Dynamical systems modeling with deep learning tools

Marco Forgione, Dario Piga

IDSIA Dalle Molle Institute for Artificial Intelligence USI-SUPSI, Lugano, Switzerland

April 15, 2021

Context

The connection between deep learning and dynamical system theory is under intensive investigation:

- Analysis of deep architectures through the lens of system theory

  E. Haber and L. Ruthotto, Stable architectures for deep neural networks.
  Inverse Problems, 2017

- Dynamical systems as deep learning layers

  T. Q. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural Ordinary Differential Equations.
  In: *Advances in Neural Information Processing Systems, 2018*

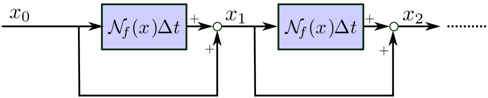
- Modeling of dynamical systems with deep learning tools

  M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
  Journal of Computational Physics, 2019

This connection is beneficial to both fields.

Residual Networks & Dynamical Systems

A Residual Network is equivalent to the forward Euler integration of an underlying continuous-time neural dynamics



$$x_{k+1} = x_k + \mathcal{N}_f(x_k)\Delta t$$

Corresponding ordinary differential equation (ODE):

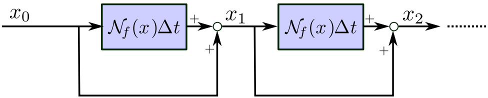
$$\dot{x} = \mathcal{N}_f(x)$$

Related questions:

- Stability/well-posedness of the ResNet related to the ODE?
- Is the neural ODE a useful layer for deep learning?
- Are deep learning tools useful for dynamical systems modeling?

Residual Networks & Dynamical Systems

A Residual Network is equivalent to the forward Euler integration of an underlying continuous-time neural dynamics



$$x_{k+1} = x_k + \mathcal{N}_f(x_k)\Delta t$$

Corresponding ordinary differential equation (ODE):

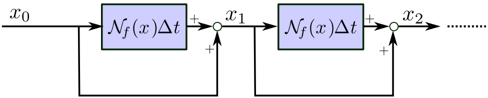
$$\dot{x} = \mathcal{N}_f(x)$$

Related questions:

- Stability/well-posedness of the ResNet related to the ODE?
- Is the neural ODE a useful layer for deep learning?
- Are deep learning tools useful for dynamical systems modeling?

Residual Networks & Dynamical Systems

A Residual Network is equivalent to the forward Euler integration of an underlying continuous-time neural dynamics



$$x_{k+1} = x_k + \mathcal{N}_f(x_k)\Delta t$$

Corresponding ordinary differential equation (ODE):

$$\dot{x} = \mathcal{N}_f(x)$$

Related questions:

- Stability/well-posedness of the ResNet related to the ODE?
- Is the neural ODE a useful layer for deep learning?
- Are deep learning tools useful for dynamical systems modeling?

Tailor-made state-space neural model structures

Motivations

Neural state-space models (Recurrent Neural Networks) are widely used for dynamical systems. However, they seldom exploit a priori knowledge.

We present:

- tailor-made neural model structures for system identification;
- efficient algorithms to fit these model structures to data.

M. Forgione and D. Piga. Continuous-time system identification with neural networks: model structures and fitting criteria. *European Journal of Control, 59(2021), pp 69-81*

B. Mavkov, M. Forgione and D. Piga. Integrated Neural Networks for Nonlinear Continuous-Time System Identification. *IEEE Control Systems Letters, 4(4), pp 851-856, 2020.*

M. Forgione and D. Piga. Model structures and fitting criteria for system identification with neural networks. *In Proc. of the 14th. IEEE Application of Information and Communication Technologies Conference (AICT), 2020*

https://github.com/forgi86/sysid-neural-continuous

https://github.com/bmavkov/INN-for-Identification

https://github.com/forgi86/sysid-neural-structures-fitting

Motivations

Neural state-space models (Recurrent Neural Networks) are widely used for dynamical systems. However, they seldom exploit a priori knowledge.

We present:

- tailor-made neural model structures for system identification;
- efficient algorithms to fit these model structures to data.

M. Forgione and D. Piga. Continuous-time system identification with neural networks: model structures and fitting criteria. *European Journal of Control, 59(2021), pp 69-81*

B. Mavkov, M. Forgione and D. Piga. Integrated Neural Networks for Nonlinear Continuous-Time System Identification. *IEEE Control Systems Letters, 4(4), pp 851-856, 2020.*

M. Forgione and D. Piga. Model structures and fitting criteria for system identification with neural networks. *In Proc. of the 14th. IEEE Application of Information and Communication Technologies Conference (AICT), 2020*

https://github.com/forgi86/sysid-neural-continuous

https://github.com/bmavkov/INN-for-Identification

https://github.com/forgi86/sysid-neural-structures-fitting

Motivations

Neural state-space models (Recurrent Neural Networks) are widely used for dynamical systems. However, they seldom exploit a priori knowledge.

