Course #4:

Recurrent Neural Networks

Roadmap

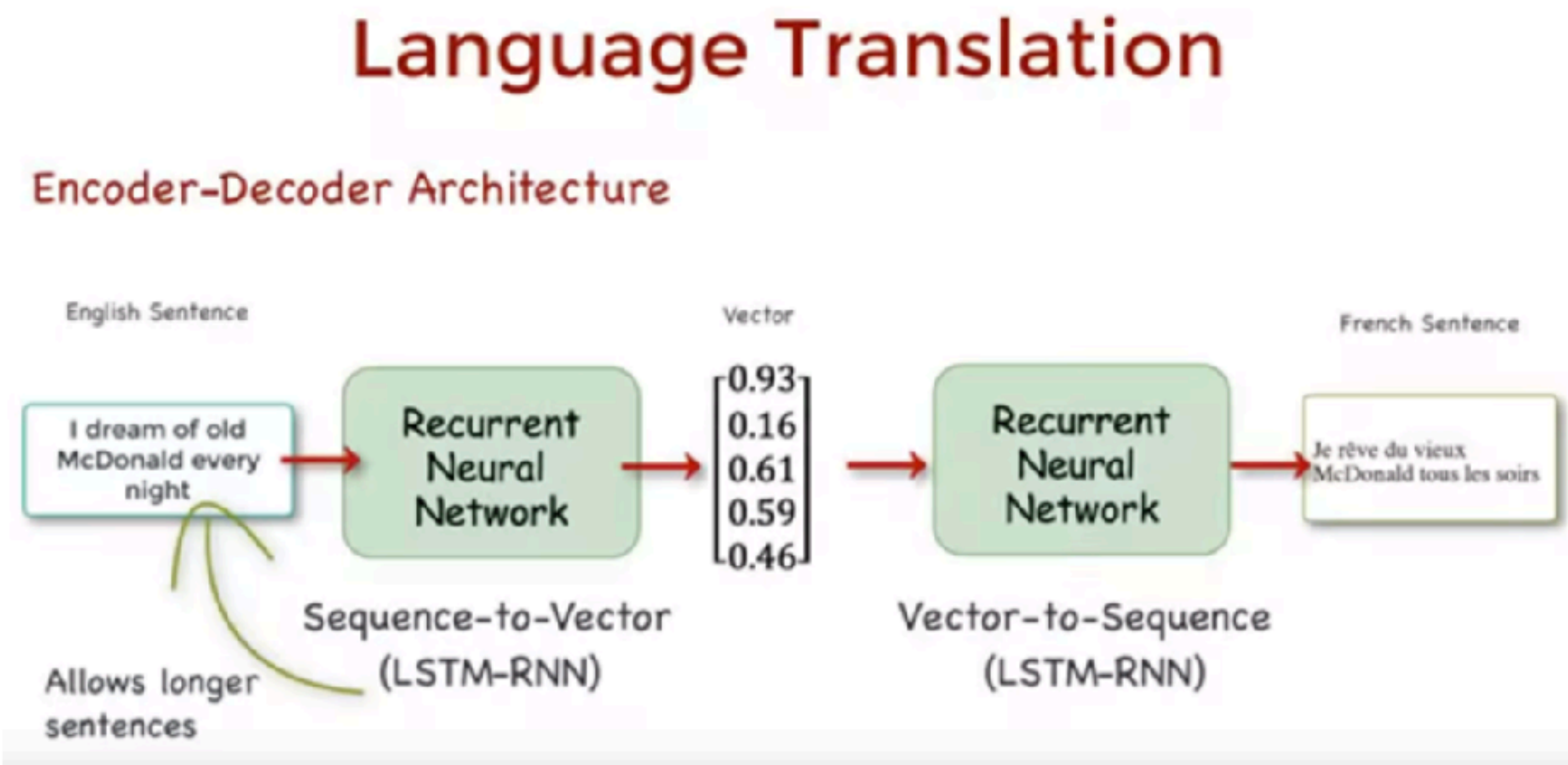
- Recap from course #3
- Auto-encoders
- Recurrents Neural Networks

Lecture. #3
Things to know (AE)

- Auto-encoder
- Latent variable
- UNet
- ResNet

Recurrent Neural Networks

Application to text data



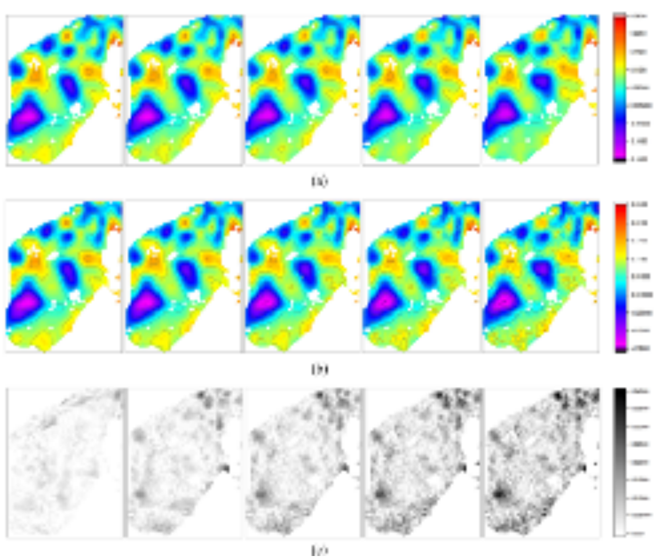
https://towardsdatascience.com/understanding-neural-machine-translation-encoder-decoder-architecture-80f205643ba4

