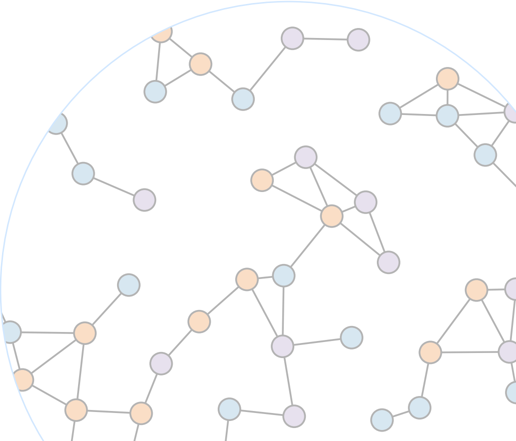
# NetSci 2020: Network Embedding (Session 4E)

# Deep Learning of Epidemics Spreading on Complex Networks

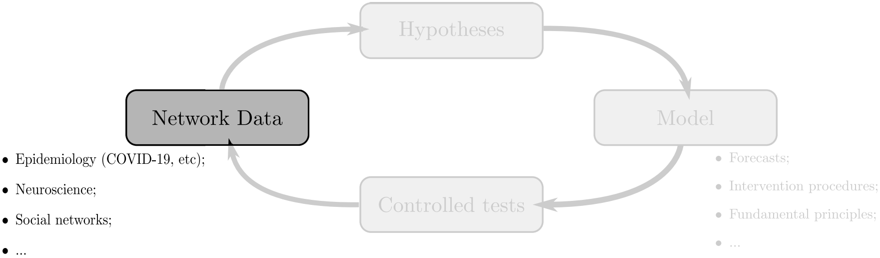
**Charles Murphy**, Edward Laurence, Antoine Allard

September 22nd, 2020

Département de physique, de génie physique, et d'optique
Université Laval, Québec (Qc, Canada)

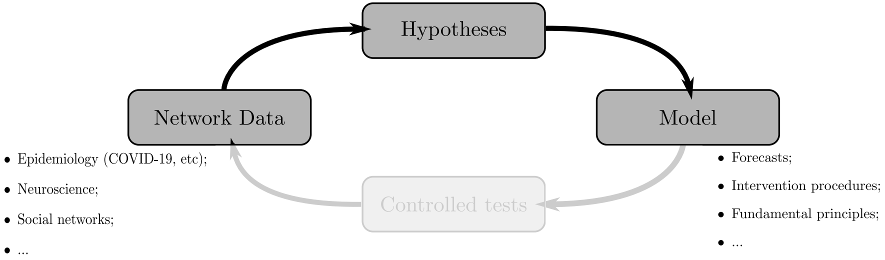


## How we build models



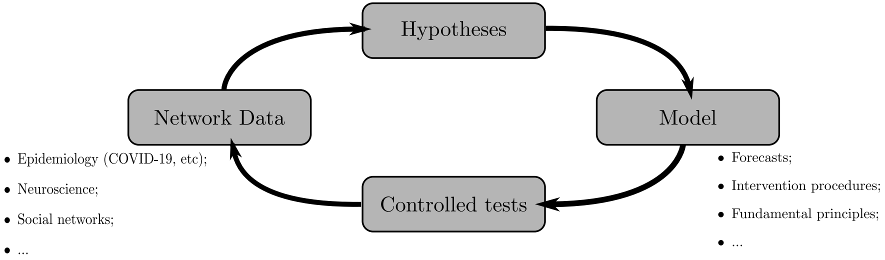

- Hard to gain insight;
- Few data points.

## How we build models



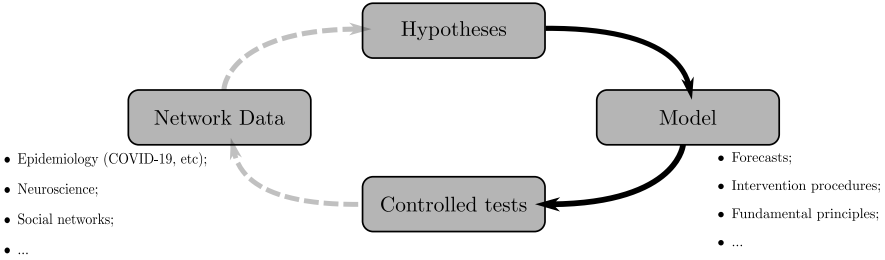

- Hard to gain insight;
- Few data points.

