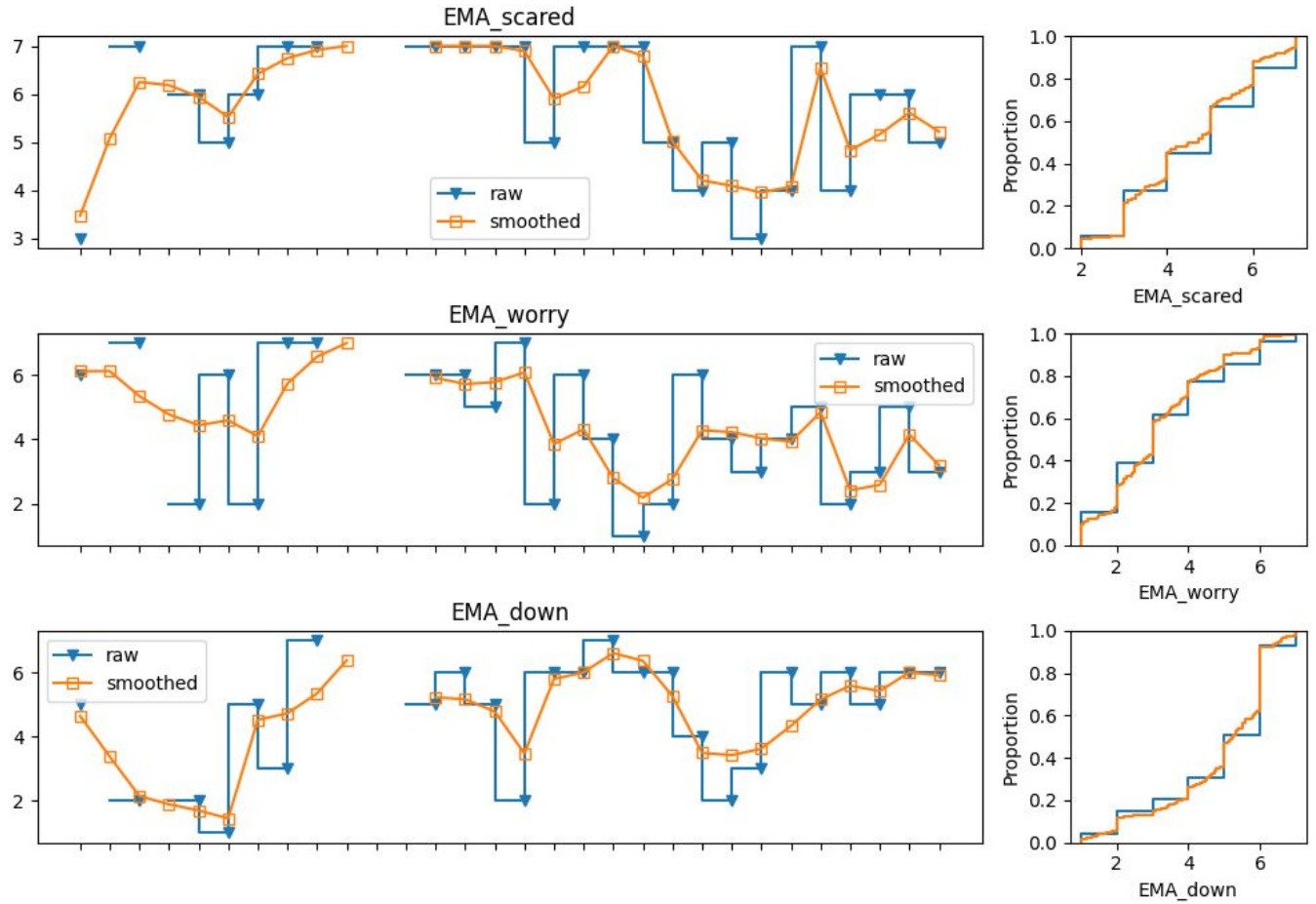
Training on Ordinal Data

- (Pre-)ordered data, no distance metric
- Distribution assumption violated



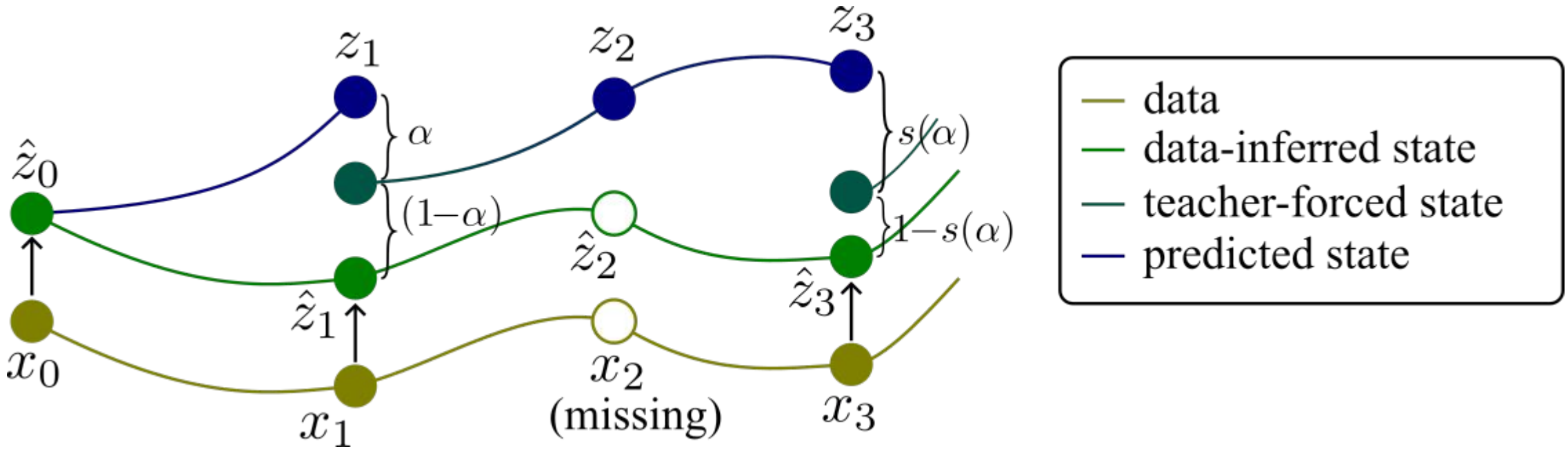
Training on Ordinal Data