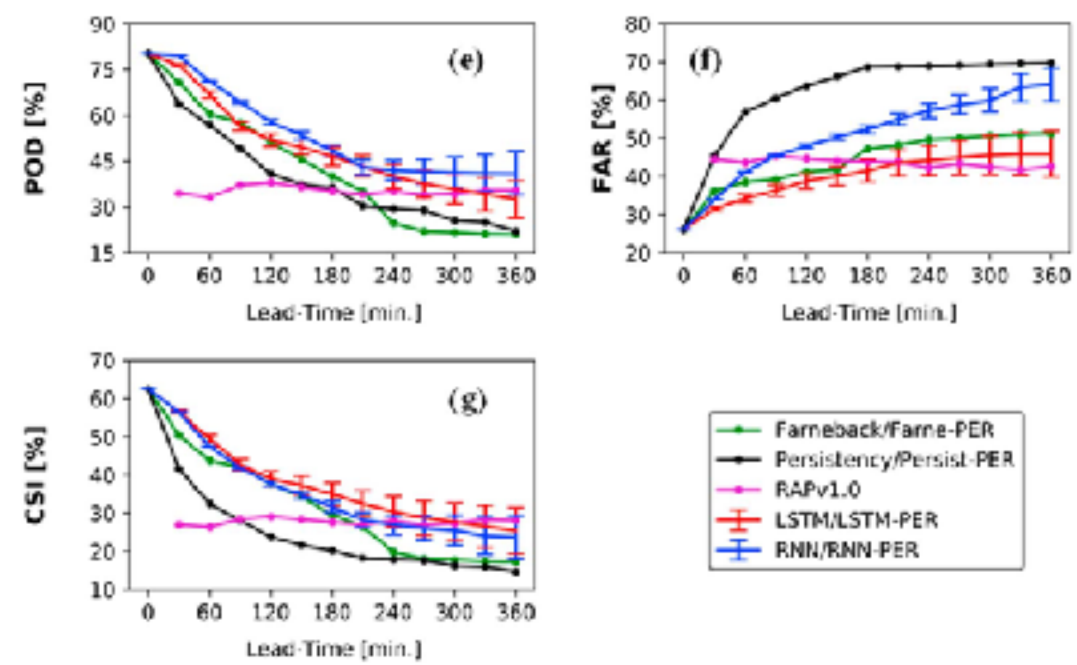
Applications to geoscience

IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, VOL. 13, 2020 2853

A Deep Learning Method With Merged LSTM Neural Networks for SSHA Prediction

Tao Song, *Senior Member, IEEE*, Jingyu Jiang, Wei Li, and Danya Xu





AGU100 ADVANCING EARTH AND SPACE SCIENCE

Journal of Geophysical Research: Atmospheres

RESEARCH ARTICLE

10.1029/2018JD028375

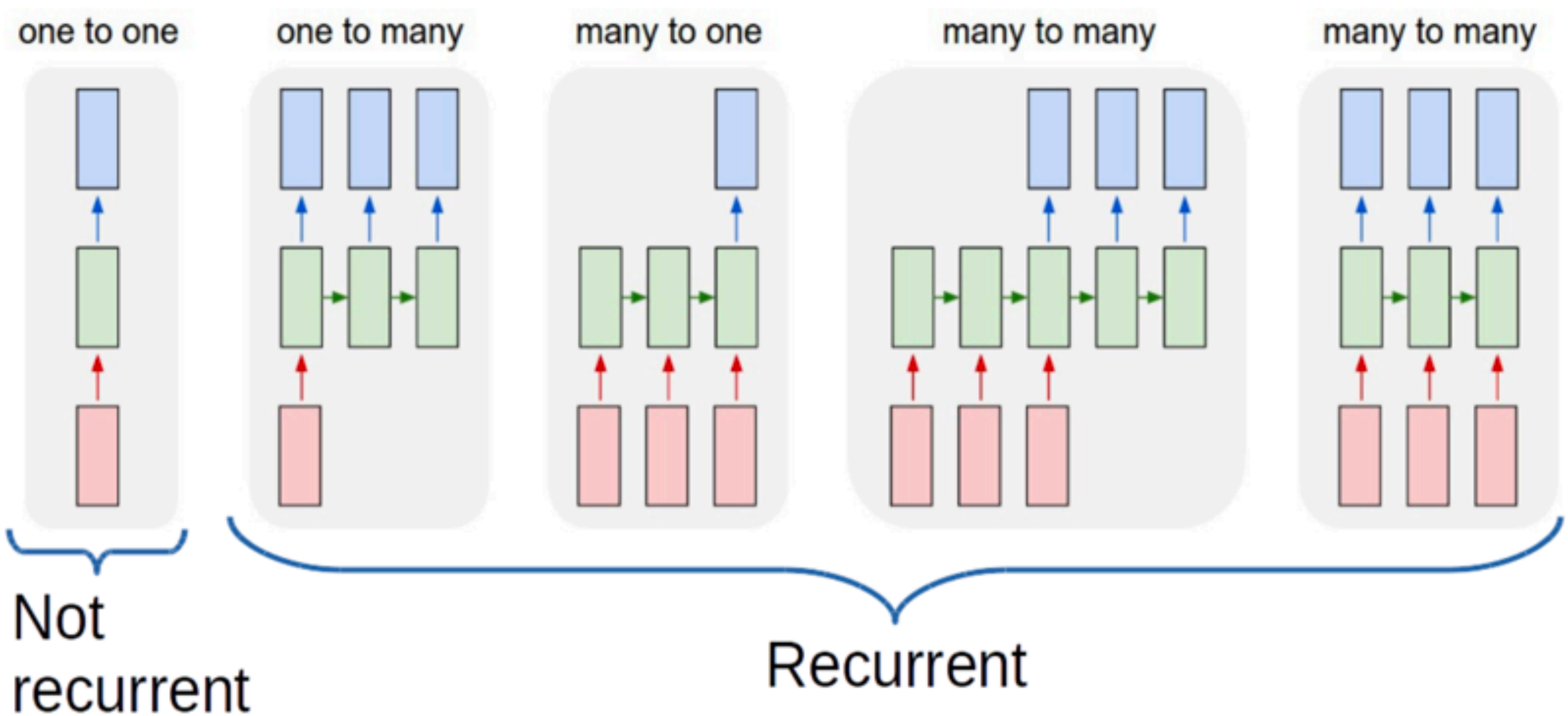
Short-Term Precipitation Forecast Based on the PERSIANN System and LSTM Recurrent Neural Networks

