Bridging Physics and Learning: application to ocean dynamics

R. Fablet et al.
ronan.fablet@imt-atlantique.fr
web: rfablet.github.io

Webinar IMT Data & AI, July 2020

Lab-STICC
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

General context

Introduction to deep learning and NNs

Bridging physics paradigms and deep learning

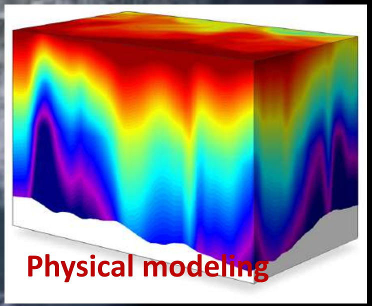
Beyond Ocean Dynamics

General question

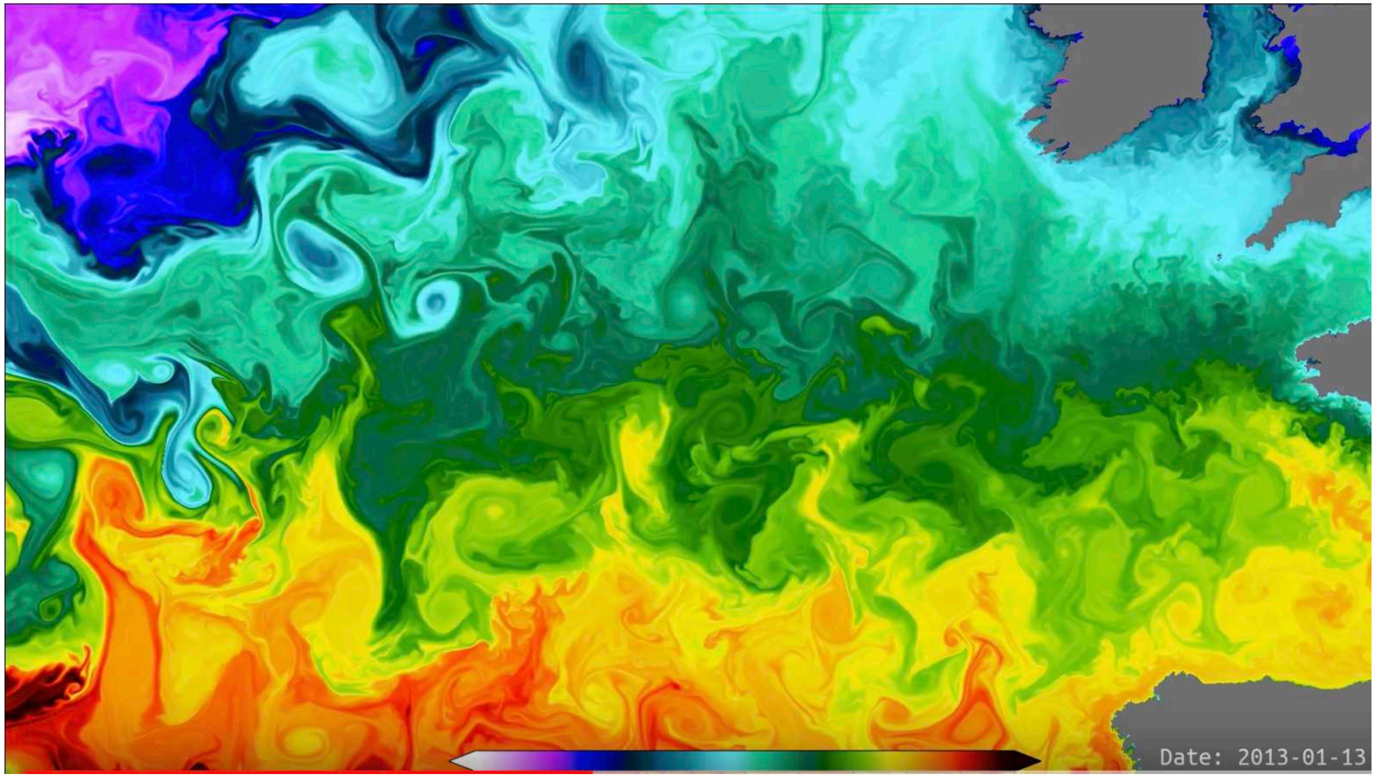
How to solve sampling gaps and extract high-level information for ocean monitoring and surveillance ?

+

+



Context: No observation / simulation system to resolve all scales and processes simultaneously



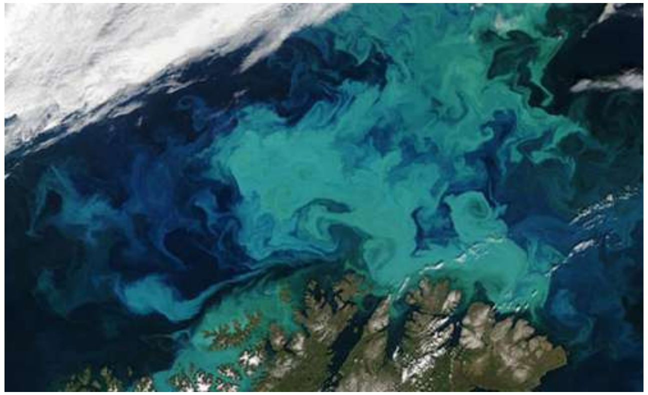
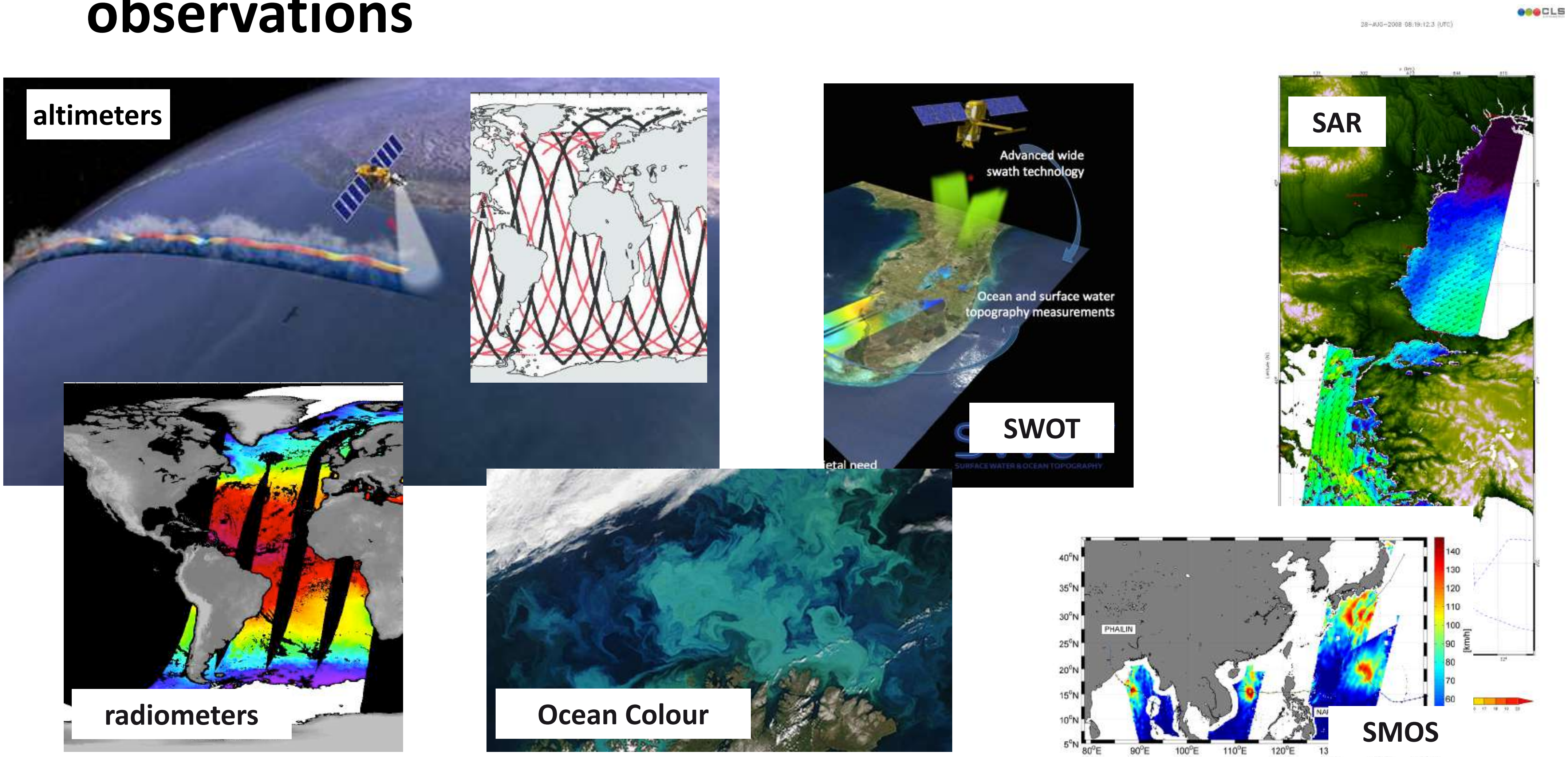
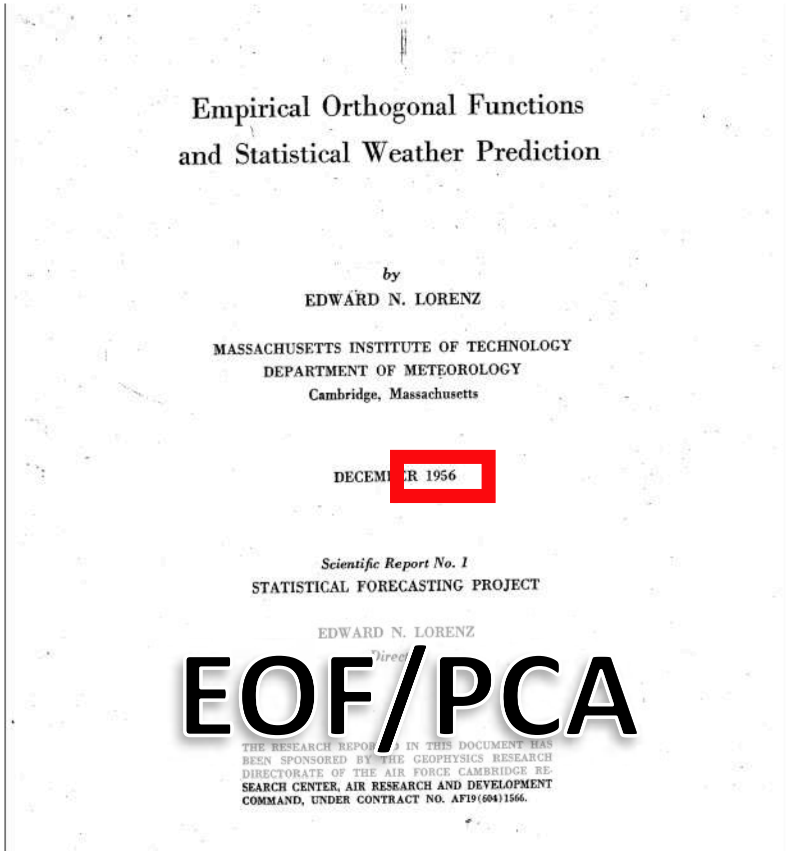


Illustration of satellite-derived sea surface observations



What about deep learning to solve sampling gaps and infer higher-level information ?

Learning & Geoscience: an old story ?