- Gaussian kernel smoothing



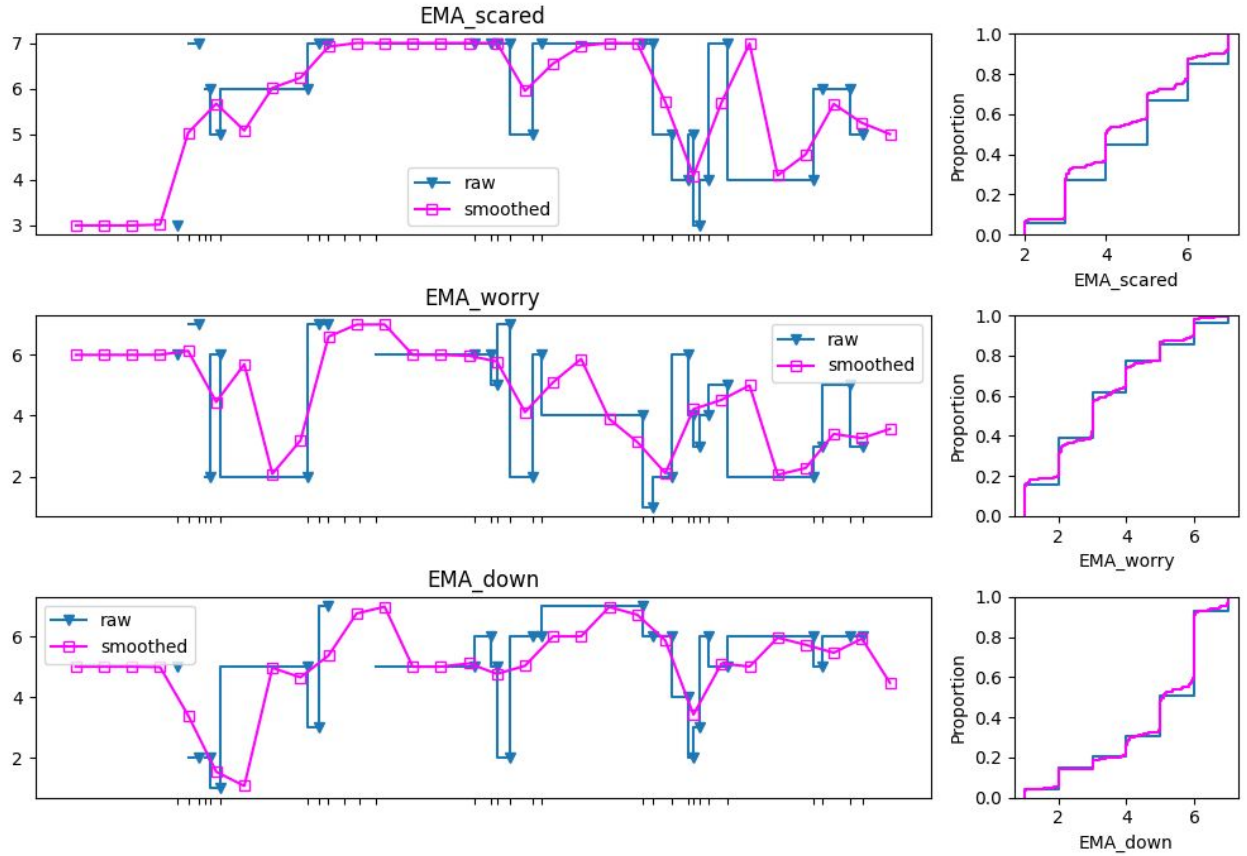
Missing data

- Interpolate missing values, but don't train on them
- Omit interpolated values in teacher forcing
- Possibly adapt α after missing values