Ata Akbari Asanjan[1], Tiantian Yang[1,2], Kuolin Hsu[1,3], Soroosh Sorooshian[1], Junqiang Lin[4], and Qidong Peng[4]

[1]Department of Civil and Environmental Engineering, Center for Hydrometeorology and Remote Sensing, University of California, Irvine, CA, USA, [2]School of Civil Engineering and Environmental Science, University of Oklahoma, Norman, OK, USA, [3]Center for Excellence for Ocean Engineering, National Taiwan Ocean University, Keelung, Taiwan, [4]China Institute of Water Resources and Hydropower Research, Beijing, China

Key Points:
- Artificial intelligence techniques are useful tools in support of forecasting complex precipitation in short range (0–6 hr)
- Long Short-Term Memory structure is capable of learning spatial and temporal correlations efficiently
- The framework provides accurate

Recurrent Neural Networks



Applications:
- Time-series forecasting
- Audio processing
- Translation
- ....

Do CNNs also apply ?

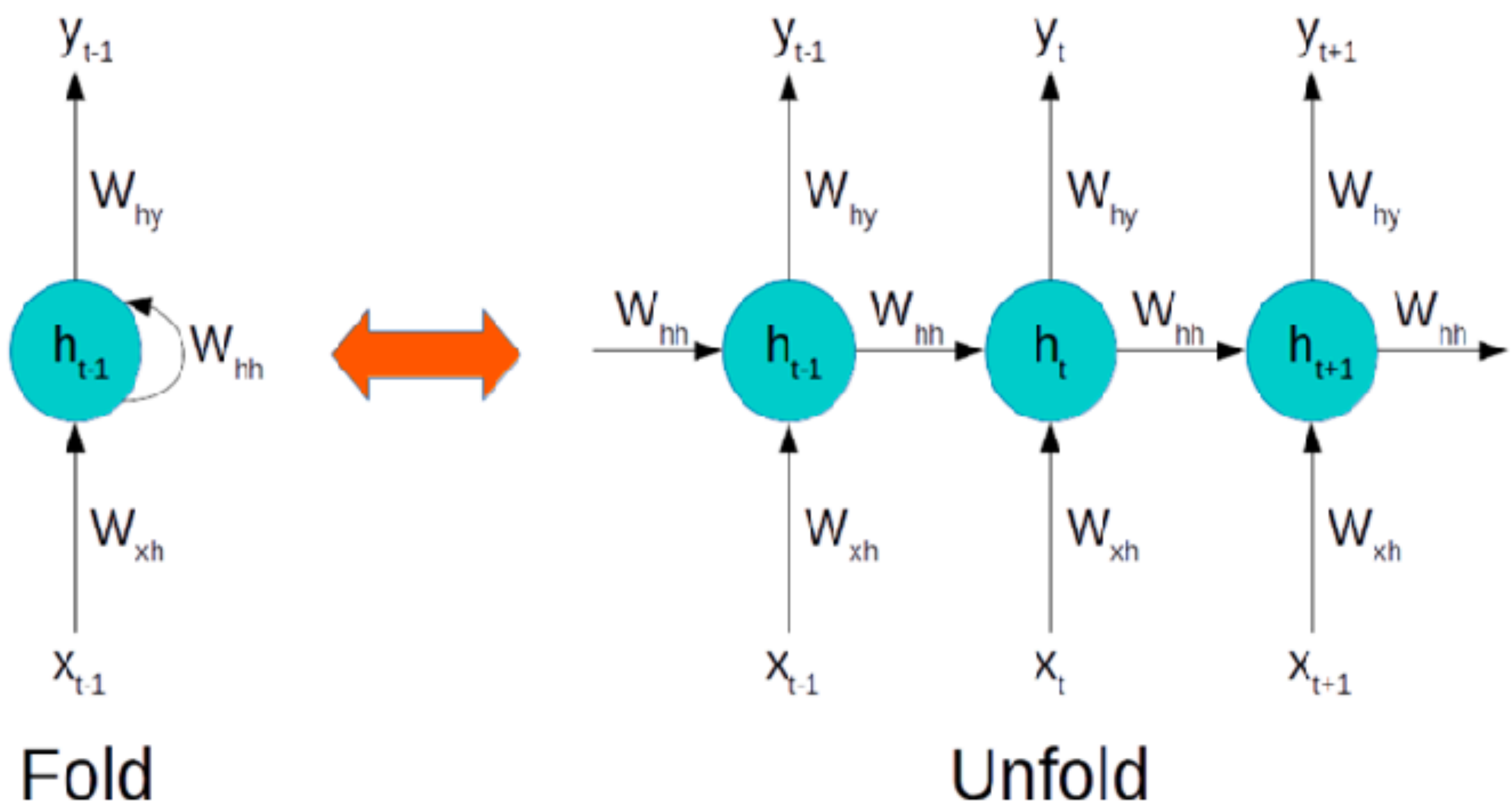
RNN: Underlying formulation (similar to state-space representation)