We present:

- tailor-made neural model structures for system identification;
- efficient algorithms to fit these model structures to data.

M. Forgione and D. Piga. Continuous-time system identification with neural networks: model structures and fitting criteria. *European Journal of Control, 59(2021), pp 69-81*

B. Mavkov, M. Forgione and D. Piga. Integrated Neural Networks for Nonlinear Continuous-Time System Identification. *IEEE Control Systems Letters, 4(4), pp 851-856, 2020.*

M. Forgione and D. Piga. Model structures and fitting criteria for system identification with neural networks. *In Proc. of the 14th. IEEE Application of Information and Communication Technologies Conference (AICT), 2020*

https://github.com/forgi86/sysid-neural-continuous

https://github.com/bmavkov/INN-for-Identification

https://github.com/forgi86/sysid-neural-structures-fitting

Motivations

Neural state-space models (Recurrent Neural Networks) are widely used for dynamical systems. However, they seldom exploit a priori knowledge.

We present:

- tailor-made neural model structures for system identification;
- efficient algorithms to fit these model structures to data.

M. Forgione and D. Piga. Continuous-time system identification with neural networks: model structures and fitting criteria. *European Journal of Control, 59(2021), pp 69-81*

B. Mavkov, M. Forgione and D. Piga. Integrated Neural Networks for Nonlinear Continuous-Time System Identification. *IEEE Control Systems Letters, 4(4), pp 851-856, 2020.*

M. Forgione and D. Piga. Model structures and fitting criteria for system identification with neural networks. *In Proc. of the 14th. IEEE Application of Information and Communication Technologies Conference (AICT), 2020*

`https://github.com/forgi86/sysid-neural-continuous`

`https://github.com/bmavkov/INN-for-Identification`

`https://github.com/forgi86/sysid-neural-structures-fitting`

Settings

The true system is assumed to have a state-space representation:

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t))\\ y^{\circ}(t) &= g(x(t)),\end{aligned}$$

with noisy discrete-time measurements available: $y_k = y^{\circ}(t_k) + e_k$.

Training dataset $\mathcal{D}$: a single input/output sequence with:

- input samples $U = \{u_0,\ u_1, \ldots,\ u_{N-1}\}$
- output samples $Y = \{y_0,\ y_1, \ldots,\ y_{N-1}\}$
- time instants $T = \{t_0,\ t_1, \ldots,\ t_{N-1}\}$

Objective: estimate a dynamical model of the system.

Settings

The true system is assumed to have a state-space representation:

$$\dot{x}(t) = f(x(t), u(t))$$
$$y^{\circ}(t) = g(x(t)),$$

with noisy discrete-time measurements available: $y_k = y^{\circ}(t_k) + e_k$.

Training dataset $\mathcal{D}$: a single input/output sequence with:

- input samples $U = \{u_0,\ u_1, \ldots,\ u_{N-1}\}$
- output samples $Y = \{y_0,\ y_1, \ldots,\ y_{N-1}\}$
- time instants $T = \{t_0,\ t_1, \ldots,\ t_{N-1}\}$

Objective: estimate a dynamical model of the system.

Settings

The true system is assumed to have a state-space representation:

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t))\\ y^{\circ}(t) &= g(x(t)),\end{aligned}$$

with noisy discrete-time measurements available: $y_k = y^{\circ}(t_k) + e_k$.

Training dataset $\mathcal{D}$: a single input/output sequence with:

- input samples $U = \{u_0,\ u_1, \ldots,\ u_{N-1}\}$
- output samples $Y = \{y_0,\ y_1, \ldots,\ y_{N-1}\}$
- time instants $T = \{t_0,\ t_1, \ldots,\ t_{N-1}\}$

Objective: estimate a dynamical model of the system.

Neural Dynamical Models

A very generic neural model structure:

$$\dot{x} = \mathcal{N}_f(x, u;\ \theta)$$
$$y = \mathcal{N}_g(x; \theta)$$

where $\mathcal{N}_f$, $\mathcal{N}_g$ are feed-forward neural networks.

Can be specialized, according to available system knowledge:

- State fully observed $\Rightarrow$

$$\dot{x} = \mathcal{N}_f(x, u;\ \theta)$$
$$y = x$$

- ...