Empirical Orthogonal Functions and Statistical Weather Prediction

by

EDWARD N. LORENZ

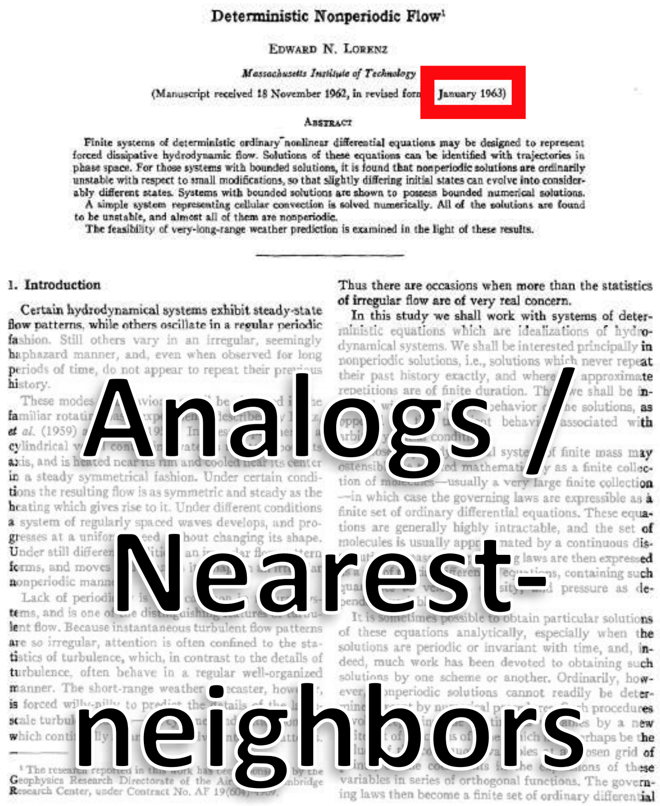
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF METEOROLOGY
Cambridge, Massachusetts

DECEMBER 1956

Scientific Report No. 1
STATISTICAL FORECASTING PROJECT

EDWARD N. LORENZ

THE RESEARCH REPORTED IN THIS DOCUMENT HAS BEEN SPONSORED BY THE GEOPHYSICS RESEARCH DIRECTORATE OF THE AIR FORCE CAMBRIDGE RESEARCH CENTER, AIR RESEARCH AND DEVELOPMENT COMMAND, UNDER CONTRACT NO. AF19(604)1566.



Deterministic Nonperiodic Flow[1]

EDWARD N. LORENZ

Massachusetts Institute of Technology

(Manuscript received 18 November 1962, in revised form January 1963)

ABSTRACT

Finite systems of deterministic ordinary nonlinear differential equations may be designed to represent forced dissipative hydrodynamic flow. Solutions of these equations can be identified with trajectories in phase space. For those systems with bounded solutions, it is found that nonperiodic solutions are ordinarily unstable with respect to small modifications, so that slightly differing initial states can evolve into considerably different states. Systems with bounded solutions are shown to possess bounded numerical solutions.

A simple system representing cellular convection is solved numerically. All of the solutions are found to be unstable, and almost all of them are nonperiodic.

The feasibility of very-long-range weather prediction is examined in the light of these results.

1. Introduction

Learning & Geoscience: Data-driven approaches for data assimilation


October 2017 LGUENSAT ET AL. 4093

The Analog Data Assimilation

Redouane Lguensat and Pierre Tandeo
IMT Atlantique, Lab-STICC, Université Bretagne Loire, Brest, France

Pierre Ailliot
Laboratoire de Mathématiques de Bretagne Atlantique, University of Western Brittany, Brest, France

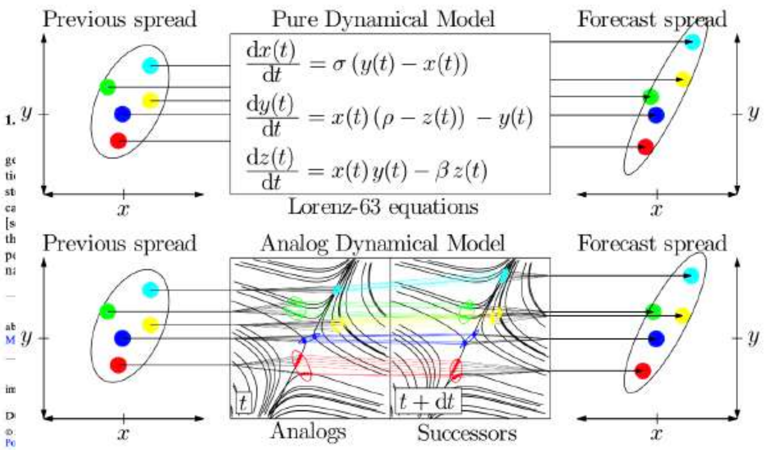
Manuel Pulido
Department of Physics, Universidad Nacional del Nordeste, and CONICET, Corrientes, Argentina

Ronan Fablet
IMT Atlantique, Lab-STICC, Université Bretagne Loire, Brest, France

(Manuscript received 23 November 2016, in final form 31 July 2017)

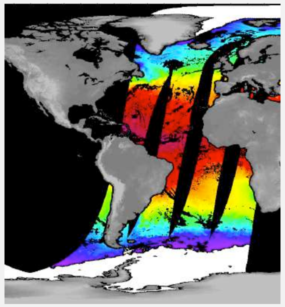
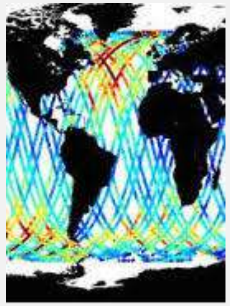
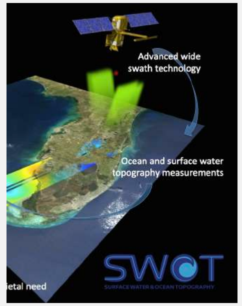
ABSTRACT

In light of growing interest in data-driven methods for oceanic, atmospheric, and climate sciences, this work focuses on the field of data assimilation and presents the analog data assimilation (AnDA). The proposed framework produces a reconstruction of the system dynamics in a fully data-driven manner where no explicit knowledge of the dynamical model is required. Instead, a representative catalog of trajectories of the system is assumed to be available. Based on this catalog, the analog data assimilation combines the nonparametric



The analog data assimilation [Lguensat et al., 2017]

- Combination of analog forecasting strategies and EnKF assimilation schemes
- Extension to 2D+t geophysical dynamics

Open questions

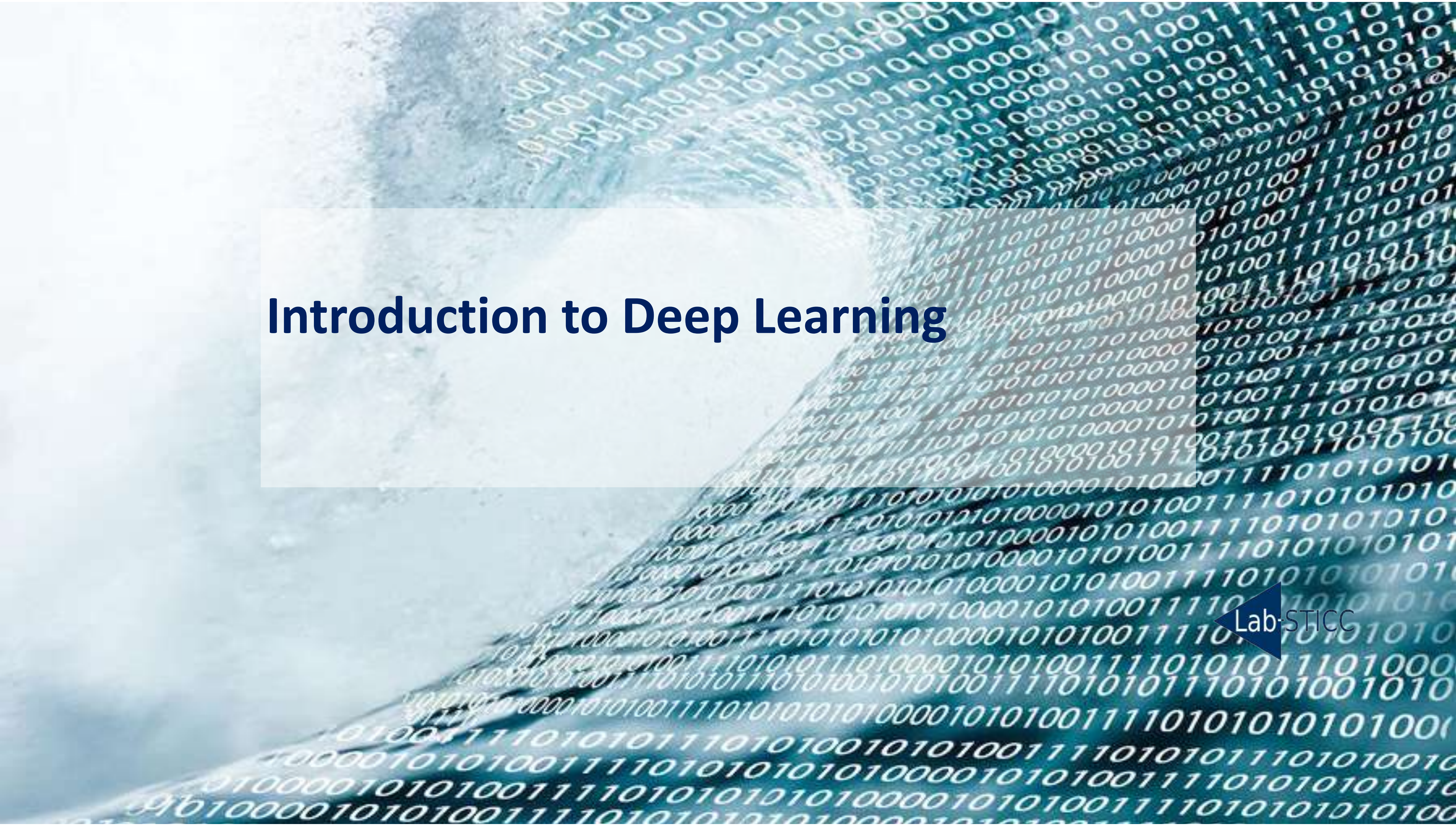
- Bridging model-driven and data-driven paradigms
- Learning data-driven representations from real observation data

General context

Introduction to deep learning and NNs

Bridging physics paradigms and deep learning

Beyond Ocean Dynamics



Introduction to Deep Learning

Resources

- **Book: Deep Learning**

Goodfellow, Bengio, Courville, MIT Press

Online version http://www.deeplearningbook.org/

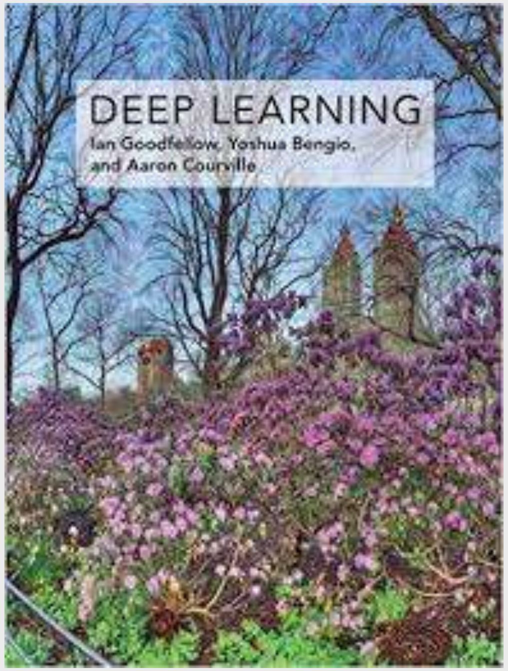
- **Online course by Andrew Ng (Stanford/Baidu)**

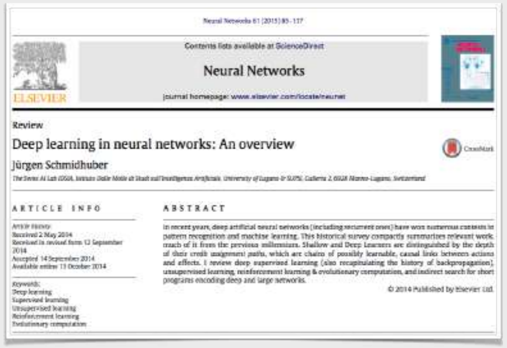
  Youtube: [link]

  Online course on Coursera: [link]

- **Review paper: Deep learning in neural networks by J. Schmidhuber**

  pdf: [link]