Irregularly sampled data

- Interpolate gaps
- Evaluate smoothing function at regular time points

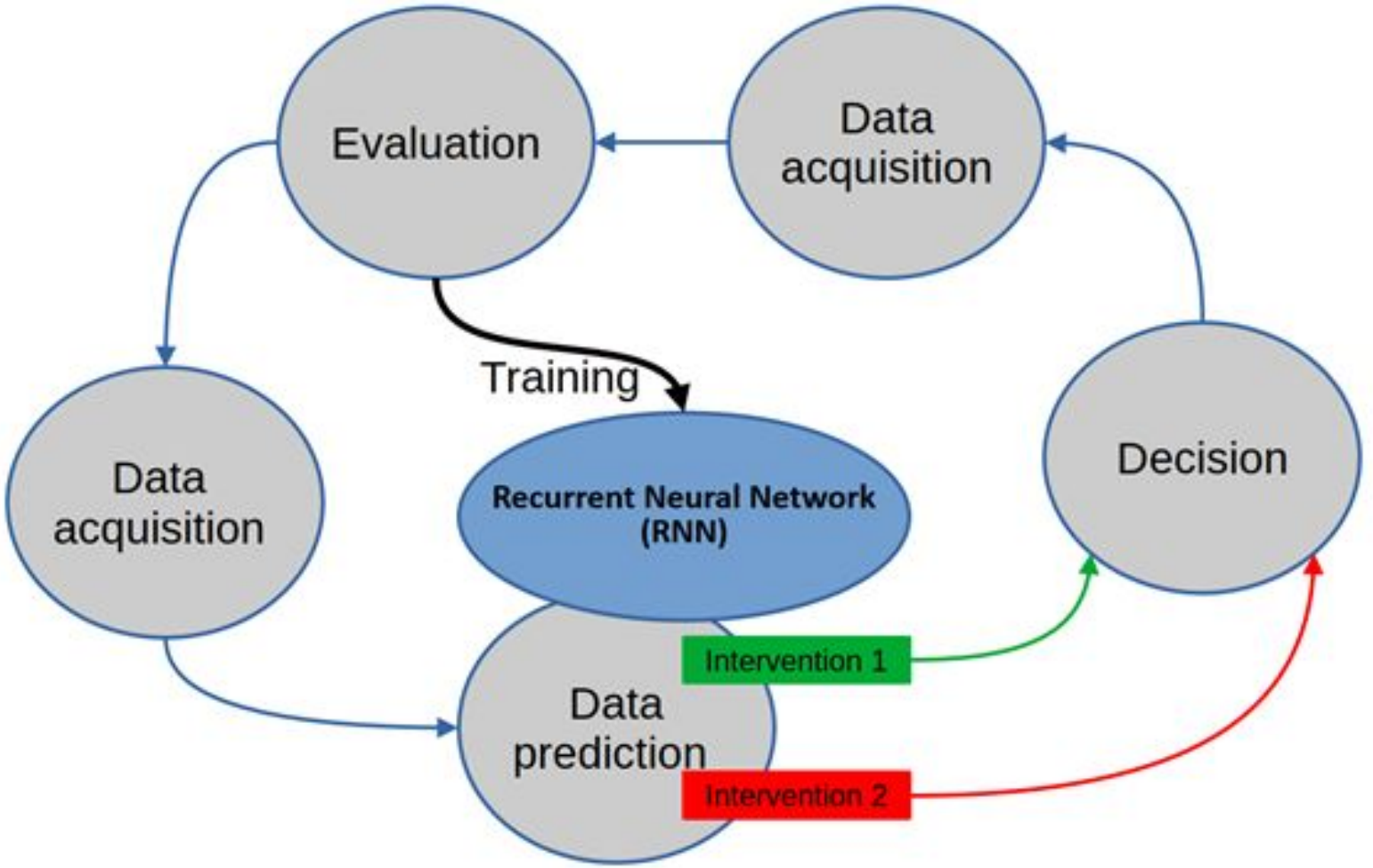


Control theoretic application of trained PLRNNs

Using trained PLRNNs to optimize control input

- In a control theoretic framework, the PLRNN can be used to optimize control input to a system, e.g. by optimizing the impulse response

$$\mathrm{IR}(s,t) = F_\theta^t(Cs)$$



Observation model for fMRI data

Alena Brändle

BOLD (Blood-oxygen-level dependent) observation model

$$x_t = B \cdot (\mathrm{hrf} * z)_t + J \cdot r_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \Gamma)$$

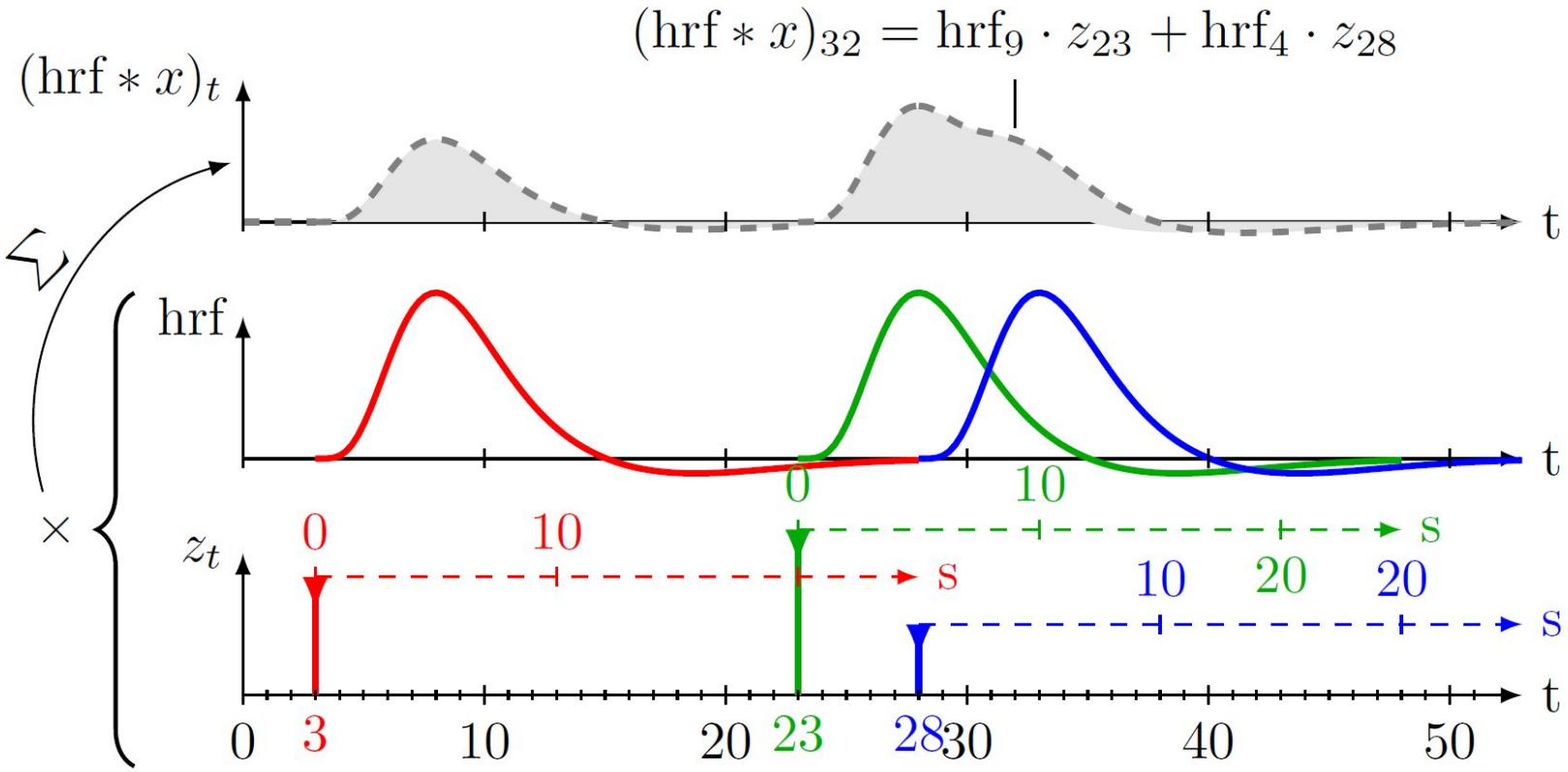
- Need to invert observation model to get data inferred latent states (as initial state, for teacher forcing)