$x \rightarrow$ input vector

$y \rightarrow$ output vector

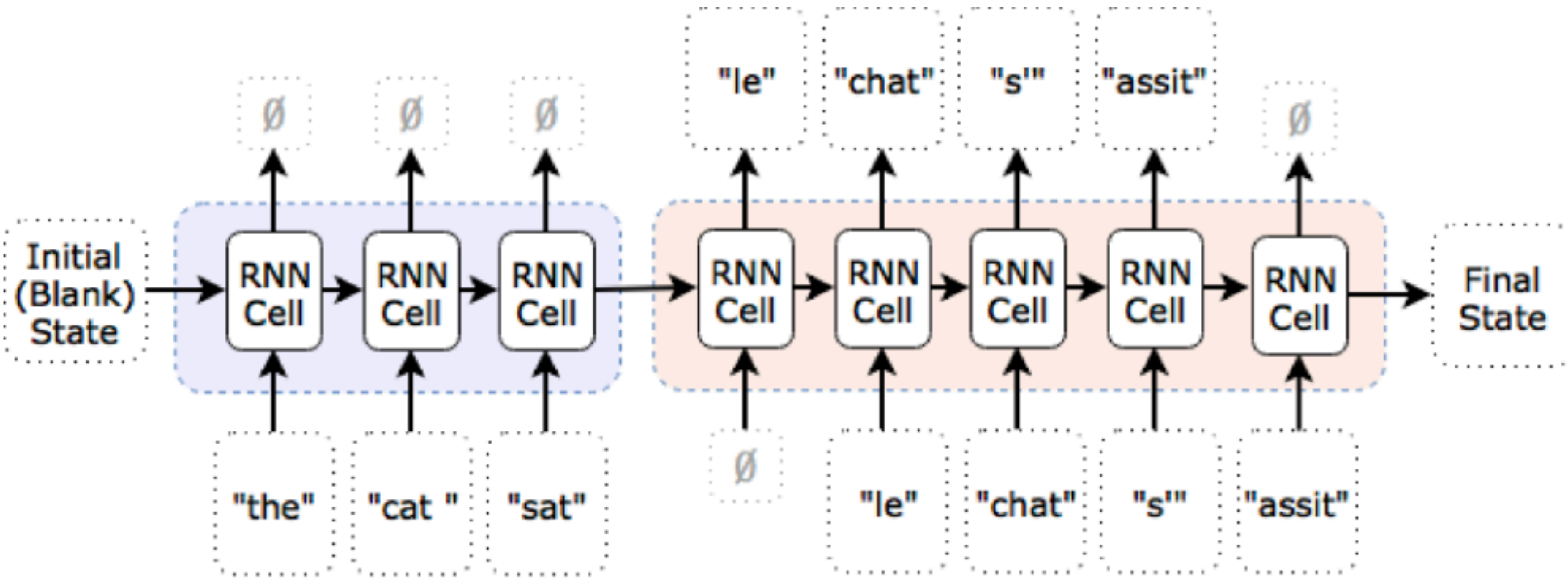
$h \rightarrow$ hidden state

$t \rightarrow$ index (time, index, etc...)



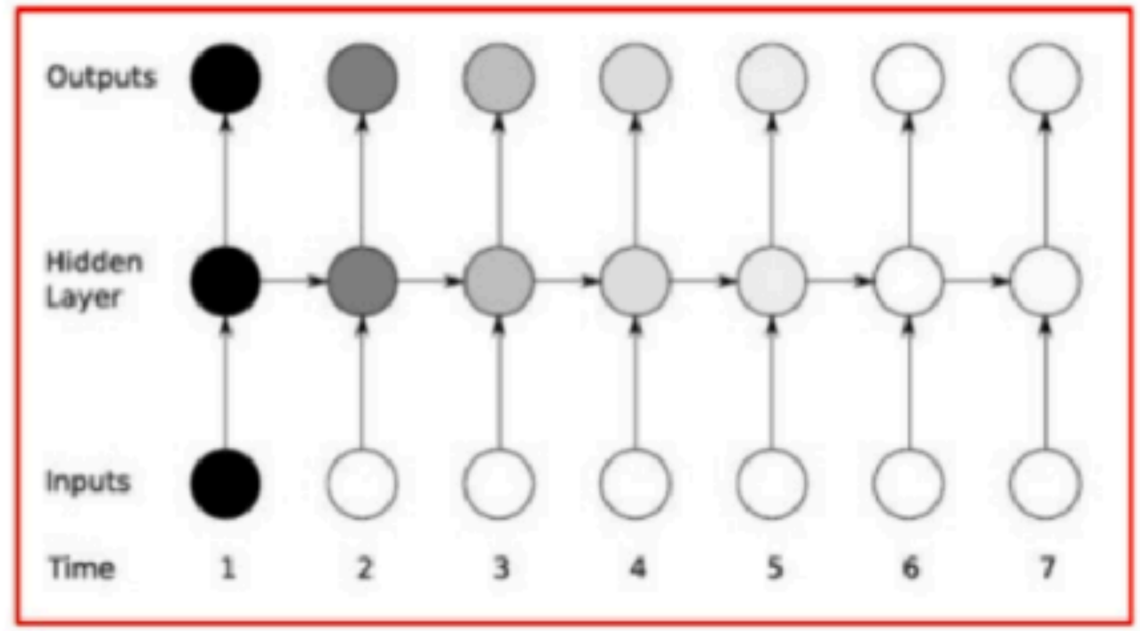
$$h_t = f\left(W_{hh}h_{t-1} + W_{xh}x_t\right) \text{ and } y_t = g\left(W_{hy}h_t\right)$$

RNN: Underlying formulation (similar to state-space representation)