Neural Networks 61 (2015) 85–117

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Review

Deep learning in neural networks: An overview

Jürgen Schmidhuber

ARTICLE INFO

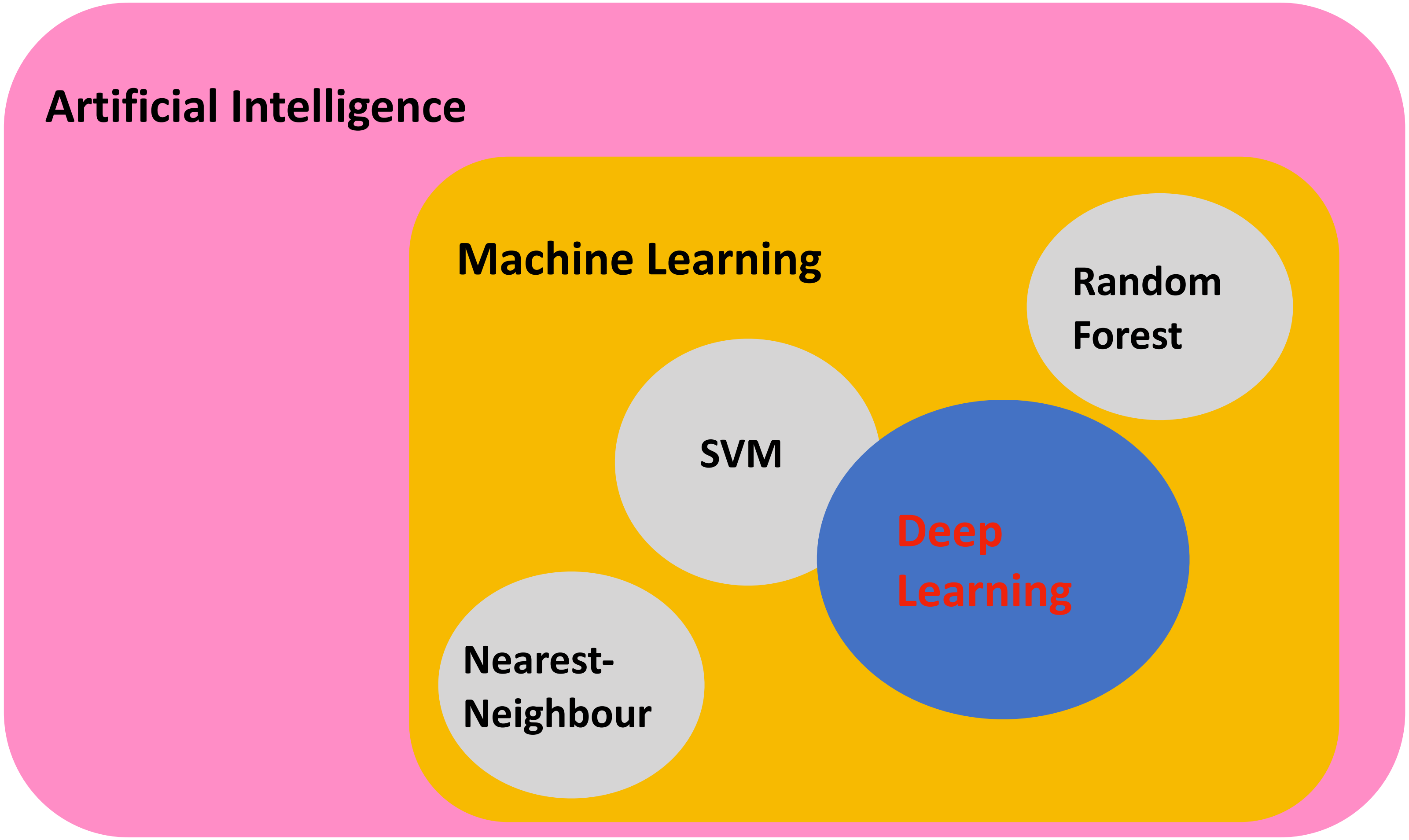
Article history:
Received 2 May 2014
Received in revised form 12 September 2014
Accepted 14 September 2014
Available online 13 October 2014

Keywords:
Deep learning
Supervised learning
Unsupervised learning
Reinforcement learning
Evolutionary computation

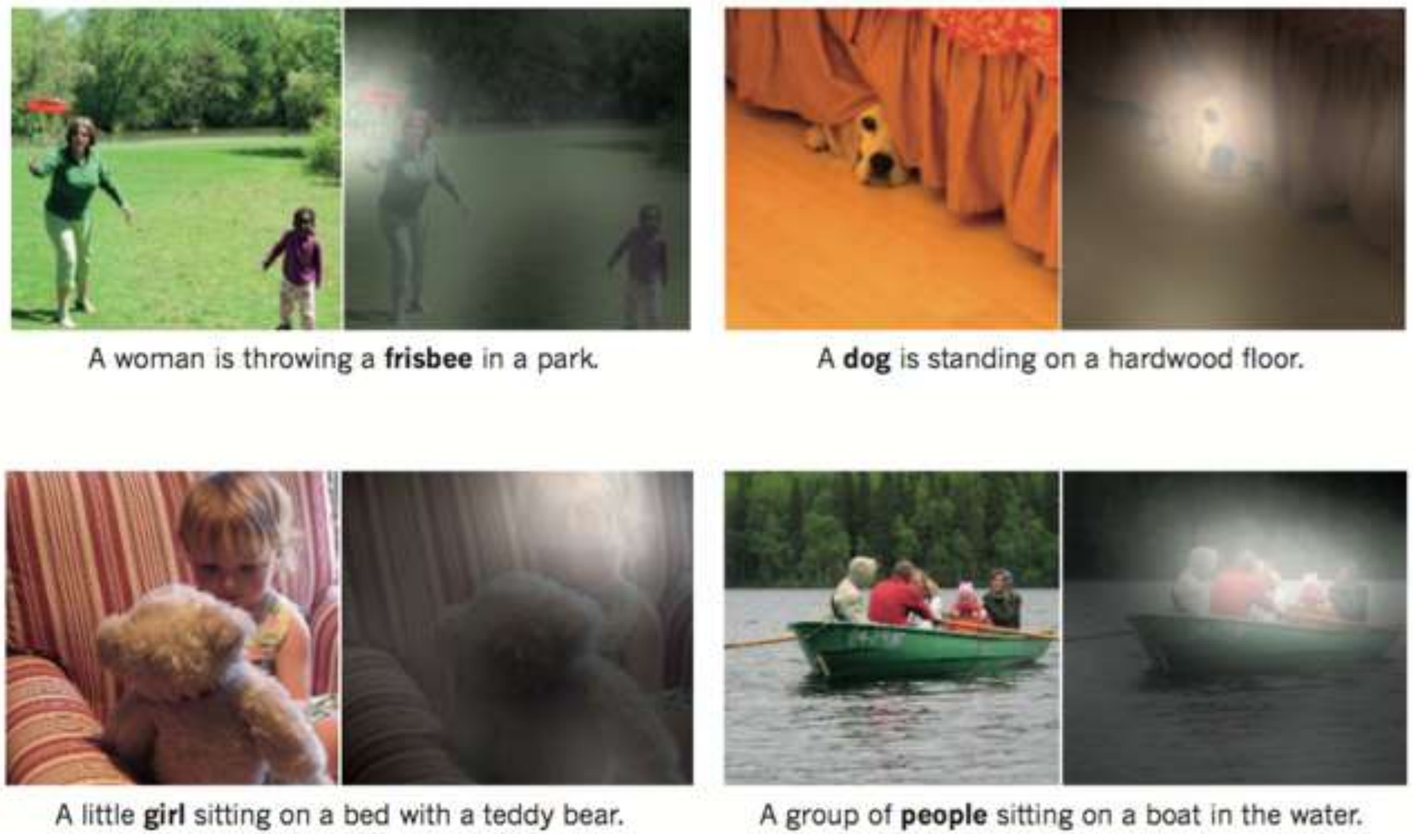
ABSTRACT

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous millennium. Shallow and Deep Learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

What's Deep Learning?

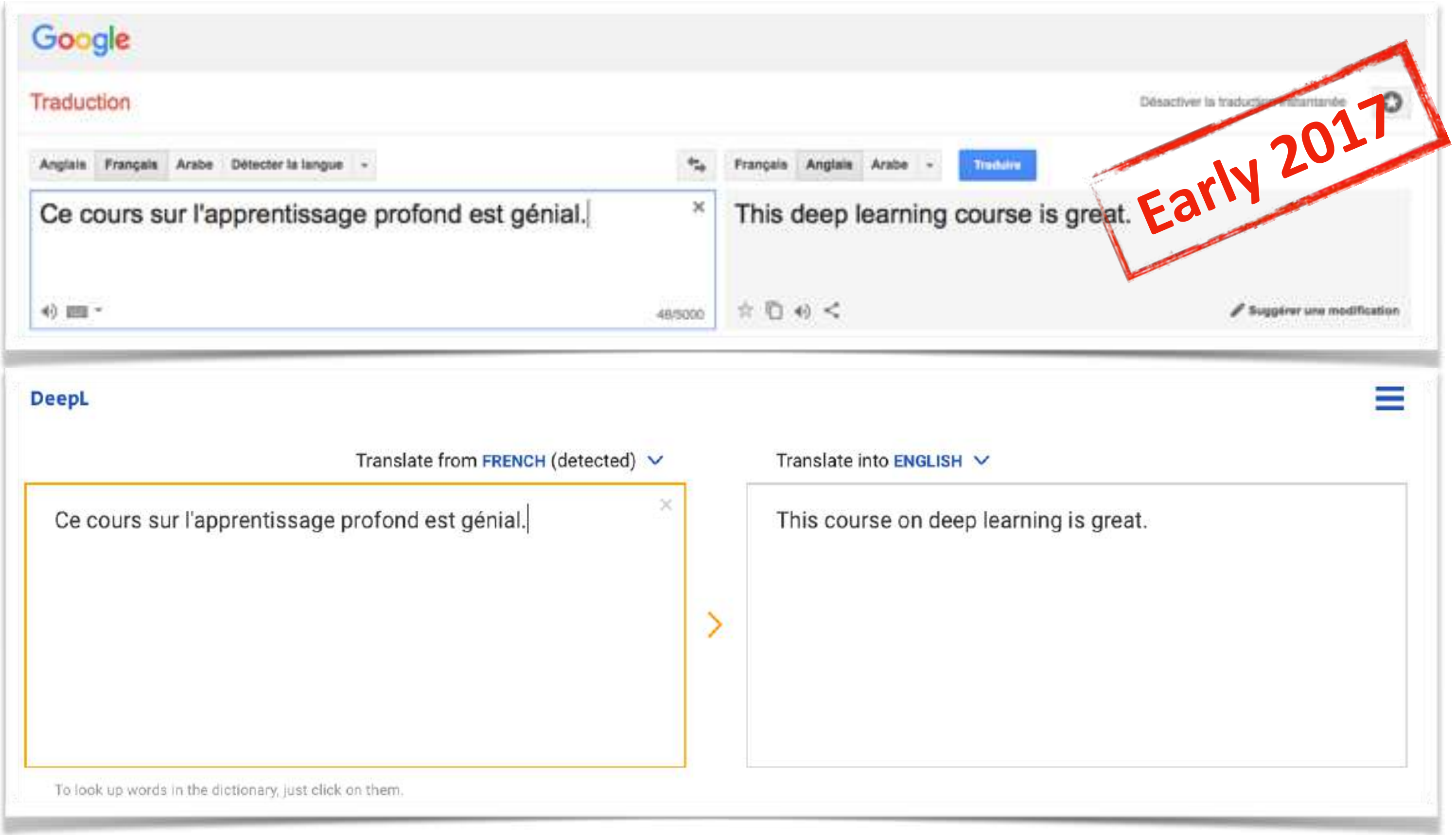


From Image to Text
(image captioning)



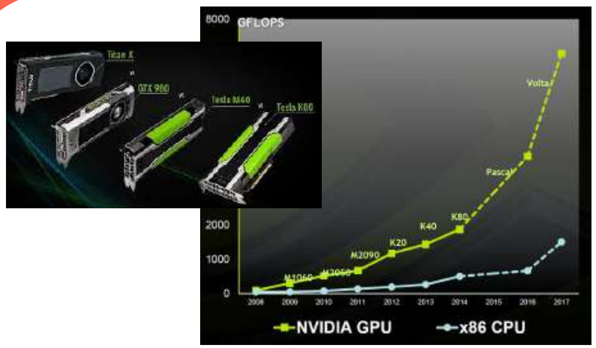
Vinyals et al., ICML 2014

Automatic Translation



and many more…….