- Convolution with the hemodynamic response function **hrf** leads to a dependence of measured state **x**□ on multiple latent states {**z**□}

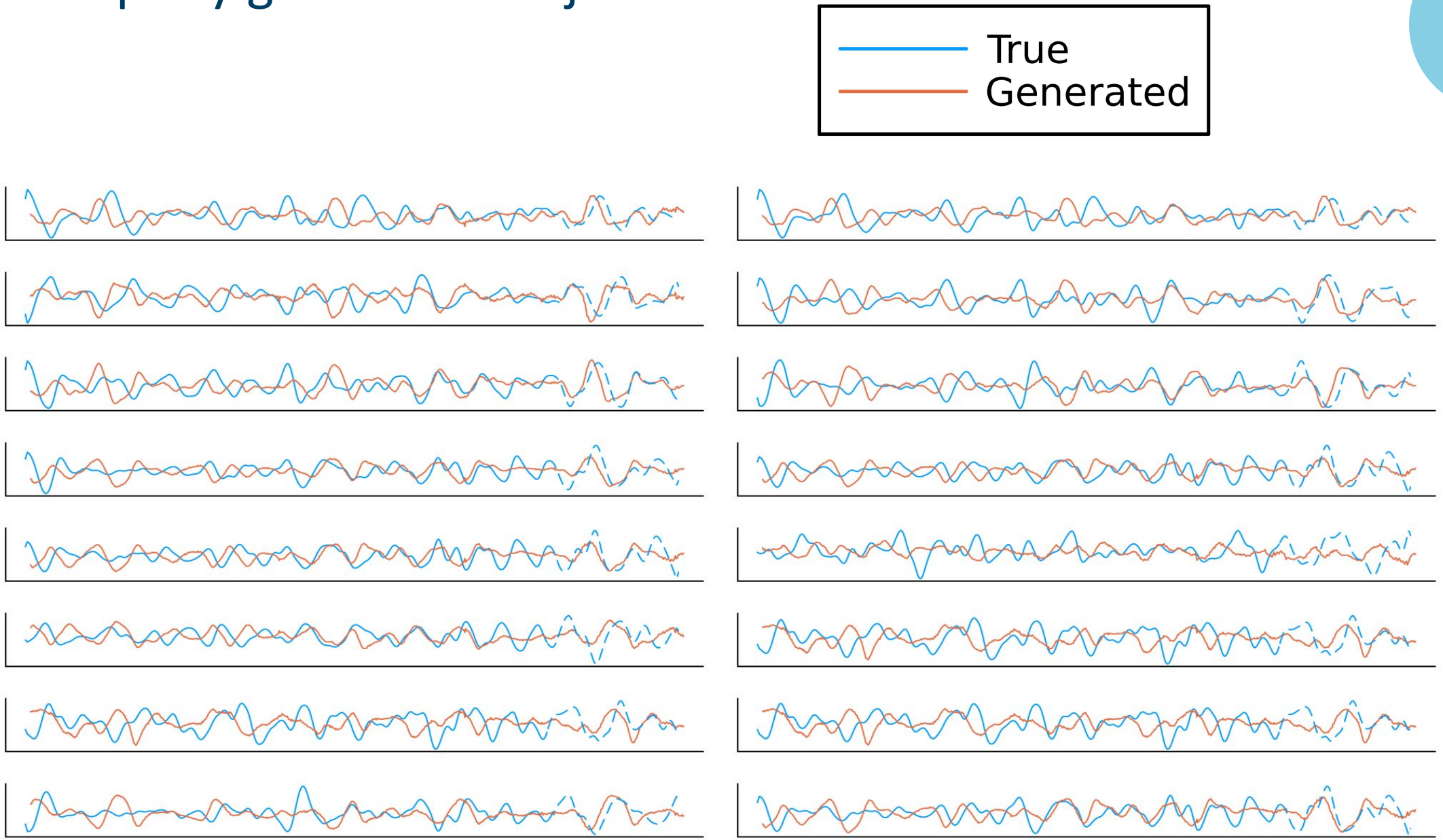
  ⇒ Need adjusted code to deal with this problem (not published yet)