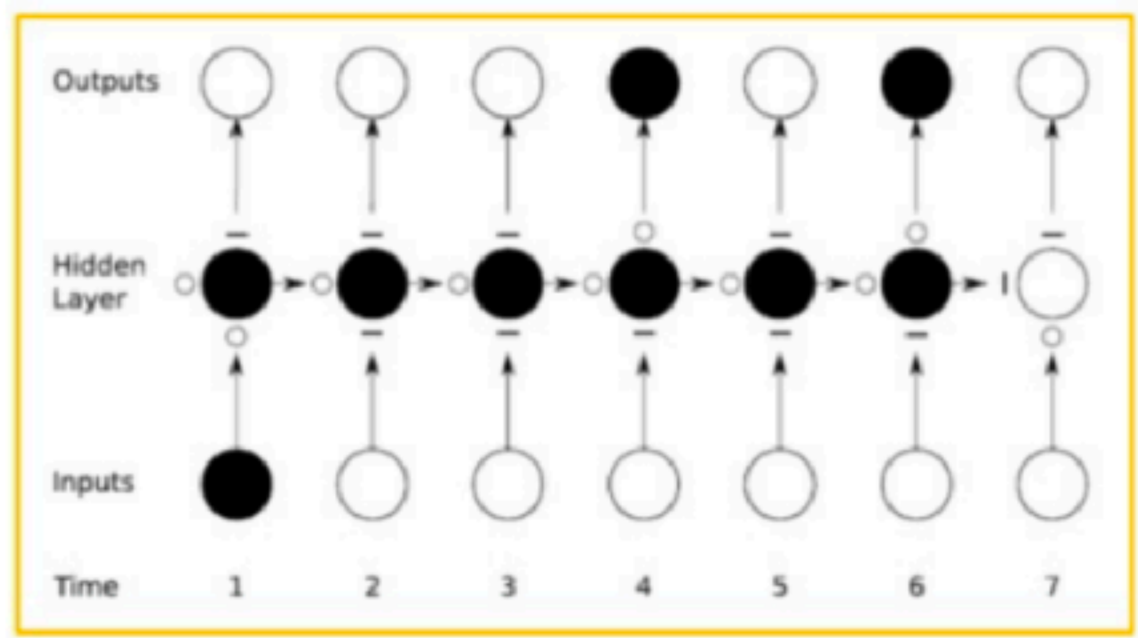


Classic RNN Architectures

- **Conventional RNN with sigmoid**
  - The sensitivity of the input values decays over time
  - The network forgets the previous input



- **Long-Short Term Memory (LSTM)** [2]
  - The cell remember the input as long as it wants
  - The output can be used anytime it wants



Dense and convolutional versions of LSTM and GRU exist
(depending on the structure of the hidden state)

Short-term forecasting application (L63 case-study)

https://github.com/CIA-Oceanix/DLOA2023/blob/main/lectures/notebooks/notebook_PytorchLightning_Forecasting_L63_students.ipynb

Lecture. #3
Things to know (RNN)

- Recurrent Neural Network
- LSTM
- Unfloded and folded representations

Physics-informed/theory-guided networks

General question

How to exploit physical knowledge in the design of neural networks ?

Bridging physics & AI: a broader picture

Representation learning

Physical model

$$\frac{\partial u}{\partial t} + \langle \nabla u, v \rangle = \kappa \Delta u$$

Trainable representation

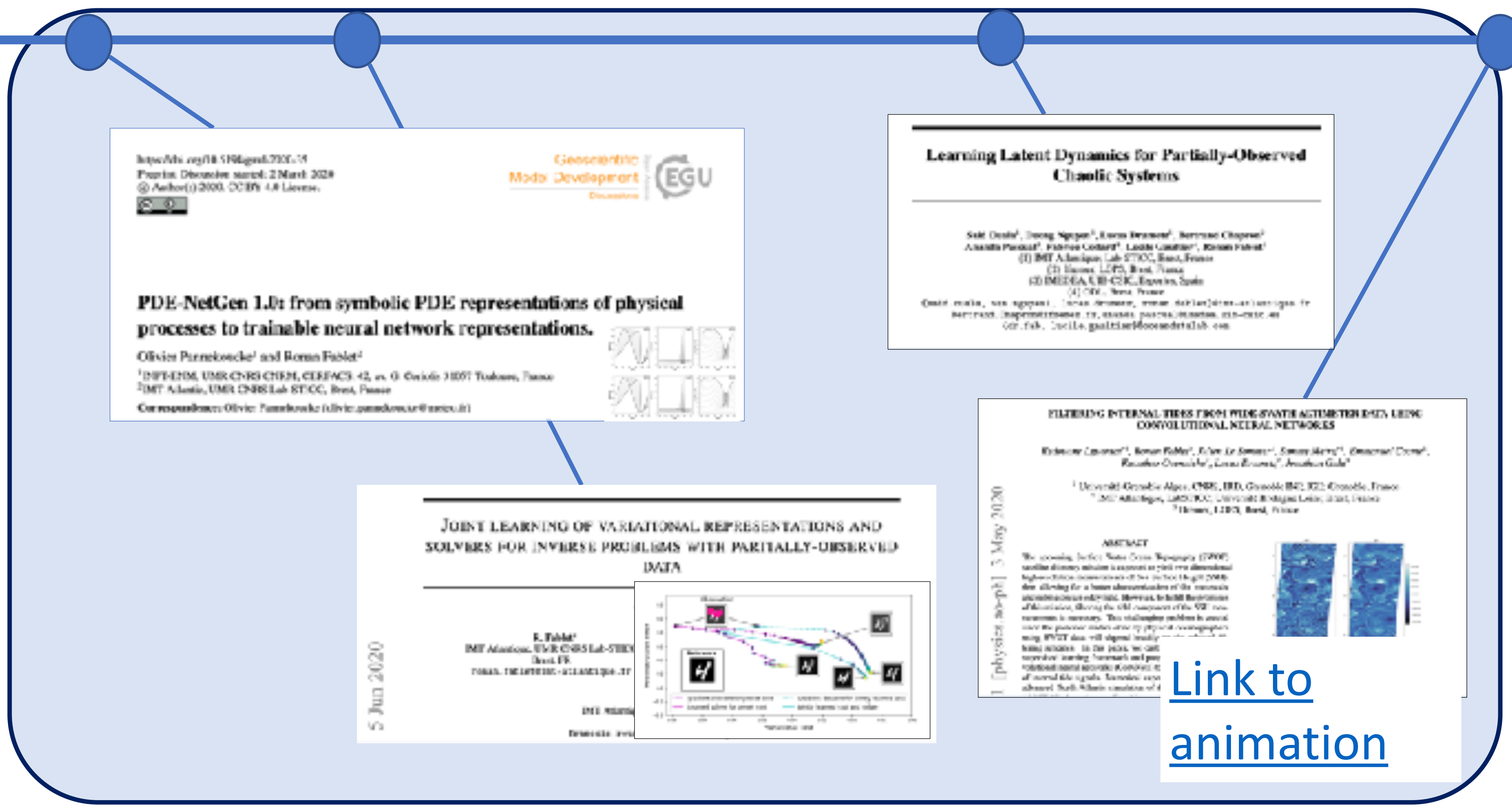
Making the most of AI and Physics Theory