Key reasons for the emergence of DL



High-performance computing (GPU)

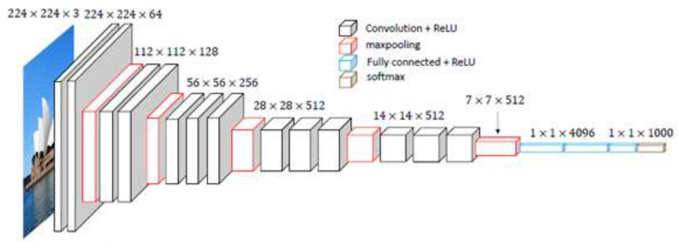
Large annotated dataset (> 1M)

TensorFlow

Keras

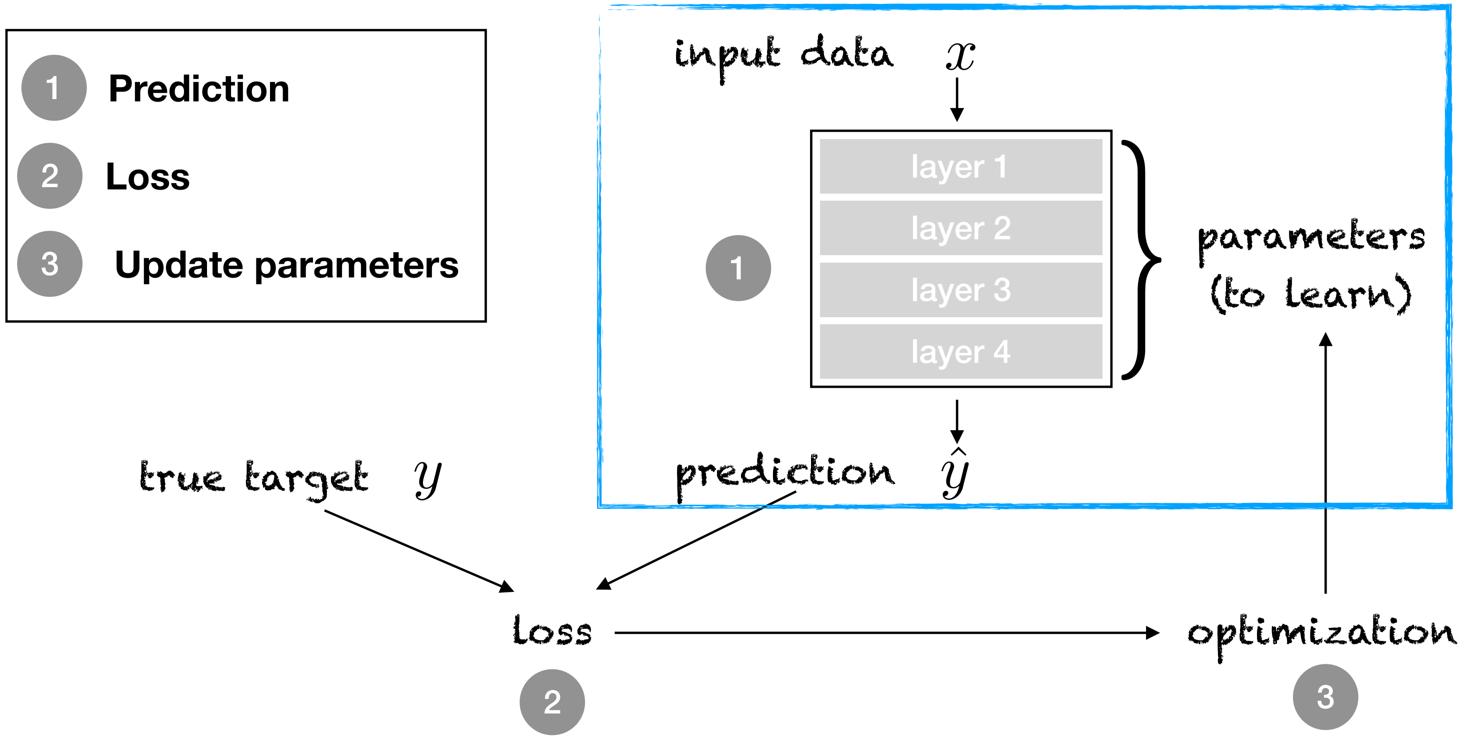
PYTORCH

Efficient & easy-to-use frameworks



End-to-end learning

What's learning (for a computer) ?



What’s learning (for a computer) ?

Mathematical formulation: Learning comes to minimising some loss function given w.r.t. model parameters and training data

$$\widehat{\theta} = \arg\min_{\theta} \mathcal{L}\left(\{x_i, y_i\}_{i\in\{1,..,N\}}; f_\theta\right)$$

Key questions:

- Which parameterisation for model f ?
- Which loss function ?

Deep learning models

Neural networks: composition idea

- **Approximation through the composition of (simple) elementary functions:**

$$f_\theta(x) = f_{\theta_N} \circ \ldots \circ f_{\theta_2} \circ f_{\theta_1}(x)$$

- **Key features:**
  - Any continuous function can be approximated as the composition of elementary functions
  - Analytical/exact computation of the derivative of f with respect to parameters and input variables
  - Direct exploitation of gradient-based optimisation schemes for learning

Neural networks: automatic differentiation

- **Given the general composition idea:**

$$f_\theta(x) = f_{\theta_N} \circ \ldots \circ f_{\theta_2} \circ f_{\theta_1}(x)$$

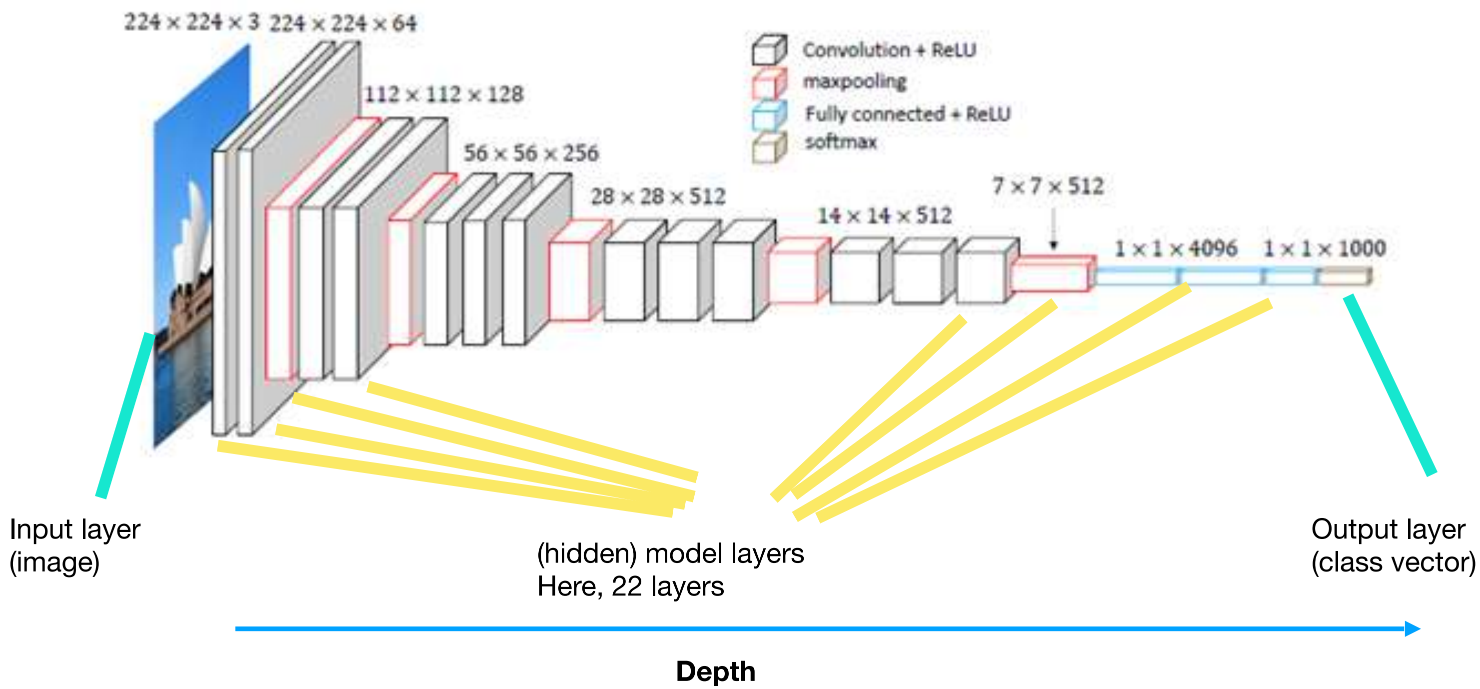
- **NNs can implement automatic differentiation knowing the symbolic differentiation for elementary functions:**
  - AD example: https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html
  - Basis of the backpropagation algorithm (similar to adjoint method)

- **Resulting gradient descent for learning model parameters:**

$$\widehat{\theta} = \arg\min_\theta \mathcal{L}\left(\{x_i, y_i\}_i; f_\theta\right) \Longrightarrow \theta^{(k+1)} = \theta^{(k+1)} + \lambda_k \nabla_\theta \mathcal{L}\left(\{x_i, y_i\}_i; f_\theta\right)$$

Deep learning models

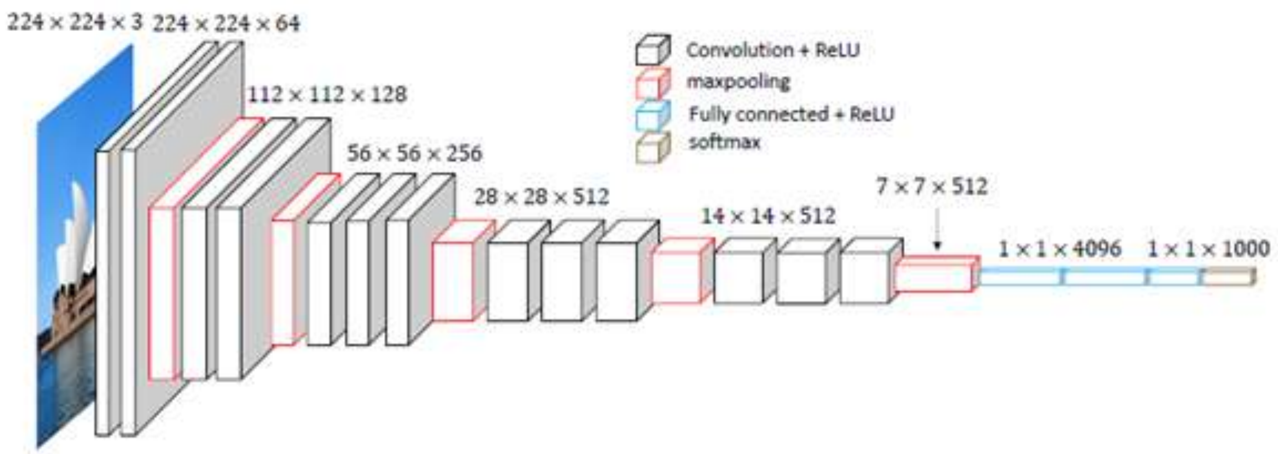
DL models are (in general) feedforward models. VGG16 as an illustration



The more layers, the deeper…… Some models may have up to several hundreds to thousands of layers.