## How we build models



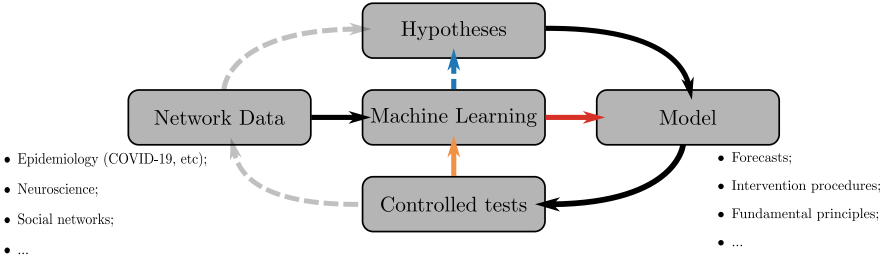

- Hard to gain insight;
- Few data points.

## How we build models




- Hard to gain insight;
- Few data points.

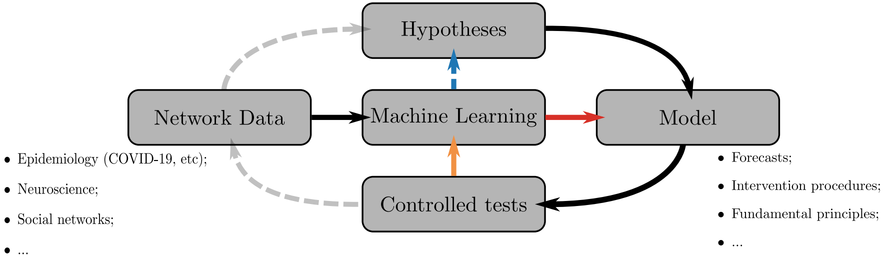
## A numerical Petri dish




- Even non-interpretable ML models *can inspire us* to build better models;
- We can *use them directly* for prediction;
- They allow us to *build controlled environments*.

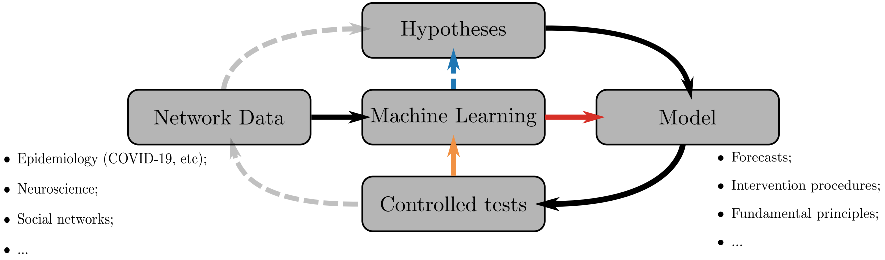
# Model building in Network Science (revisited)

## A numerical Petri dish



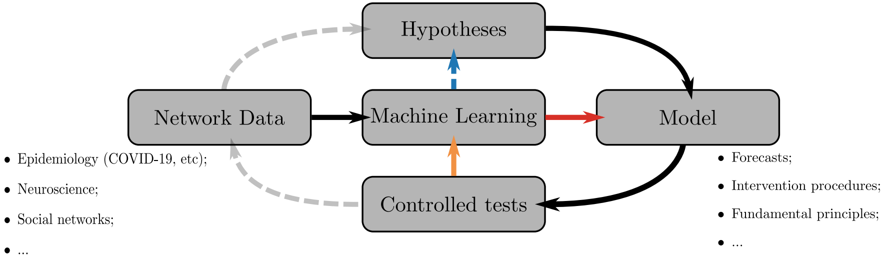

- Even non-interpretable ML models *can inspire us* to build better models;
- We can *use them directly* for prediction;
- They allow us to *build controlled environments*.

## A numerical Petri dish




- Even non-interpretable ML models *can inspire us* to build better models;
- We can *use them directly* for prediction;
- They allow us to *build controlled environments*.

## A numerical Petri dish



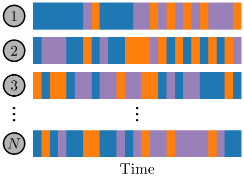

- Even non-interpretable ML models *can inspire us* to build better models;
- We can *use them directly* for prediction;
- They allow us to *build controlled environments*.