Neural Dynamical Models

A very generic neural model structure:

$$
\begin{aligned}
\dot{x} &= \mathcal{N}_f(x, u;\ \theta) \\
y &= \mathcal{N}_g(x; \theta)
\end{aligned}
$$

where $\mathcal{N}_f$, $\mathcal{N}_g$ are feed-forward neural networks.

Can be specialized, according to available system knowledge:

- State fully observed $\Rightarrow$

$$
\begin{aligned}
\dot{x} &= \mathcal{N}_f(x, u;\ \theta) \\
y &= x
\end{aligned}
$$

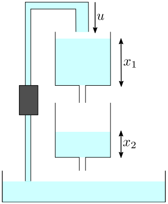
- ...

Neural Dynamical Models

Physics-inspired model structures

Two-tank system. Input: flow u in upper tank, output: lower tank level x_2.

- The system has two states: x_1 and x_2
- State x_1 does not depend on x_2
- State x_2 does not depend directly on u
- The state x_2 is measured

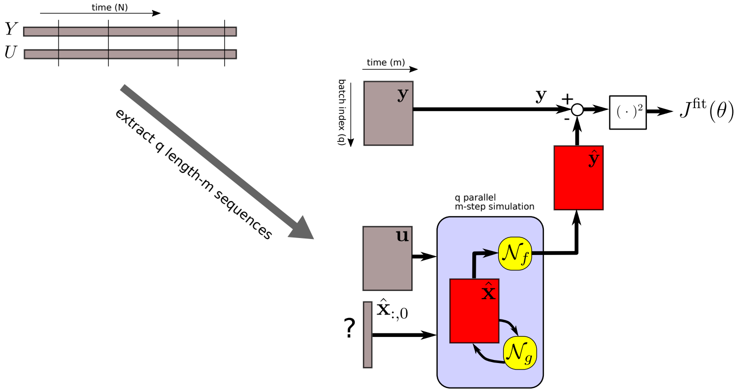


These observations are embedded in the physics-inspired neural model:

$$\dot{x}_1 = \mathcal{N}_1(x_1, u;\ \theta)$$
$$\dot{x}_2 = \mathcal{N}_2(x_1, x_2;\ \theta)$$
$$y = x_2$$

Training Neural Dynamical Models

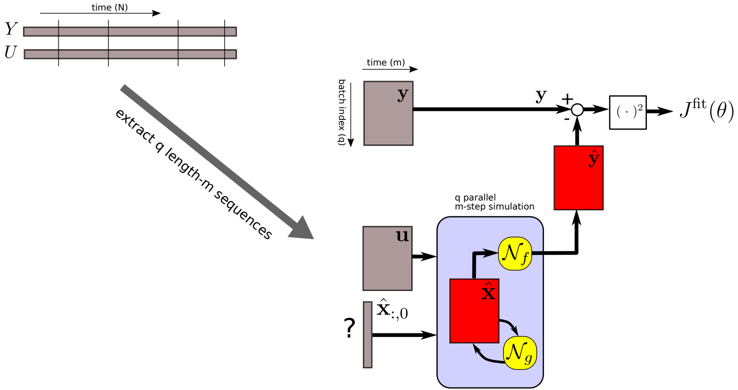
How to fit the network efficiently to a single, long training sequence?
How to split it in sub-sequences for minibatch training?



Problem: how do we choose $\hat{x}_{:,0}$, the initial state for each sub-sequence?
We need it to initialize all simulations.

Training Neural Dynamical Models

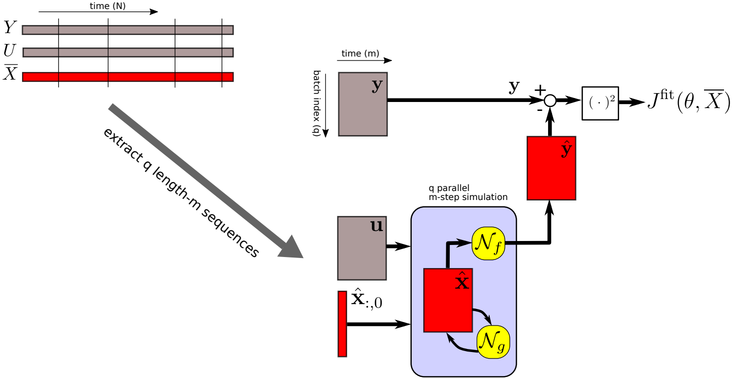
How to fit the network efficiently to a single, long training sequence?
How to split it in sub-sequences for minibatch training?