Convolution with the hemodynamic response function

$$(\mathrm{hrf} * z)_t = \sum_{s=0}^{\tau} \mathrm{hrf}_s \cdot z_{t-s}$$



Exemplary generated trajectories

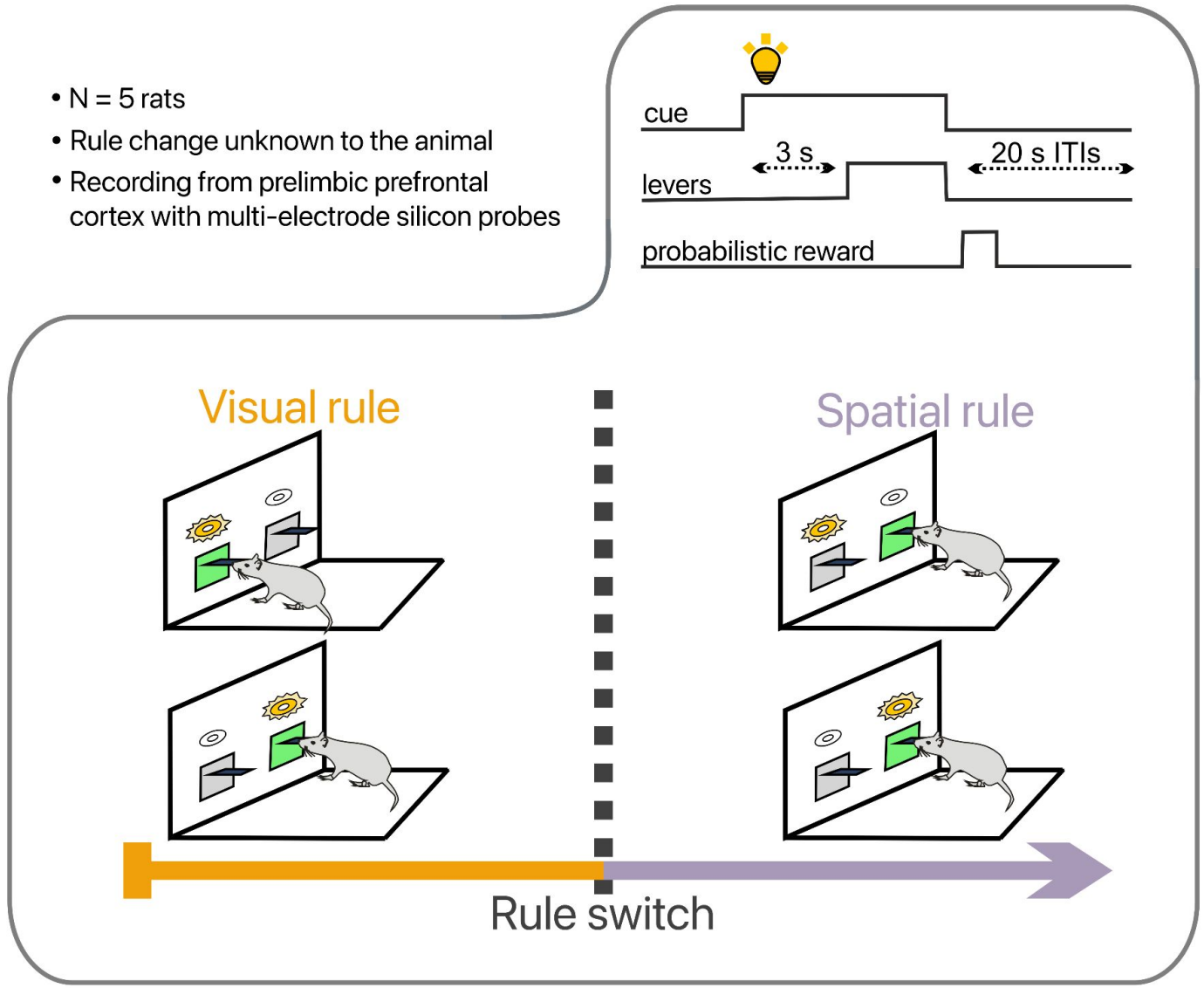


Non-stationarity during rule learning

Max Ingo Thurm

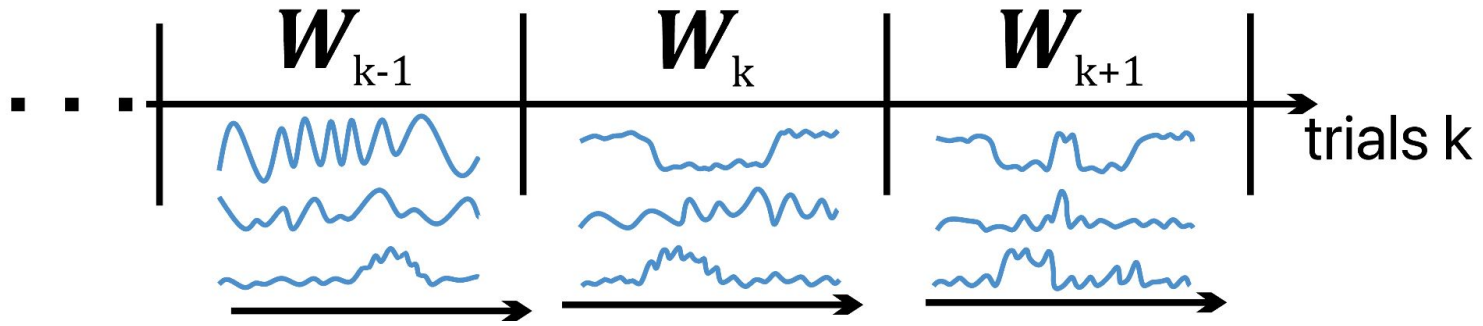
The experiment

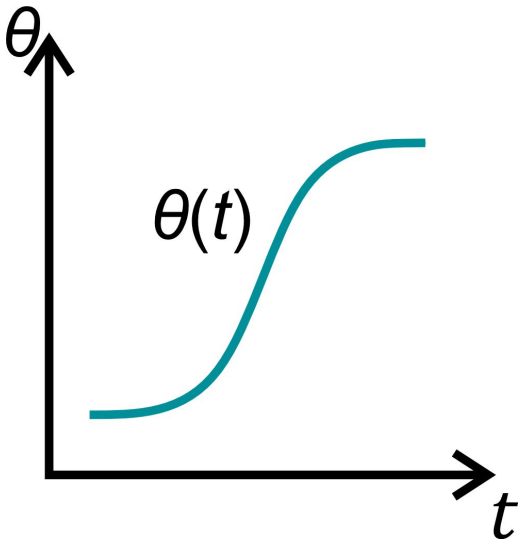
- N = 5 rats
- Rule change unknown to the animal
- Recording from prelimbic prefrontal cortex with multi-electrode silicon probes



Adapted from : Floresco, S. B., Block, A. E., & Tse, M. T. L. (2008). Inactivation of the medial prefrontal cortex of the rat impairs strategy set-shifting, but not reversal learning, using a novel, automated procedure. Behavioural Brain Research.