- Model-Driven/Theory-Guided & Data-Constrained schemes
- Data-Driven & Physics-Aware schemes (eg, Ouala et al., 2019)

Bridging Physics & AI: a broader picture

Model-driven

ODEs/PDEs
Variational models
State-space models
Data Assimilation
….



Data-driven

Analog methods
Kernel methods
Neural Networks
…..

Physics-informed & Data-constrained

Data-driven & Physics-aware

Bridging physics & AI: a broader picture

Representation learning

Physical model

$$\frac{\partial u}{\partial t} + \langle \nabla u, v \rangle = \kappa \Delta u$$

Trainable representation

Making the most of AI and Physics Theory

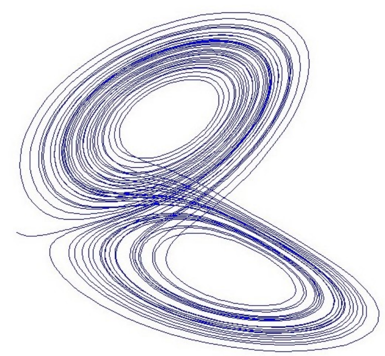
- **Model-Driven/Theory-Guided & Data-Constrained schemes**
- Data-Driven & Physics-Aware schemes (eg, Ouala et al., 2019)

How to embed physics-driven priors in DL models ?

An illustration through L63 dynamics: numerical experiments (Fablet et al., 2018)

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \sigma\left(y(t) - x(t)\right)$$
$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = x(t)\left(\rho - z(t)\right) - y(t)$$
$$\frac{\mathrm{d}z(t)}{\mathrm{d}t} = x(t)\,y(t) - \beta\, z(t)$$

Lorenz-63 equations

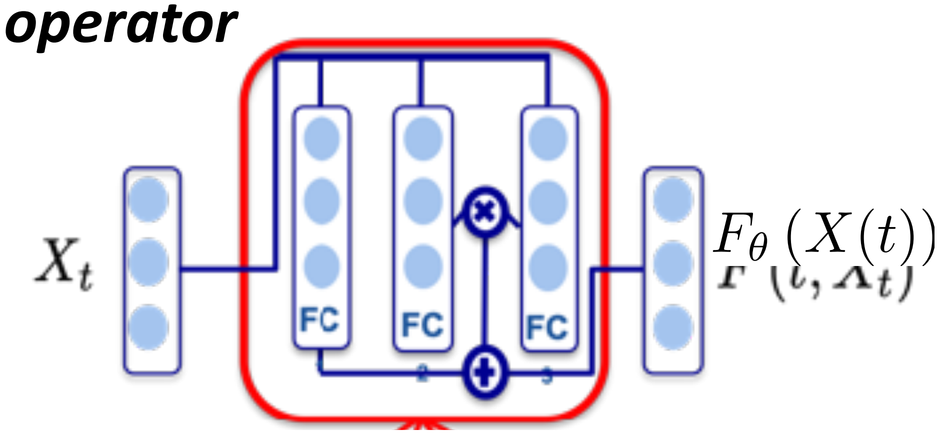


Associated Euler integration scheme

$$d_t X(t) = F_\theta\left(X(t)\right)$$

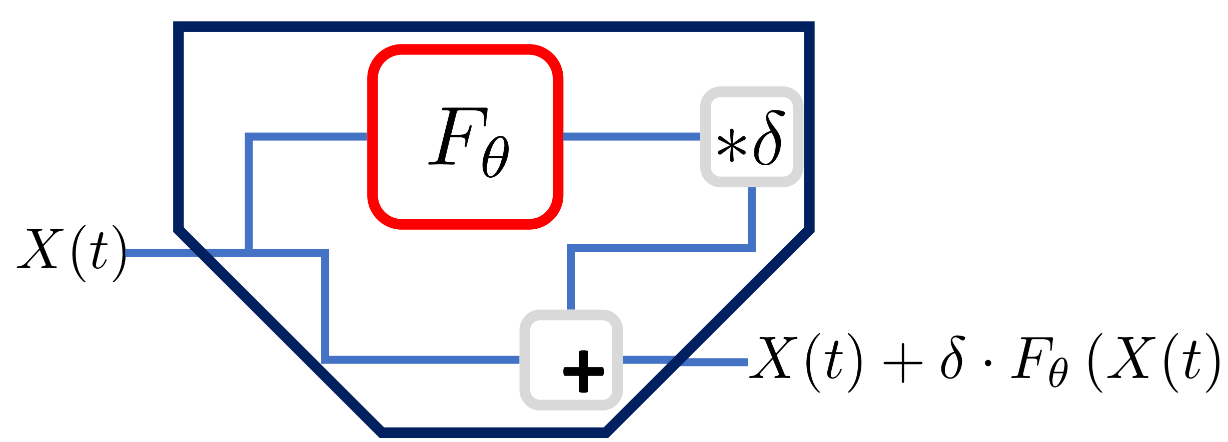
$$X(t+\delta) = X(t) + \delta \cdot F_\theta\left(X(t)\right)$$