Problem: how do we choose $\hat{\boldsymbol{x}}_{:,0}$, the initial state for each sub-sequence?
We need it to initialize all simulations.

Training Neural Dynamical Models

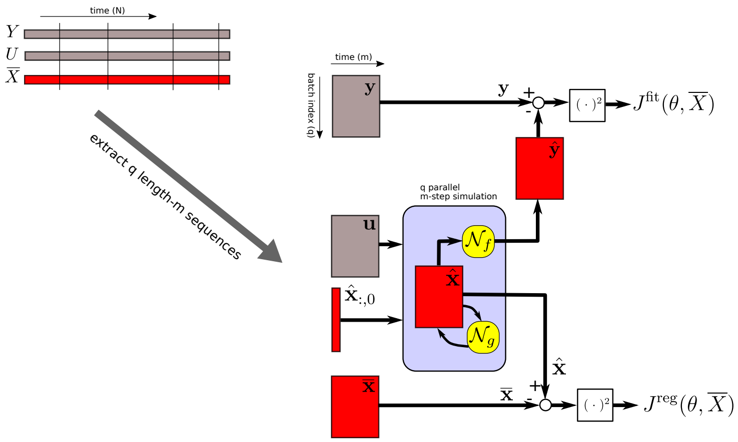
We consider the unknown state sequence $\overline{X}$ as an optimization variable.
We sample from $\overline{X}$ to obtain the initial state for simulation in each batch.



$J^{\rm fit}$ is now a function of both θ and $\overline{X}$. We optimize w.r.t. both!

Training Neural Dynamical Models

The hidden state sequence $\overline{X}$ should also satisfy the identified dynamics!
We enforce this by adding a regularization term in the cost function.



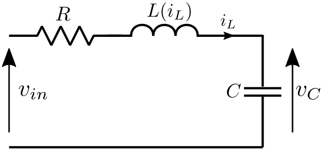
We minimize a weighted sum of J^{fit} and J^{reg} w.r.t. both θ and $\overline{X}$.

Example

RLC circuit

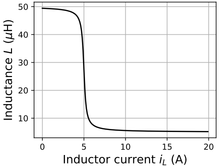
We consider a nonliner RLC circuit:

$$\begin{bmatrix}\dot v_C \\ \dot i_L\end{bmatrix} = \begin{bmatrix}0 & \frac{1}{C} \\ \frac{-1}{L(i_L)} & \frac{-R}{L(i_L)}\end{bmatrix}\begin{bmatrix}v_C \\ i_L\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{L(i_L)}\end{bmatrix} v_{in}$$



with nonlinear inductance $L(i_L)$

$$L(i_L) = L_0\left[\left(\frac{0.9}{\pi}\arctan\big(-5(|i_L|-5)+0.5\right)+0.1\right]$$



Input: voltage v_{in}. Output: voltage v_C, current i_L. SNR=20

Neural model structure: fully observed state

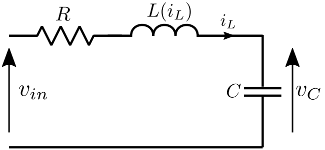
$$\dot x = \mathcal{N}_f(x, u; \theta)$$

$$y = x$$

Example

RLC circuit

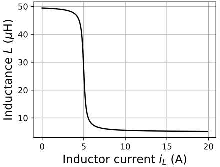
We consider a nonliner RLC circuit:



$$\begin{bmatrix}\dot{v}_C \\ \dot{i}_L\end{bmatrix} = \begin{bmatrix}0 & \frac{1}{C} \\ \frac{-1}{L(i_L)} & \frac{-R}{L(i_L)}\end{bmatrix}\begin{bmatrix}v_C \\ i_L\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{L(i_L)}\end{bmatrix} v_{in}$$

with nonlinear inductance $L(i_L)$

$$L(i_L) = L_0\left[\left(\frac{0.9}{\pi}\arctan\left(-5(|i_L|-5\right)+0.5\right)+0.1\right]$$