## A couple of interesting papers to look at

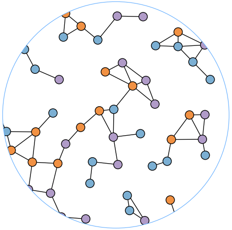
- B. Lusch, N. J. Kutz, S. L. Brunton, "*Deep learning for universal linear embeddings of nonlinear dynamics*", Nat. Commun. **9**, 4950 (2018).
- J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, "*Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach*", Phys. Rev. Lett. **120**, 024102 (2018).
- F. A. Rodrigues, T. Peron, C. Connaughton, J. Kurths, Y. Moreno, "*A machine learning approach to predicting dynamical observables from network structure*", `arxiv:1910.00544` (2019).
- C. Shah, N. Dehmamy, N. Perra, M. Chinazzi, A.-L. Barabási, A. Vespignani, R. Yu, "*Finding Patient Zero: Learning Contagion Source with Graph Neural Networks*", `arxiv:2006.11913` (2020).
- *And many more.*

## Assumptions

*Time series* $X \sim M(G)$




*Network* $G = (\mathscr{V}, \mathscr{E})$



*Time invariance*

$$\boxed{M(G;t)} = \boxed{M(G;t+\tau)}$$

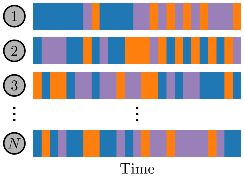
## Objectives

- Train a model $\hat{M}(G';\Theta)$ such that $\hat{M}(G';\Theta) \approx M(G')$—*ideally* for any $G'$;
- $\hat{M}(G;\Theta)$ is a graph neural network (GNN) with trainable parameters $\Theta$;

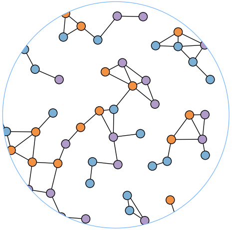
# Overview of our strategy

## Assumptions

*Time series* $X \sim M(G)$




*Network* $G = (\mathscr{V}, \mathscr{E})$



*Time invariance*

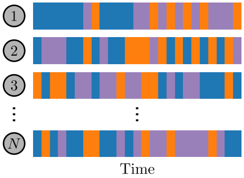
$$M(G;t) = M(G;t+\tau)$$

## Objectives

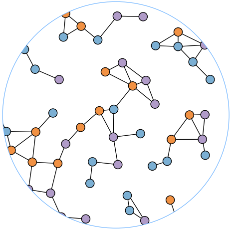
- Train a model $\hat{M}(G'; \boldsymbol{\Theta})$ such that $\hat{M}(G'; \boldsymbol{\Theta}) \approx M(G')$—*ideally* for any $G'$;
- $\hat{M}(G; \boldsymbol{\Theta})$ is a graph neural network (GNN) with trainable parameters $\boldsymbol{\Theta}$;

## Assumptions

*Time series* $X \sim M(G)$




*Network* $G = (\mathscr{V}, \mathscr{E})$



*Time invariance*

$$\boxed{M(G;t)} = \boxed{M(G;t+\tau)}$$

---

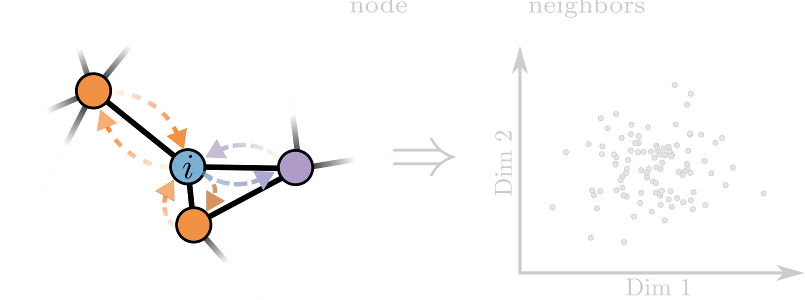
## Objectives

- Train a model $\hat{M}(G'; \boldsymbol{\Theta})$ such that $\hat{M}(G'; \boldsymbol{\Theta}) \approx M(G')$—*ideally* for any $G'$;
- $\hat{M}(G; \boldsymbol{\Theta})$ is a graph neural network (GNN) with trainable parameters $\boldsymbol{\Theta}$;