Nonstationary RNNs





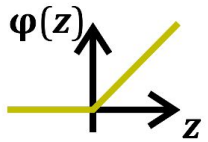
PLRNN: Trial-dependent connectivity matrix

$$z_t = Az_{t-1} + W_k\varphi(z_{t-1}) + h + Cs_t + \varepsilon_t,\ \varepsilon_t \sim N(0,\Sigma)$$ Latent model

External Inputs

$$x_t = B\varphi(z_{t-1}) + \eta_t,\ \eta_t \sim N(0,\Gamma)$$ Observation model

$$\varphi(z) = max(0, z)$$ *ReLU*



Regularization:

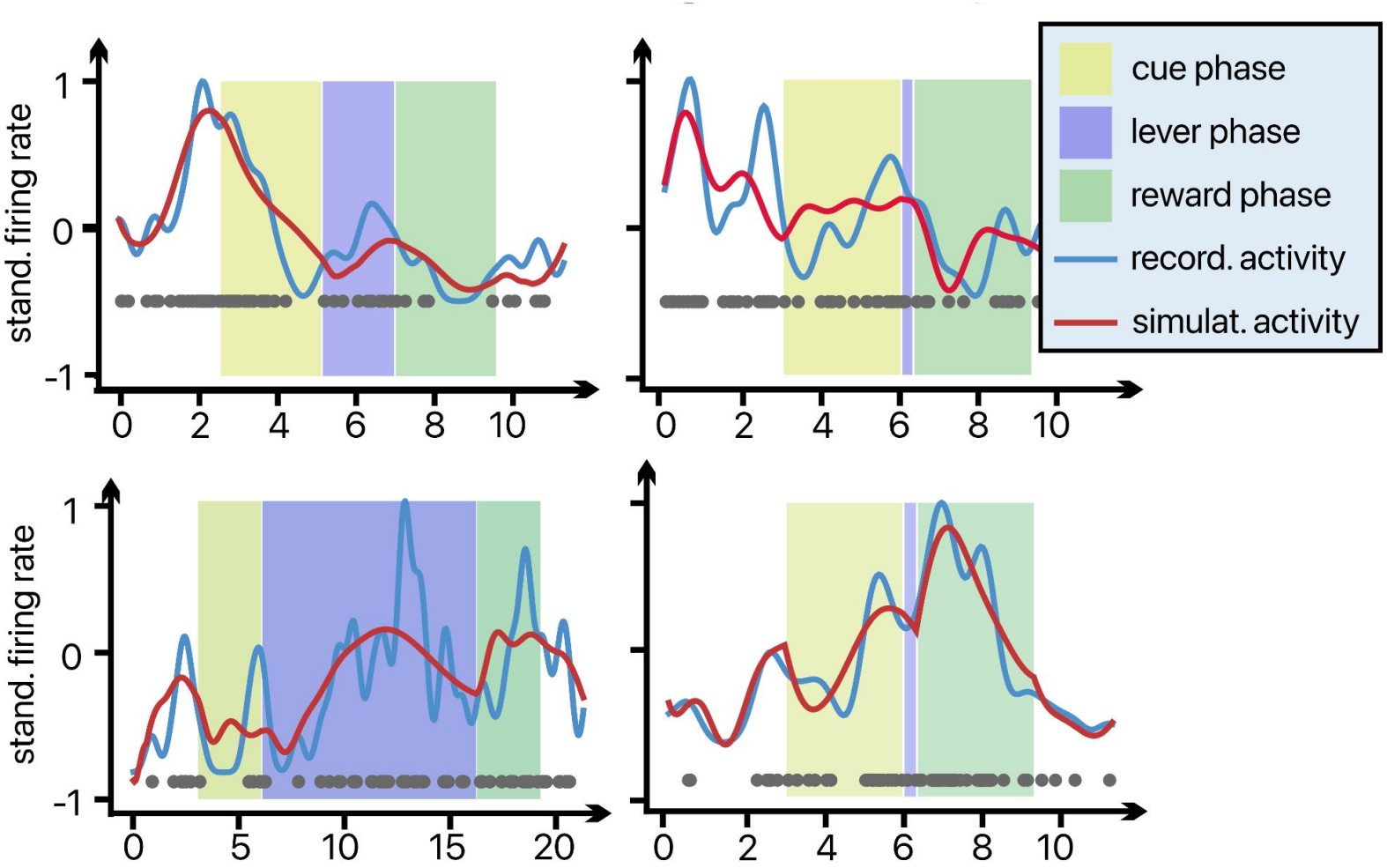
$$Loss(Z, \Theta) = -ELBO(Z, \Theta) + \frac{\lambda_0}{2n}\sum_{k=1}^{N}\|W_k\|_2^2 + \frac{\lambda_1}{2n}\sum_{k=1}^{N-1}\|W_{k+1} - W_k\|_2^2 + \frac{\lambda_2}{2n}\sum_{k=1}^{N-2}\|W_{k+2} - 2W_{k+1} + W_k\|_2^2$$

Weight regularization

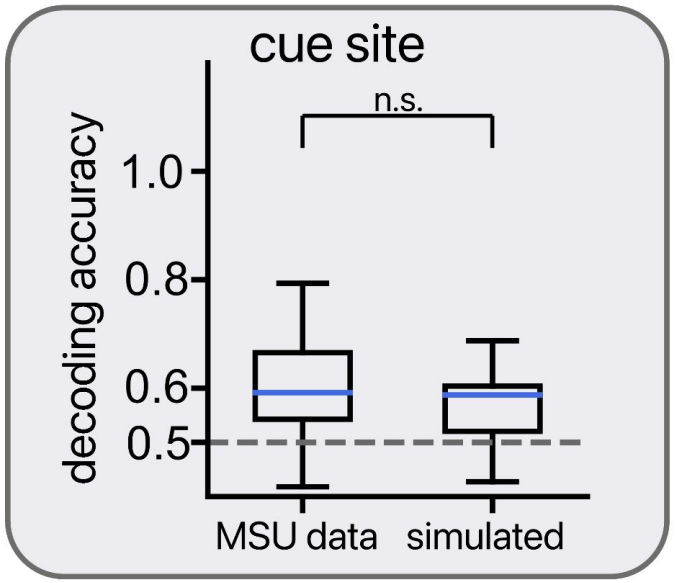
Continuity prior (1^{st}-order)