Basics of DL models

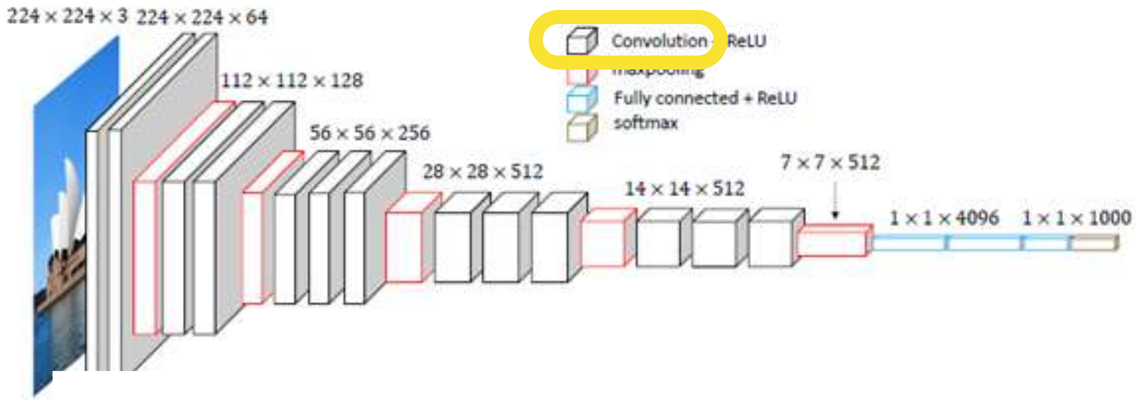
DL models are (in general) feedforward models. VGG16 as an illustration



Elementary components

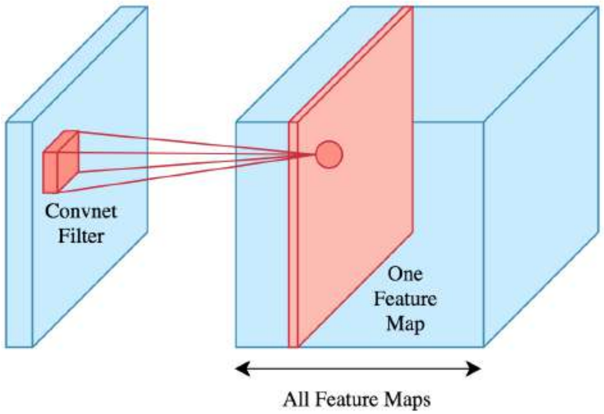
- Convolution layers
- Activation layers
- Pooling layers
- Dense layers

Basics of DL models



Elementary components

- Convolution layers
- Activation layers
- Pooling layers
- Dense layers

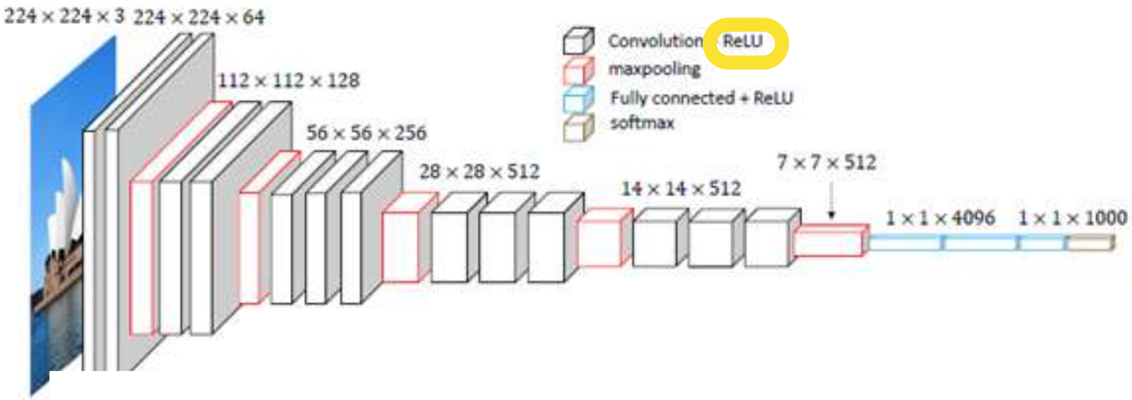


Number of parameters:
Filter size x Number of filters

e.g. 3x3xKxNfilt

Independent on the sizes of the input and output layer

Basics of DL models



Elementary components

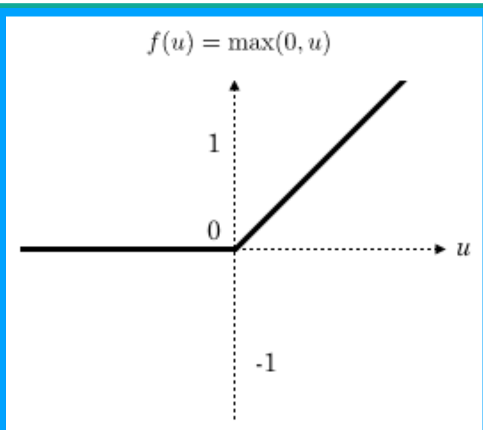
Convolution layers

Activation layers

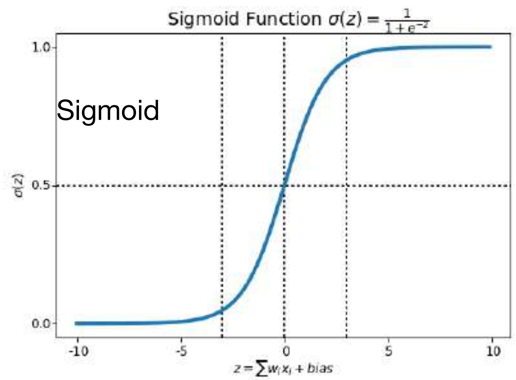
Pooling layers

Dense layers

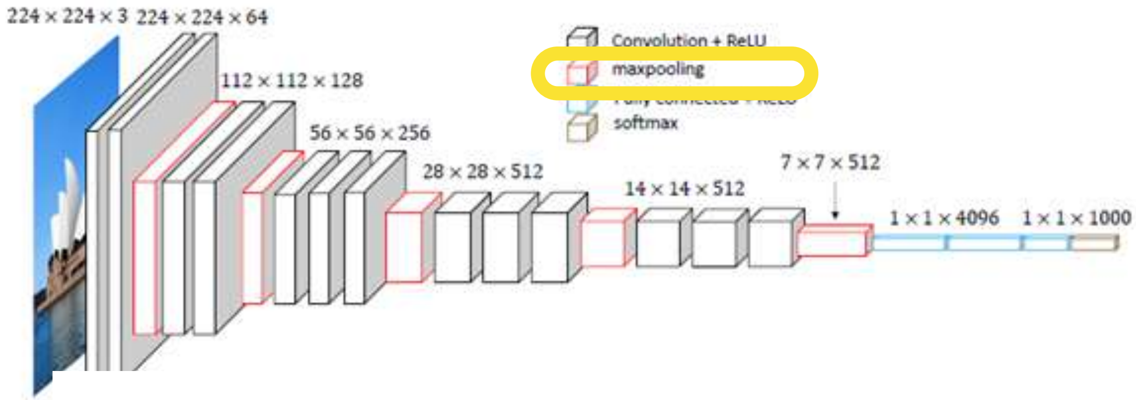
ReLU
(Rectified Linear Unit)