# Graph Neural Network

$\Rightarrow$

$$\left[\mathrm{GNN}(\{\bigcirc\}, G)\right]_i = \mathrm{NN}\left(\underbrace{\textcircled{\scriptsize i}}_{\text{node}}, \mathrm{AGG}(\underbrace{\bigcirc, \cdots, \bigcirc}_{\text{neighbors}}; \Theta_{\mathrm{AGG}}); \Theta_{\mathrm{NN}}\right)$$

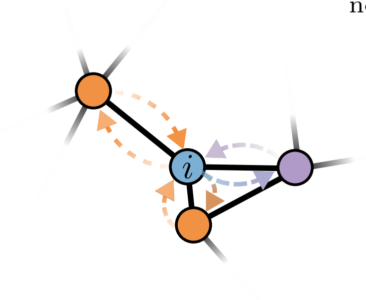



- A graph neural network receives as its input the *node features* and the *network*;
- NN and AGG are *both* trainable neural networks;
- AGG aggregates the features of a node's neighborhood *locally*;
- Usually used for *network embedding task* and *structure learning*.

# Graph Neural Network

$\Rightarrow$

$$\left[\mathrm{GNN}(\{\bigcirc\}, G)\right]_i = \mathrm{NN}\Big(\underbrace{\,\textcircled{$i$}\,}_{\text{node}}, \mathrm{AGG}(\underbrace{\bullet\,, \cdots, \bullet}_{\text{neighbors}}; \boldsymbol{\Theta}_{\mathrm{AGG}}); \boldsymbol{\Theta}_{\mathrm{NN}}\Big)$$




$\Longrightarrow$

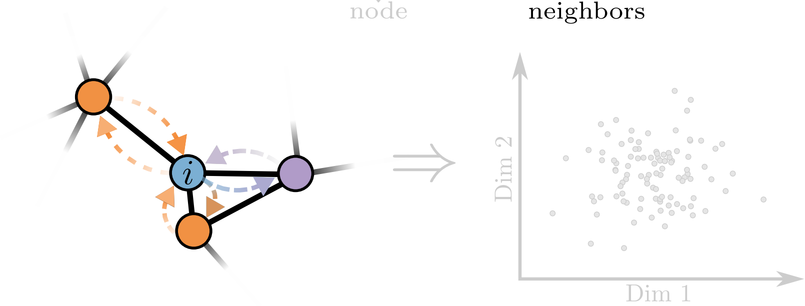



- A graph neural network receives as its input the *node features* and the *network*;
- NN and AGG are *both* trainable neural networks;
- AGG aggregates the features of a node's neighborhood *locally*;
- Usually used for *network embedding task* and *structure learning*.

# Graph Neural Network

⇒

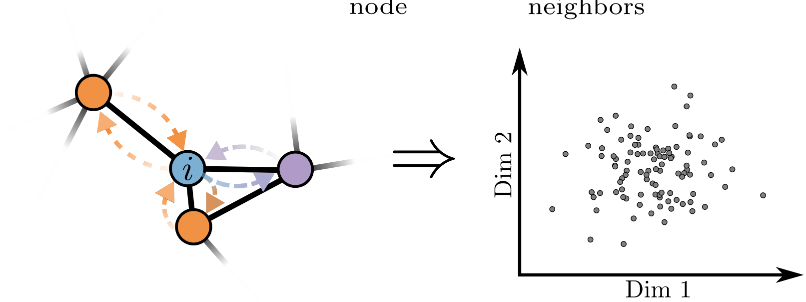
$$\left[\mathrm{GNN}(\{\bigcirc\}, G)\right]_i = \mathrm{NN}\left(\underbrace{\textcircled{i}}_{\text{node}}, \mathrm{AGG}(\underbrace{\bullet, \cdots, \bullet}_{\text{neighbors}}; \boldsymbol{\Theta}_{\mathrm{AGG}}); \Theta_{\mathrm{NN}}\right)$$




- A graph neural network receives as its input the *node features* and the *network*;
- NN and AGG are *both* trainable neural networks;
- AGG aggregates the features of a node's neighborhood *locally*;
- Usually used for *network embedding task* and *structure learning*.