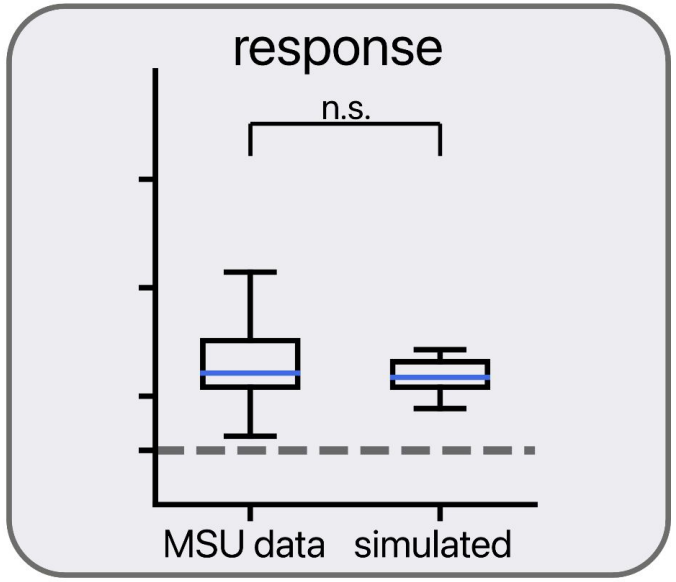
Smoothness prior (2^{nd}-order)