NN architecture for differential operator



Bilinear architecture

NN architecture for integration scheme



ResNet architecture (Residual Network)

How to embed physics-driven priors in DL models ?

An illustration through L63 dynamics: numerical experiments (Fablet et al., 2018)

$$\frac{dx(t)}{dt} = \sigma\left(y(t) - x(t)\right)$$
$$\frac{dy(t)}{dt} = x(t)\left(\rho - z(t)\right) - y(t)$$
$$\frac{dz(t)}{dt} = x(t)\,y(t) - \beta\, z(t)$$

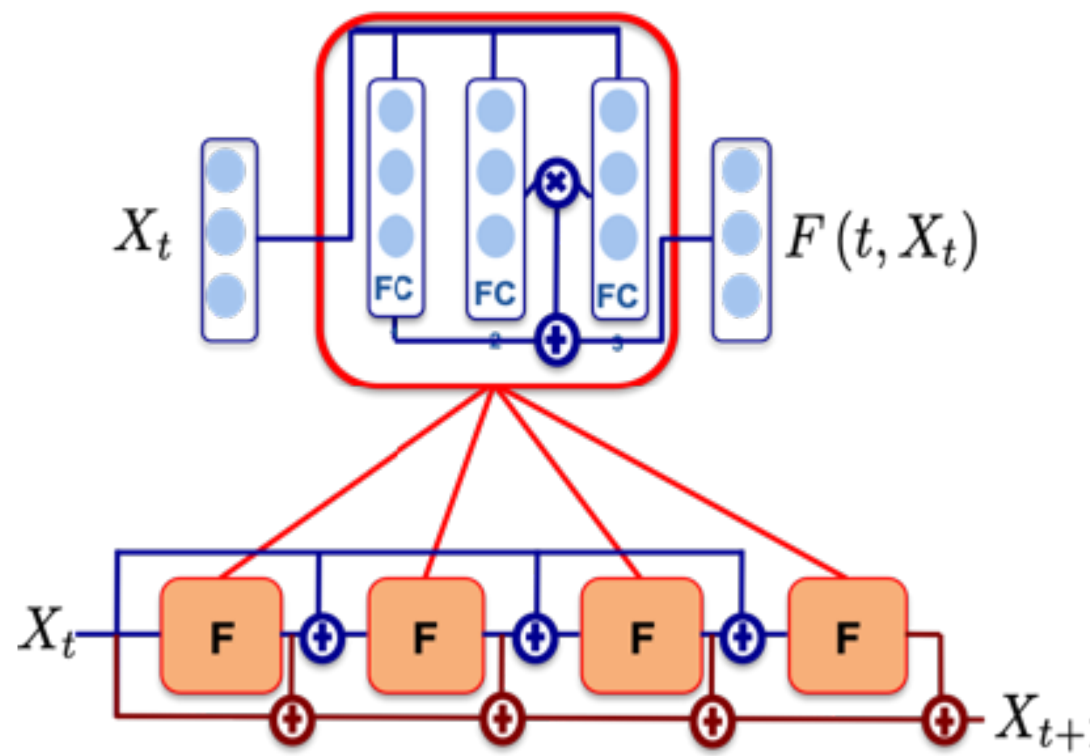
Lorenz-63 equations

Generalization to higher-order integration schemes (eg, RK4)

$$d_t X(t) = F_\theta\left(X(t)\right)$$

$$X(t+\delta) = X(t) + \sum_i \beta_i k_i$$

with $k_i = F_\theta\left(X(t) + \delta\alpha_i k_{i-1}\right)$



NB: Same number of trainable model parameters as the Euler-based architecture

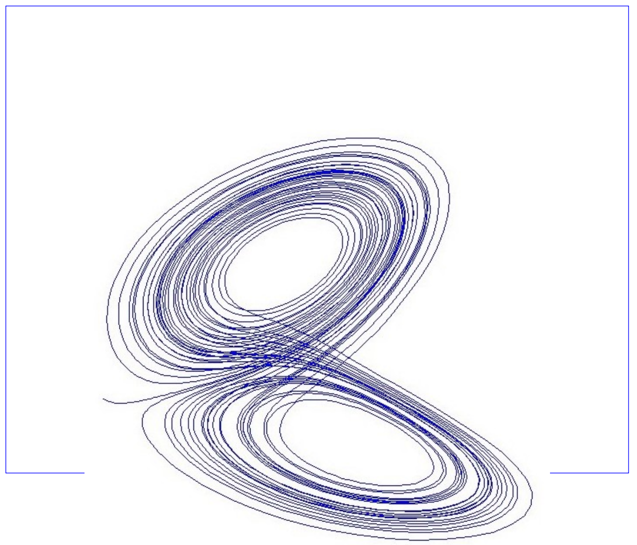
How to embed physics-driven priors in DL models ?

Implementation for L63 dynamics

https://github.com/CIA-Oceanix/DLOA2023/blob/main/lectures/notebooks/notebook_PytorchLightning_Forecasting_L63_students.ipynb

How to embed physics-driven priors in DL models ?