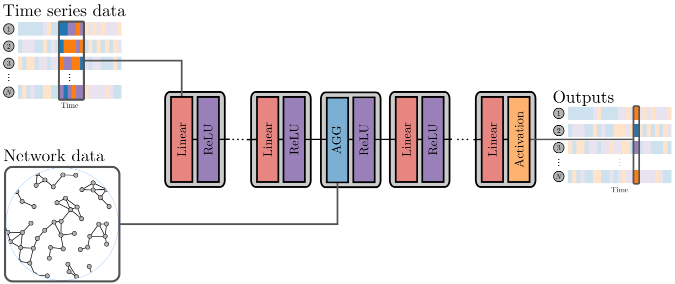
# Graph Neural Network

$$\Big[\mathrm{GNN}(\{\bigcirc\}, G)\Big]_i = \mathrm{NN}\Big(\underbrace{\textcircled{\scriptsize $i$}}_{\text{node}}, \mathrm{AGG}(\underbrace{\bigcirc, \cdots, \bigcirc}_{\text{neighbors}}; \boldsymbol{\Theta}_{\mathrm{AGG}}); \boldsymbol{\Theta}_{\mathrm{NN}}\Big)$$




- A graph neural network receives as its input the *node features* and the *network*;
- NN and AGG are *both* trainable neural networks;
- AGG aggregates the features of a node's neighborhood *locally*;
- Usually used for *network embedding task* and *structure learning*.

# Architecture for learning dynamical systems on networks




- Inputs:
  - States $X(t) = \big(x_i(t)\big)_{i\in\mathcal{V}}$;
  - Network $G = (\mathcal{V}, \mathcal{E})$.
- Outputs:
  - $\mathrm{GNN}(X(t), G) = \hat{Y}(t) = \big(\hat{y}_i(t)\big)_{i\in\mathcal{V}}$;
  - If $x_i(t)$ is discrete, then $\hat{y}_i(t)$ is a transition prob. vector of node $i$ at time $t+1$;
  - If $x_i(t)$ continuous, then $\hat{y}_i(t)$ predicts the state of node $i$ at time $t+1$.

# Applications and Results: Learning Epidemics on Networks

## Susceptible-infected-susceptible dynamics (SIS)

*Discrete states*

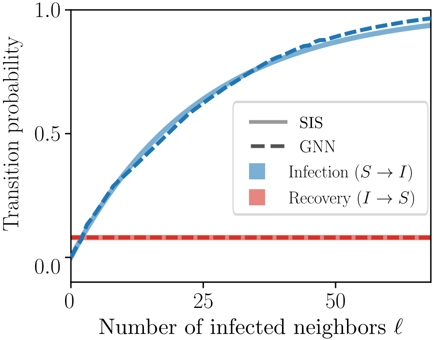
$$x_i(t) \in \{\bigcirc, \bigcirc\}$$

*Infection*

$$P(\bigcirc \to \bigcirc | \ell) = 1 - (1 - \beta)^{\ell}$$

*Recovery*

$$P(\bigcirc \to \bigcirc) = \gamma$$




*Training specifics:*

SIS with $\beta = 0.04$, $\gamma = 0.08$; Barabási-Albert network with $|\mathcal{V}| = 1000$ nodes and $\langle k \rangle = 4$; GNN model with $|\mathbf{\Theta}| \sim 5000$ parameters; Training dataset size of 10000 time steps.

## SIS dynamics with non-monotonic infection function (NM-SIS)

*Discrete states*

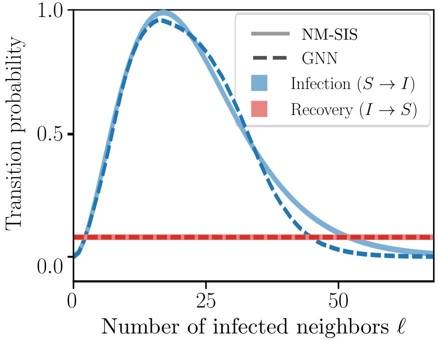
$x_i(t) \in \{$ ● , ● $\}$

*Infection* (Planck-like)

$$P(\text{●} \to \text{●} | \ell) = \frac{1}{Z(\eta)} \frac{\ell^3}{e^{-\eta\ell} - 1}$$