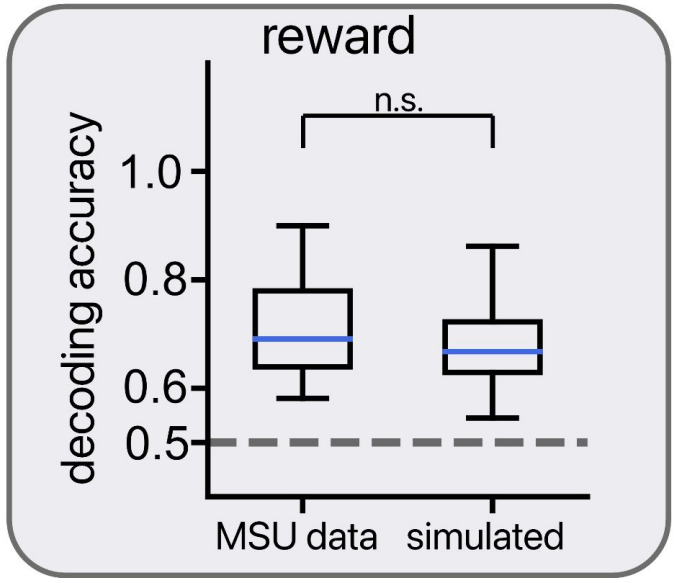
Single unit reconstructions

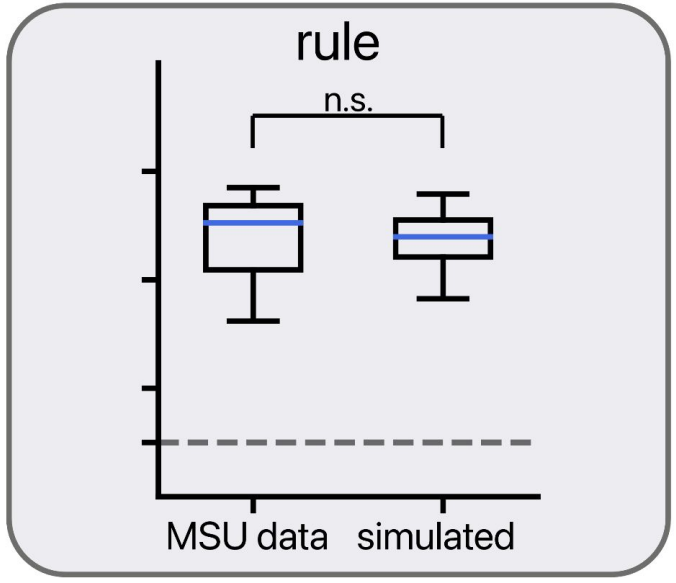


Behavioral validation

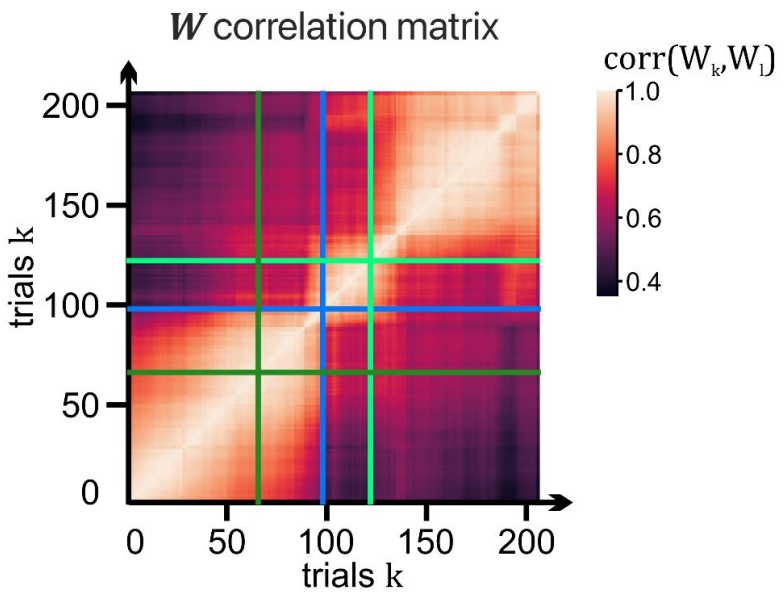


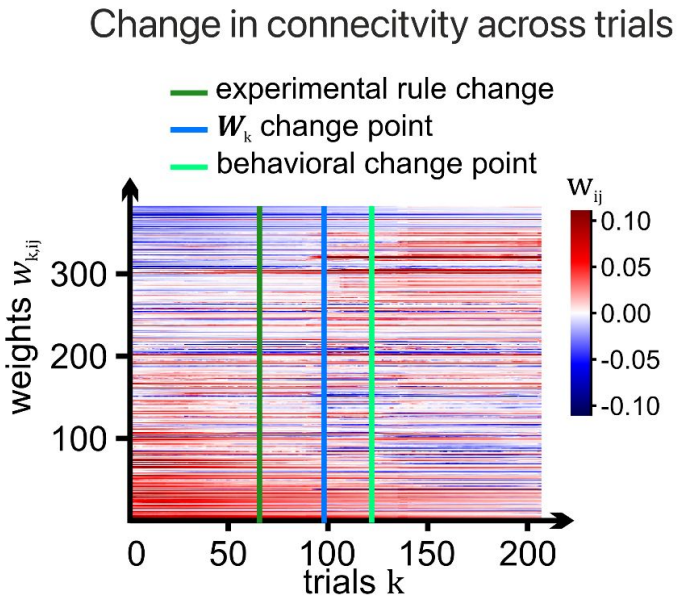


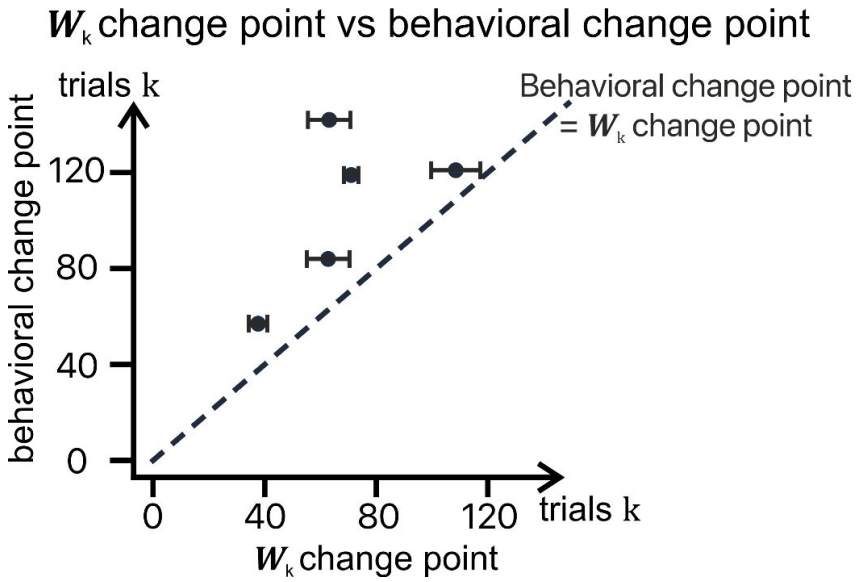


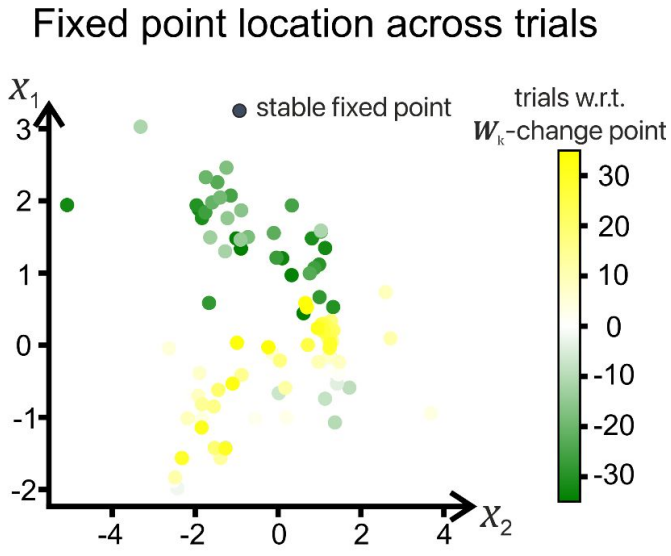


Results


W correlation matrix
corr(W_k,W_l)
1.0
0.8
0.6
0.4
trials k
200
150
100
50
0
0
50
100
150
200
trials k


Change in connecitvity across trials
experimental rule change
W_k change point
behavioral change point
w_ij
0.10
0.05
0.00
-0.05
-0.10
weights w_k,ij
300
200
100
0
50
100
150
200
trials k


W_k change point vs behavioral change point
trials k
Behavioral change point
= W_k change point
behavioral change point
120
80
40
0
0
40
80
120
trials k
W_k change point


Fixed point location across trials
X_1
stable fixed point
trials w.r.t.
W_k-change point
30
20
10
0
-10
-20
-30
3
2
1
0
-1
-2
-4
-2
0
2
4
X_2

Hands-on-Tutorial

…now it’s your turn!

Thank you for your kind attention

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Resolving prefrontal flexibility
FOR5159

STRUCTURES CLUSTER OF EXCELLENCE

Nationales Bernstein Netzwerk Computational Neuroscience

IMMERSE

AI4U Reallabor KI für psychische Gesundheit

Hands on Tutorial

Aim of the hands-on-tutorial section:

- provide a minimal running example of training a model
- provide a practical overview of most important analysis methods of trained models
- clear up general questions about using the algorithm on your own data. If you brought your own dataset, we encourage you to discuss applications and questions like hyperparameter settings with us

Hands on Tutorial

1. Go to https://github.com/DurstewitzLab/CNS-2023
2. Download data and Jupyter Notebook
3. Execute Notebook locally on your laptop or online via Google Colab