Sigmoid



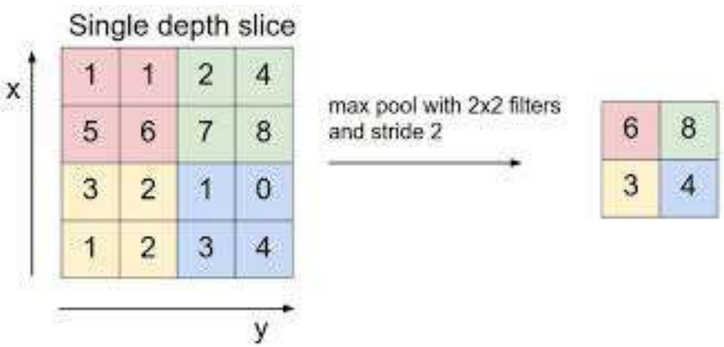
Basics of DL models



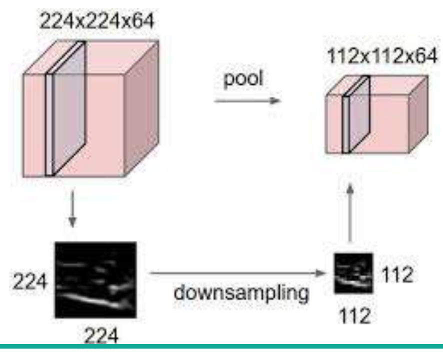
Elementary components

- Convolution layers
- Activation layers
- Pooling layers
- Dense layers

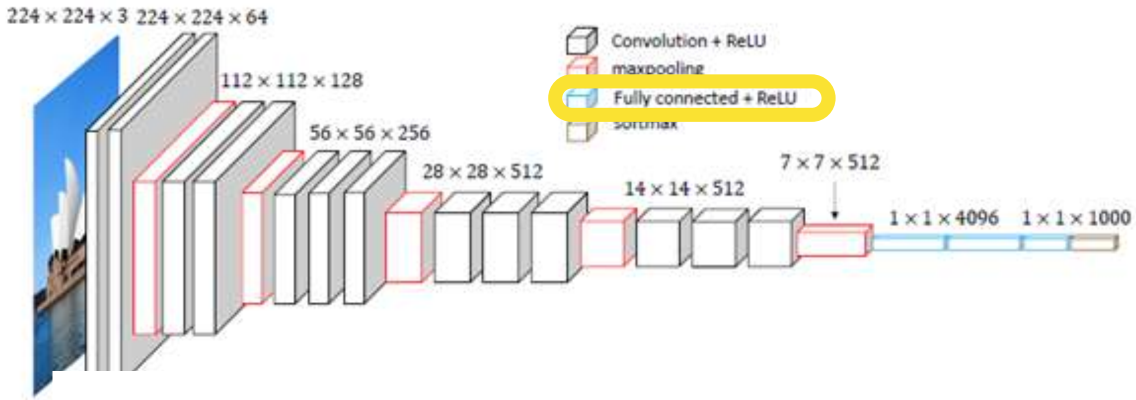
An example of max-pooling operator



Pooling downsamples the input layer



Basics of DL models

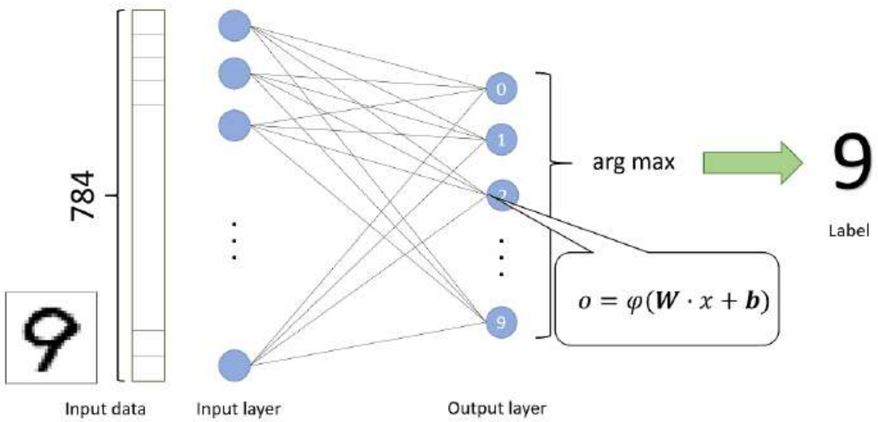


Elementary components

Convolution layers | Activation layers | Pooling layers | Dense layers

Dense layers
or
Fully-connected (FC) layer

as in a classic MLP



Let's play with MLPs using tensorflow playground

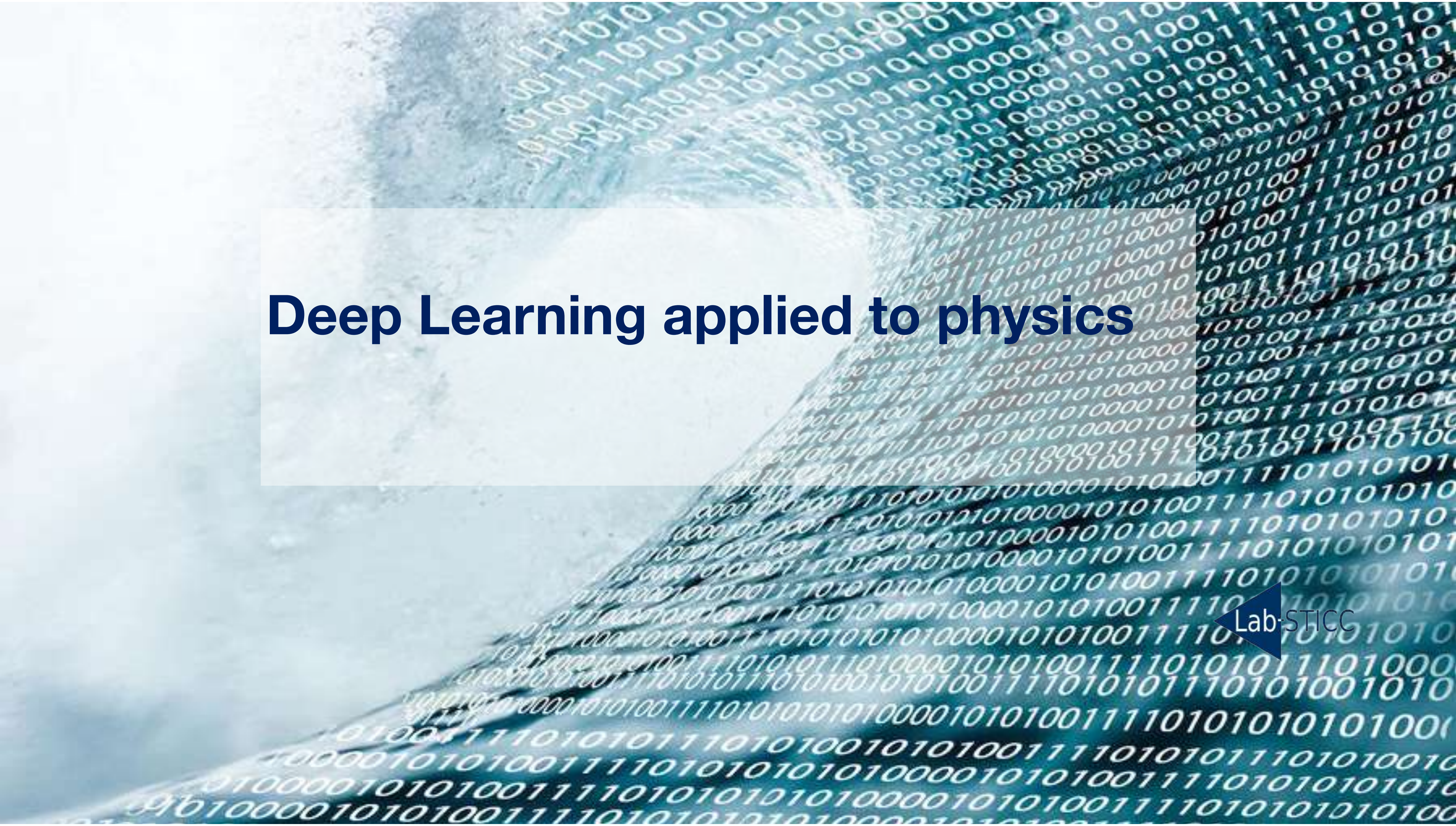
http://playground.tensorflow.org/

General context

Introduction to deep learning and NNs

Bridging physics paradigms and deep learning

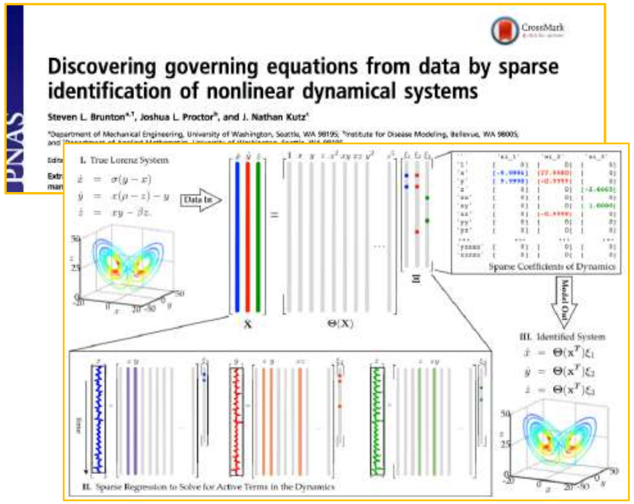
Beyond Ocean Dynamics

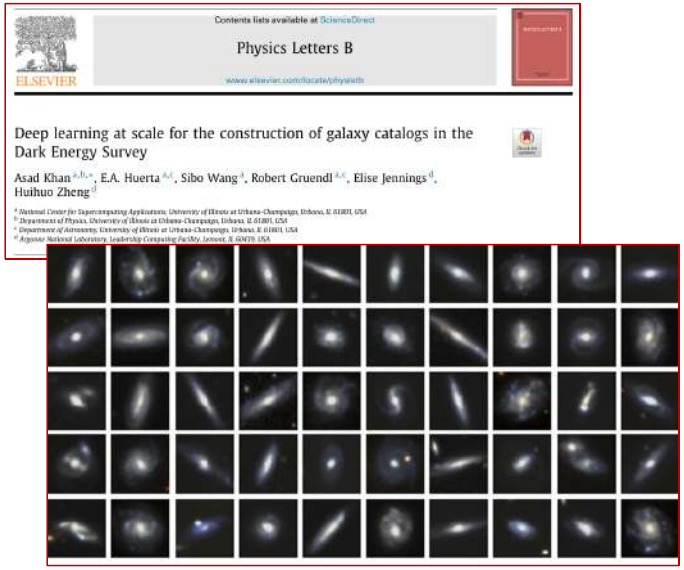


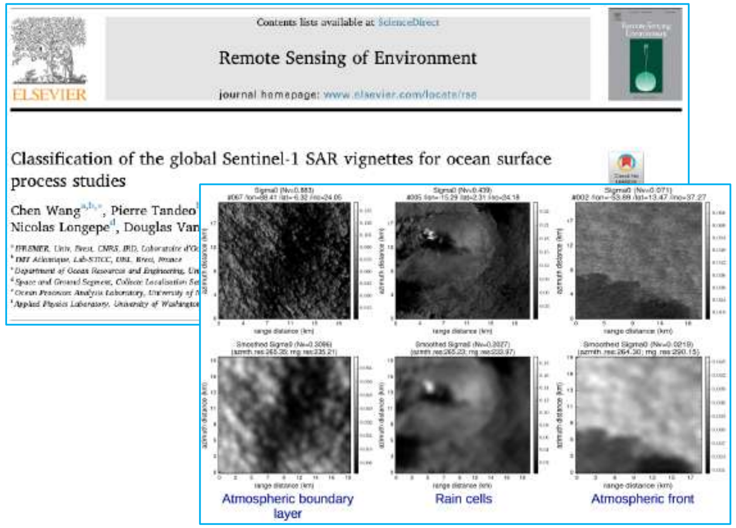
Deep Learning applied to physics

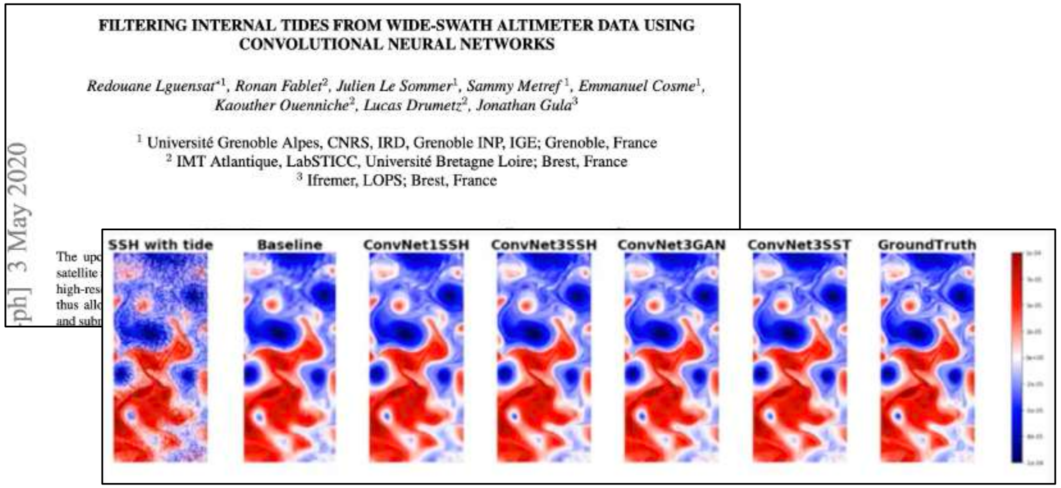
Lab-STICC

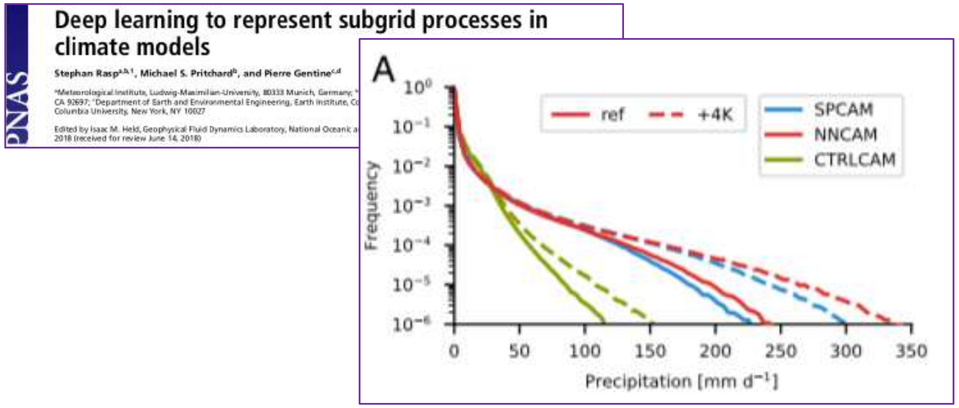
Direct applications of DL schemes to physics-related issues





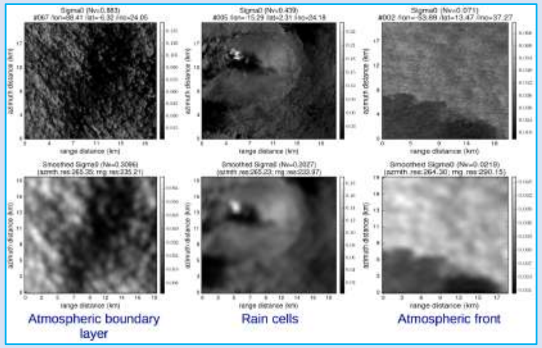




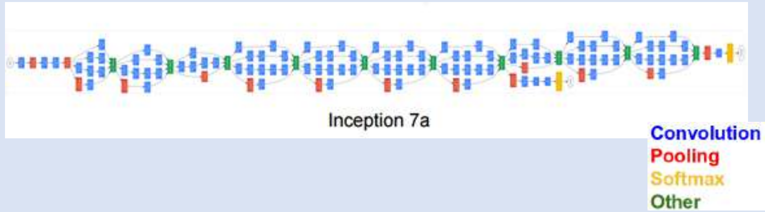


How to (directly) apply DL to physics-related issues ?

1 Build a groundtruthed dataset



2 Design or choose an architecture



Many architectures available online (VGG, ResNet, U-Net,….)