Input: voltage v_{in}. Output: voltage v_C, current i_L. SNR=20

Neural model structure: fully observed state

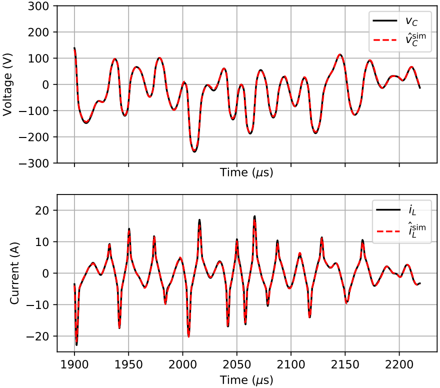
$$\dot{x} = \mathcal{N}_f(x, u; \theta)$$

$$y = x$$

Example
RLC circuit

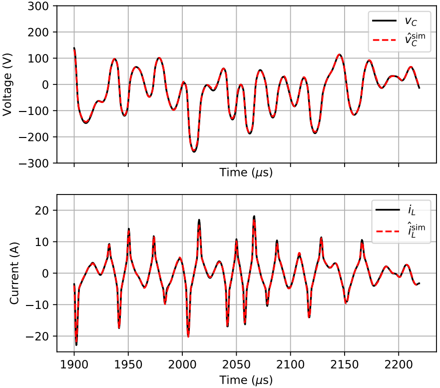
Results on the test dataset. Training with:

$q = 64$ sequences of length $m = 64$



train time: 320 s

$q = 1$ sequence of length $m = 4000$

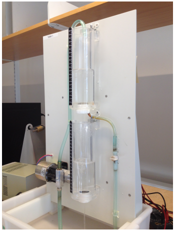


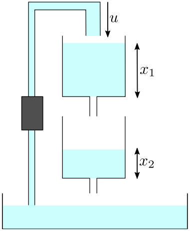
train time: 7000 s

Example

Cascaded Tank System

Dataset with real measurements from `www.nonlinearbenchmark.org`





Neural model structure: physics-inspired

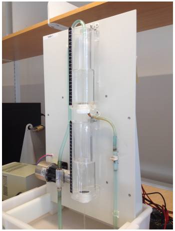
$$\dot{x}_1 = \mathcal{N}_1(x_1, u; \theta)$$
$$\dot{x}_2 = \mathcal{N}_2(x_1, x_2, u; \theta)$$
$$y = x_2$$

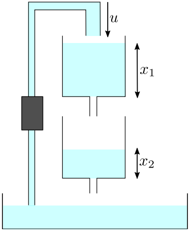
The dependency of $\mathcal{N}_2$ on u models water overflow from upper tank.

Example

Cascaded Tank System

Dataset with real measurements from `www.nonlinearbenchmark.org`





Neural model structure: physics-inspired

$$
\begin{aligned}
\dot{x}_1 &= \mathcal{N}_1(x_1, u; \theta) \\
\dot{x}_2 &= \mathcal{N}_2(x_1, x_2, u; \theta) \\
y &= x_2
\end{aligned}
$$

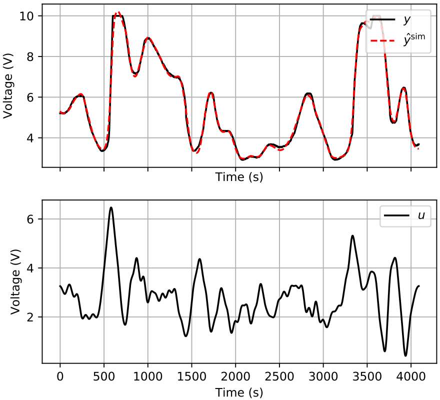
The dependency of $\mathcal{N}_2$ on u models water overflow from upper tank.

Numerical example

Cascaded Tank System

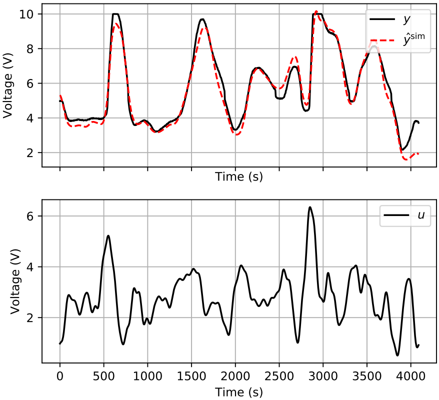
Training with $q = 64$ subsequences of length $m = 128$. Results on:

Training dataset



$R^2 = 0.99$, RMSE $= 0.08$ V

Test dataset



$R^2 = 0.97$, RMSE $= 0.33$ V

Conclusions

We have presented tailor-made neural structures for system identification embedding a priori knowledge.

We have shown how to parallelize the training using batches of short-size subsequences, and taking into account the effect of the initial condition.

Current/Future work

- Estimation and control algorithms
- Learning of Partial Differential Equations

Conclusions

We have presented tailor-made neural structures for system identification embedding a priori knowledge.

We have shown how to parallelize the training using batches of short-size subsequences, and taking into account the effect of the initial condition.

Current/Future work

- Estimation and control algorithms
- Learning of Partial Differential Equations