*Recovery*

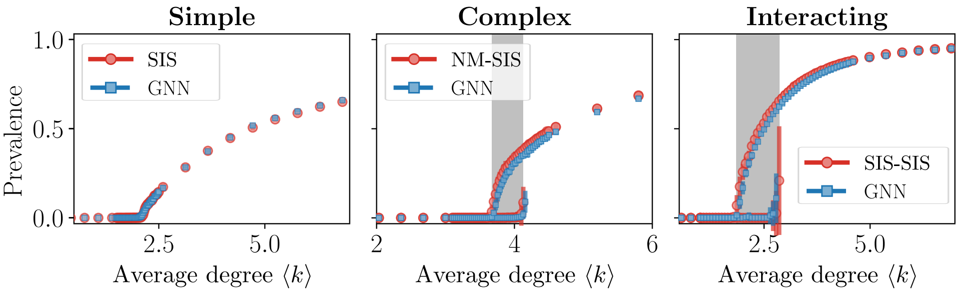
$$P(\text{●} \to \text{●}) = \gamma$$




*Training specifics:*

NM-SIS with $\eta = 10$, $\gamma = 0.08$; Barabási-Albert network with $|\mathcal{V}| = 1000$ nodes and $\langle k \rangle = 4$; GNN model with $|\mathbf{\Theta}| \sim 5000$ parameters (same as SIS); Training dataset size of 10000 time steps.

## Bifurcation diagrams on Erdős-Rényi networks




**Simple**: SIS, **Complex**: Non-monotonic SIS, **Interacting**: SIS-SIS dynamics

We sample 100 Erdős-Rényi networks of size $N = 2000$ with different $\langle k \rangle$ and use the GNN to predict the prevalence.

## Metapopulation SIR dynamics on weighted networks

*Continuous states*

$$\Big(s_j(t), i_j(t), r_j(t)\Big) \in [0,1]^3$$

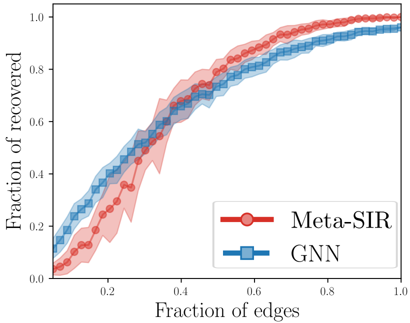
*Dynamical system*

$$\dot{s}_j = -s_j \sum_k \frac{w_{jk}}{w_j} \alpha(i_k)$$

$$\dot{i}_j = -\gamma i_j + s_j \sum_k \frac{w_{jk}}{w_j} \alpha(i_k)$$

$$\dot{r}_j = \gamma i_j$$

$$\alpha(i_j) = 1 - (1 - \beta / N_j)^{i_j N_j}$$




*Training specifics:*

Metapopulation SIR with $\beta = 1.08$, $\gamma = 0.13$; 10 Barabási-Albert networks with $|\mathcal{V}| = 100$ nodes, $\langle k \rangle = 2$, $N_j \sim \mathcal{N}(10^4, 1)$, $\omega_{jk} \sim \mathcal{U}(0, 100)$; GNN model with $|\boldsymbol{\Theta}| \sim 400\,000$ parameters; Training dataset size of 1000 time steps.

# Conclusion

## Take-home message

1. Graph neural networks can mimic epidemic spreading;
2. Highly versatile *(contagion dynamics, metapopulation, weighted networks, etc.)*;
3. If you have network data with time series, **consider using our approach**[1].

## Perspectives

1. Other systems and datasets;
2. Generalized structures *(multiplex, simplicial complexes, etc.)*;
3. Various applications *(network defects detection*[2]*, resilience analysis, etc.)*.

---

[1] *Codes available soon via GitHub.*

[2] Detecting structural perturbations from time series with deep learning, `arXiv:2006.05232`

# Thank you


## Special thanks to my collaborators:

E. Laurence (edwardlaurence.me)

A. Allard (antoineallard.info)



## To contact me:

Email: charles.murphy.1@ulaval.ca


## Pre-prints:

*Main paper*: Deep learning of stochastic contagion dynamics on complex networks, arXiv:2006.05410.

*See also*: Detecting structural perturbations from time series with deep learning, arXiv:2006.05232.

## GitHub:

*Available soon.*