3 Choose a DL framework

TensorFlow

Keras

PYTORCH

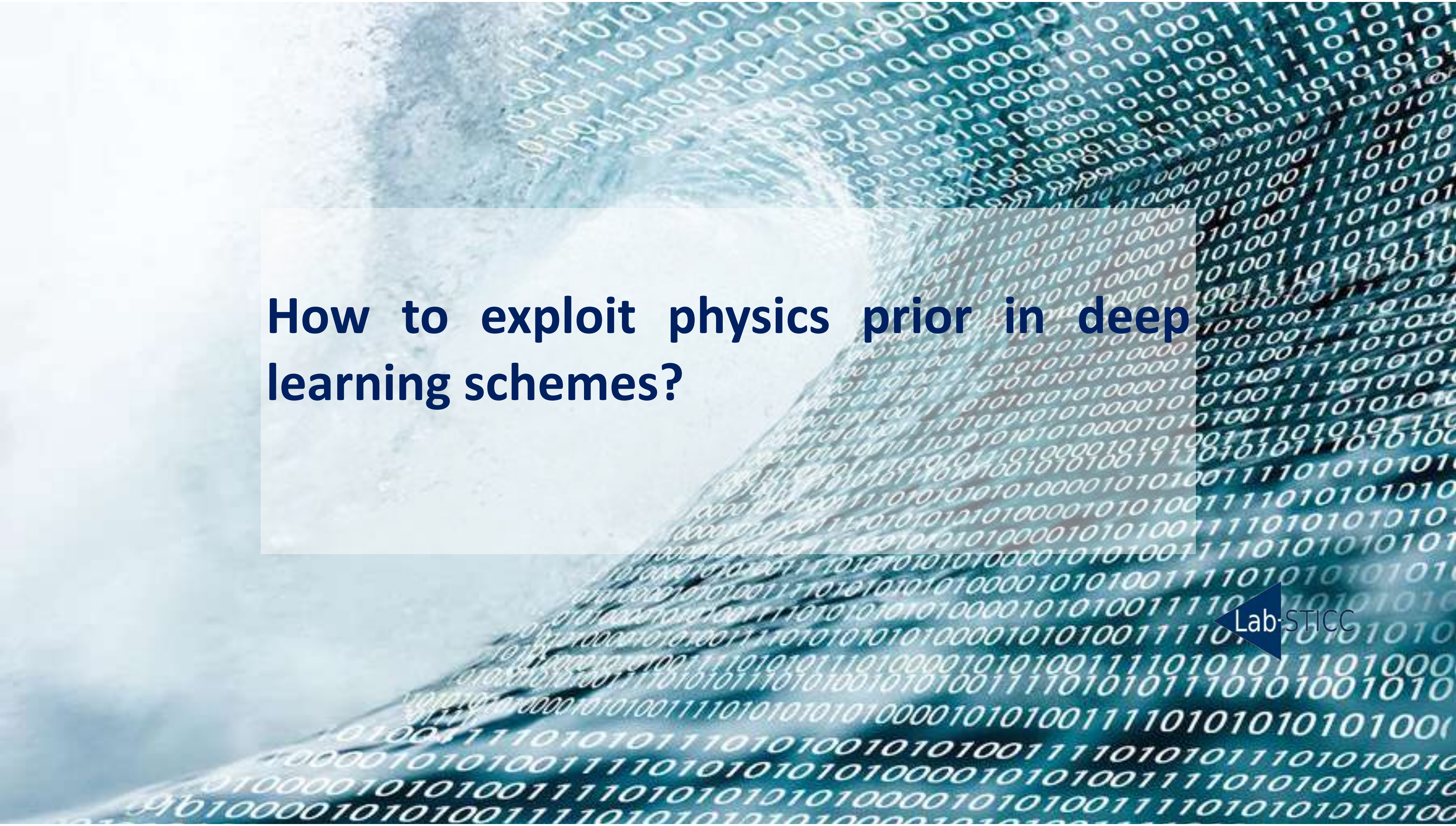
4 Learn the model

Select a training loss

Optimize model parameters using optimizers embedded in DL frameworks

A toy example using Tensorflow-Keras: https://www.tensorflow.org/tutorials/keras/classification

How to exploit physics prior in deep learning schemes?



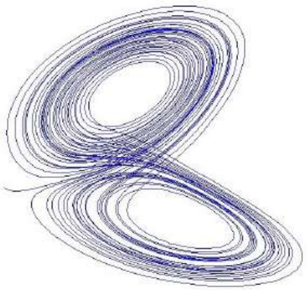
Lab-STICC

How to embed physics-driven priors in DL models ?

An illustration through neural ODE for Lorenz-63 dynamics (Fablet et al., 2018)

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

Lorenz-63 equations

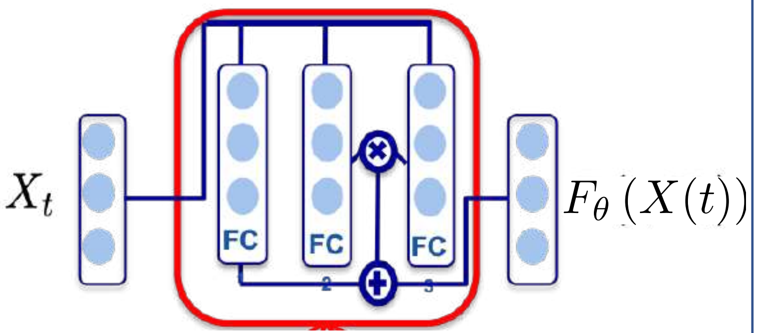


Associated Euler integration scheme

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

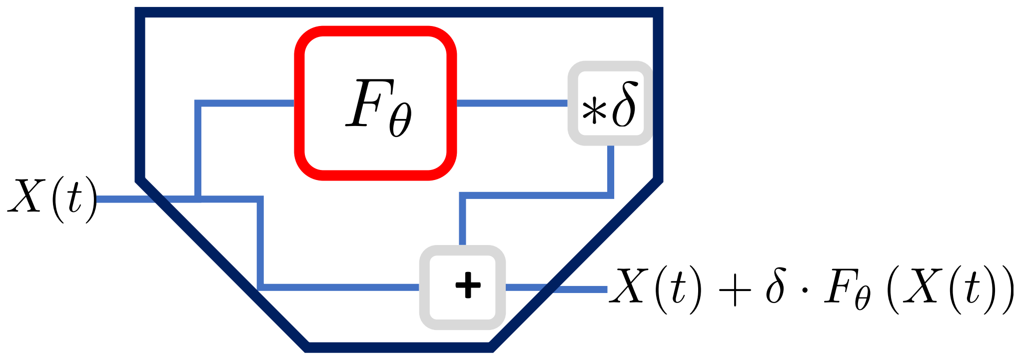
$$X(t+\delta) = X(t) + \delta \cdot F_\theta\left(X(t)\right)$$

NN architecture for differential operator



Bilinear architecture

NN architecture for integration scheme



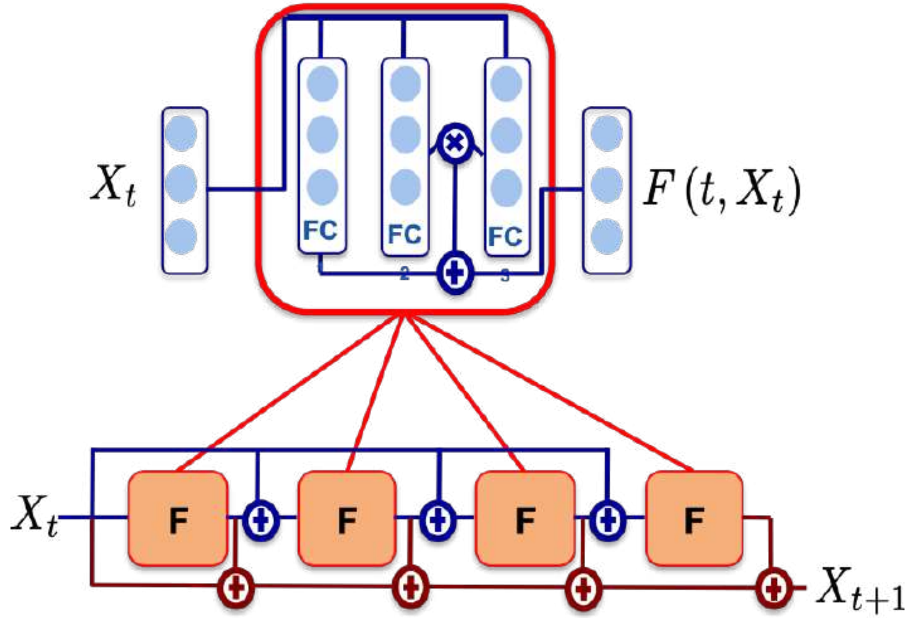
ResNet architecture (Residual Network)

How to embed physics-driven priors in DL models ?

An illustration through neural ODE for Lorenz-63 dynamics (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



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 ?

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

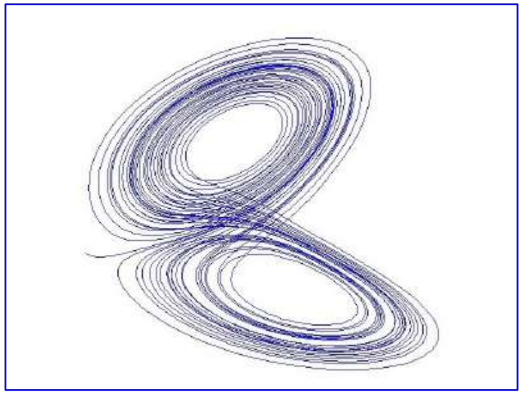
Forecasting experiments

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***

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

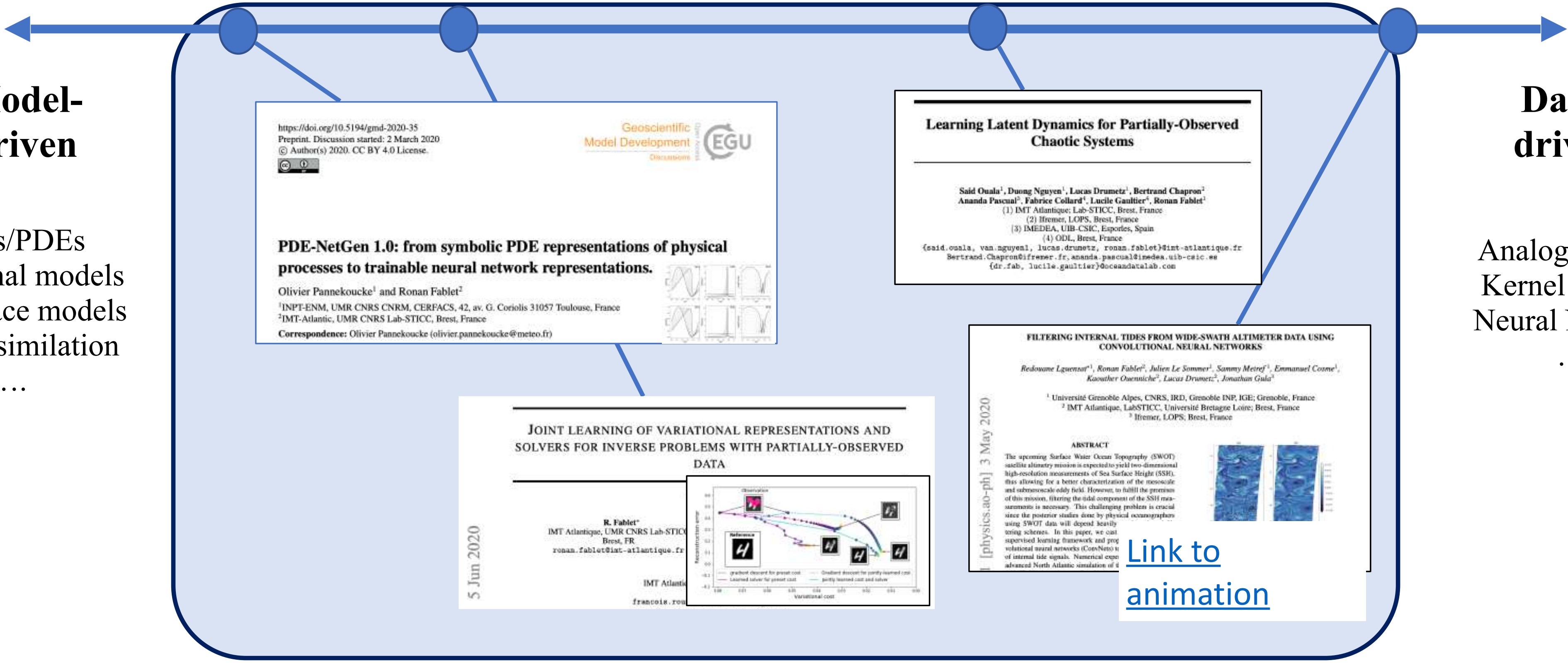
Making the most of AI and Physics Theory