An illustration through L63 dynamics: numerical experiments (Fablet et al., 2018)



Forecasting experiments

Noise-free training data

Forecasting time step	t_0+h	t_0+4h	t_0+8h
Analog forecasting	$<10^{-6}$	0.002	0.005
Sparse regression	$<10^{-6}$	0.002	0.006
MLP	$<10^{-6}$	0.018	0.044
Bi-NN(4)	***$<10^{-6}$***	***$<10^{-6}$***	***$<10^{-6}$***

Noisy training data (σ=0.5)

Forecasting time step	t_0+h	t_0+4h	t_0+8h
Analog forecasting	$<10^{-6}$	2.01	2.2
Bi-NN(4)	***$<10^{-6}$***	***0.054***	***0.14***

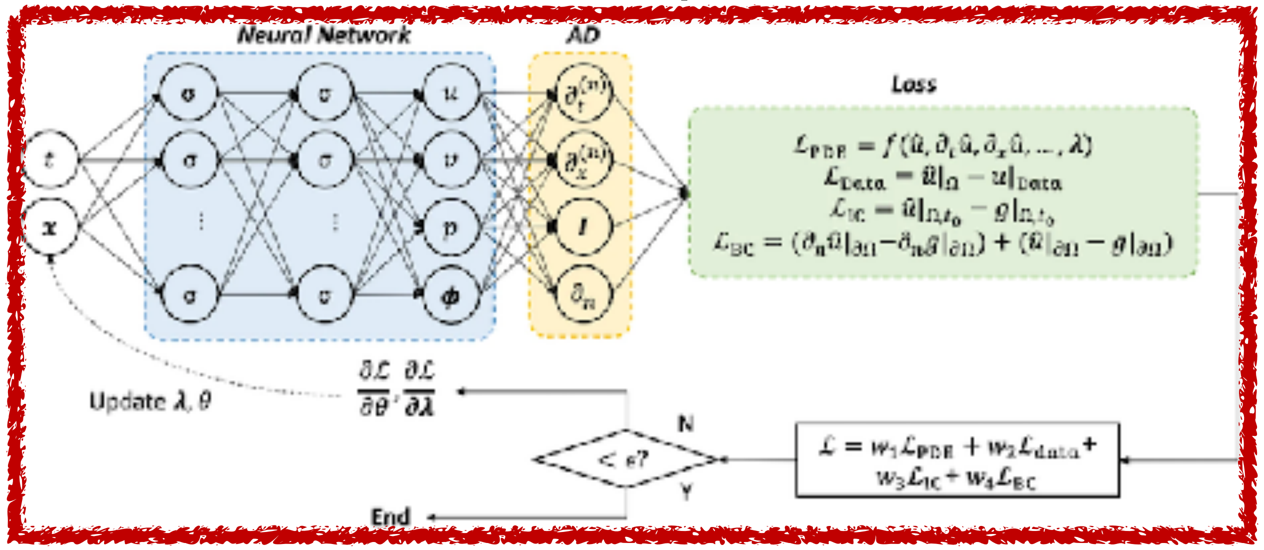
Assimilation experiment

(1 obs. every 8 time steps)

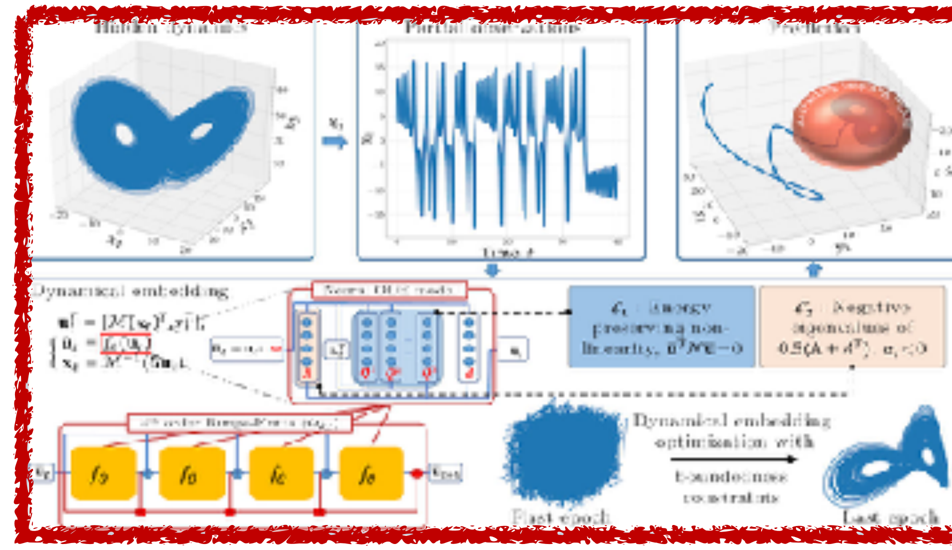
Noise standard deviation in training data	0	0.25	1
True model	0.50	-	-
Analog forecasting	0.65	1.17	1.81
Bi-NN(4)	***0.60***	***0.75***	***0.86***

Physics-informed neural architectures

PINNS for ODEs/PDEs (from Cai et al., 2021)



Identification of ODEs (Ouala et al., 2021)



Other physics-informed representation: Hamiltonian representations [eg, Greydanus et al., 2019], Fourier representations [eg, Li et al., 2021]

Variational data assimilation (Fablet e tal., 2021)

From a Variational DA formulation

$$\widehat{x} = \arg\min_{x} \|y - H(x)\|^2 + \lambda \|x - \Phi(x)\|^2$$

Trainable variational model

Trainable gradient-based solver

Associated end-to-end scheme