- 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

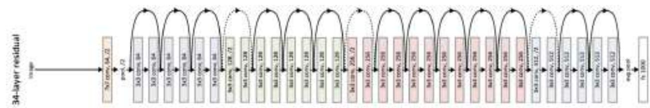
Bridging physics & AI: Expected breakthroughs

Physical model

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

Representation learning

Data-driven representation



Making the most of AI and Physics Theory

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

NN Generator from Symbolic PDEs (Pannekoucke et al., 2020)

$$\partial_t u + u\partial_x u = \kappa \partial_x^2 u$$

Symbolic calculus (Simpy)

PDE-GenNet (keras)

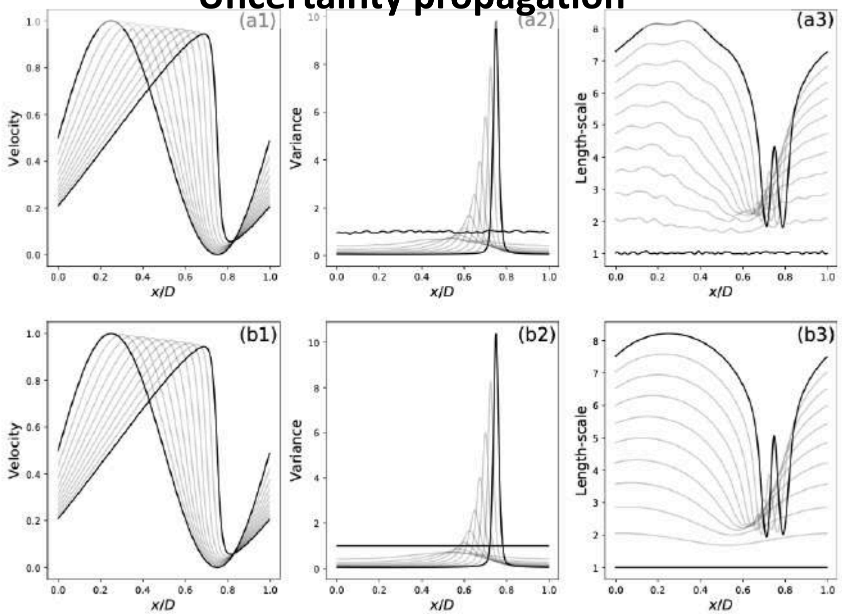
$$\begin{pmatrix} \mu_u(t) \\ \Sigma_u(t) \end{pmatrix} \rightarrow \textbf{\textit{ResNet}} \rightarrow \begin{pmatrix} \mu_u(t+1) \\ \Sigma_u(t+1) \end{pmatrix}$$

Generated code

```
# Example of computation of a derivative
kernel_Du_x_o1 = np.asarray([[0.           ...           tes.index('x')]),0.0],
        [0.0,0.0,0.0],
        [0.0,1/(2*self.dx[self.coordinates.index('x')]),0.0]]).reshape((3, 3)+(1,1))
Du_x_o1 = DerivativeFactory((3, 3),kernel=kernel_Du_x_o1,name='Du_x_o1')(u)

# Computation of trend_u
mul_1 = keras.layers.multiply([Dkappa_11_x_o1,Du_x_o1],name='MulLayer_1')
mul_2 = keras.layers.multiply([Dkappa_12_x_o1,Du_y_o1],name='MulLayer_2')
mul_3 = keras.layers.multiply([Dkappa_12_y_o1,Du_x_o1],name='MulLayer_3')
mul_4 = keras.layers.multiply([Dkappa_22_y_o1,Du_y_o1],name='MulLayer_4')
mul_5 = keras.layers.multiply([Du_x_o2,kappa_11],name='MulLayer_5')
mul_6 = keras.layers.multiply([Du_y_o2,kappa_22],name='MulLayer_6')
mul_7 = keras.layers.multiply([Du_x_o1_y_o1,kappa_12],name='MulLayer_7')
sc_mul_1 = keras.layers.Lambda(lambda x: 2.0*x,name='ScalarMulLayer_1')(mul_7)
trend_u = k
```

Uncertainty propagation



Ensemble-based prediction

NN prediction

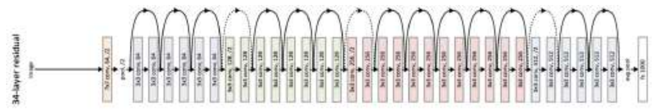
Bridging physics & AI: Expected breakthroughs

Physical model

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

Representation learning

Data-driven representation



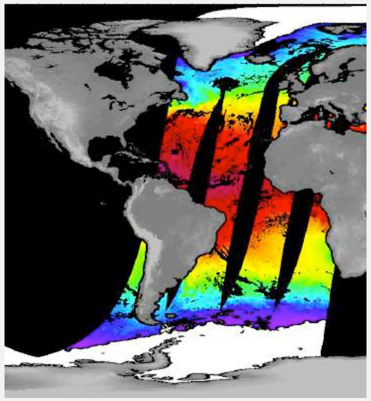
Making the most of AI and Physics Theory

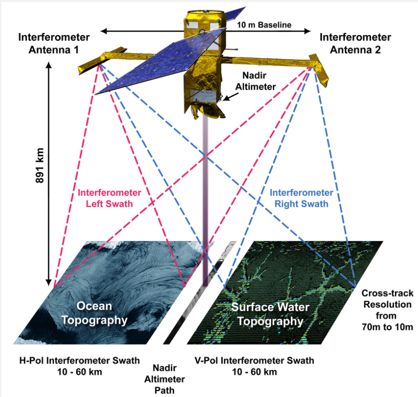
- **Model-Driven/Theory-Guided & Data-Constrained schemes**
- **Data-Driven & Physically-Sound schemes (eg, Ouala et al., 2019)**

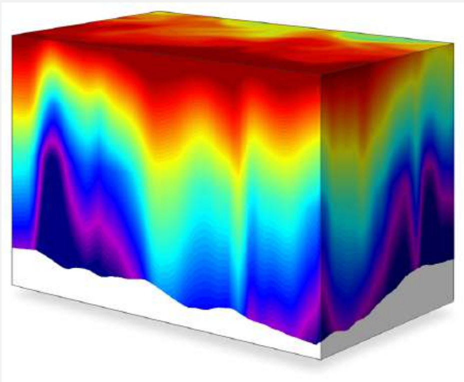
Deep learning for irregularly-sampled and partially-observed systems

learning for partially-observed systems / from irregularly-sampled data [Oualа et al., 2020; Fablet et al., 2020]

Can we learn directly from observation data ?







Generic issue:

Joint identification and inversion

Dynamical model

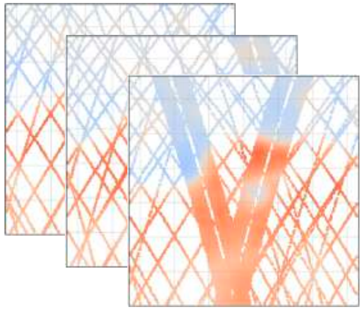
$$X_t \rightarrow \partial_t X = F(X, \xi, t, \theta) \rightarrow X_{t+1}$$

+

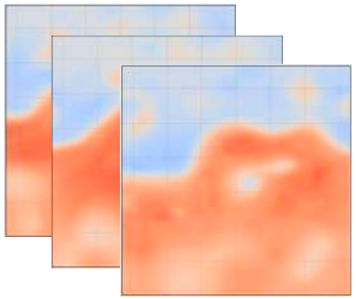
Observation model

$$Y_t = H(X, \zeta, t, \phi)$$

(Weak constraint) **4DVar Data Assimilation (DA) formulation**



Partial observations y



True states x

State-space formulation:

$$\begin{cases} \dfrac{\partial x(t)}{\partial t} & = & \mathcal{M}\left(x(t)\right) \\ y(t) & = & x(t) + \epsilon(t), \forall t \in \{t_0, t_0 + \Delta t...., , t_0 + N\Delta t\} \end{cases}$$

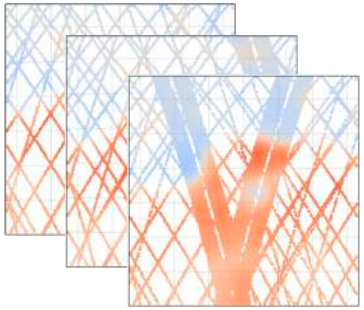
Associated variational formulation:

$$\arg\min_x \lambda_1 \sum_i \|x(t_i) - y(t_i)\|^2_{\Omega_{t_i}} + \lambda_2 \sum_n \|x(t_i) - \Phi(x)(t_i)\|^2$$

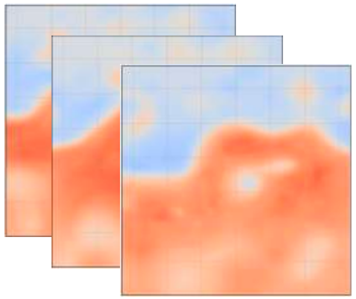
with $\Phi(x)(t) = x(t - \Delta) + \int_{t-\Delta}^{t} \mathcal{M}\left(x(u)\right) du$

$$\boxed{\arg\min_x \lambda_1 \|x - y\|^2_\Omega + \lambda_2 \|x - \Phi(x)\|^2}$$

4DVar Data Assimilation (DA) formulation



Partial observations y



True states x

Model-driven schemes:

$$\arg\min_x \underbrace{\lambda_1 \|x - y\|_\Omega^2 + \lambda_2 \|x - \Phi(x)\|^2}_{U_\Phi(x, y, \Omega)}$$

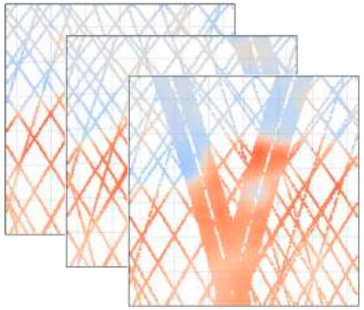
Gradient-based solver (adjoint/Euler-Lagrange method):

$$x^{(k+1)} = x^{(k)} - \alpha \nabla_x U_\Phi\left(x^{(k)}, y, \Omega\right)$$

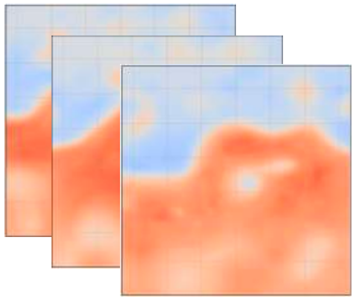
Implementation of 4DVar Data Assimilation using Deep Learning frameworks

- Implement operator Φ as a neural network (e.g., neural ODE architecture w.r.t. a known dynamical operator)
- Use embedded automatic differentiation tool to implement the above gradient descent without deriving analytically the adjoint operator
- Link to an example for L63 dynamics: https://colab.research.google.com/drive/1SyU6PzOAzgBz4_U_5WbPMriwT4bLXKvh?usp=sharing

End-to-end learning for inverse problems (Fablet etal., 2020)



Partial observations y



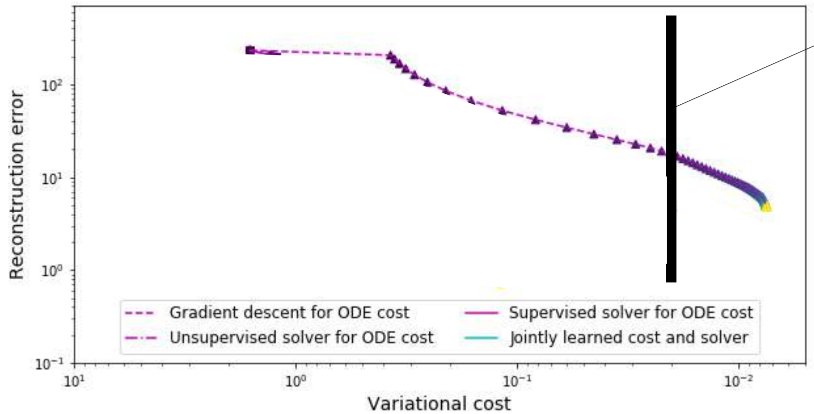
True states x

Model-driven schemes: $$\widehat{x} = \arg\min_x \underbrace{\lambda_1 \left\|x-y\right\|^2_\Omega + \lambda_2 \left\|x-\Phi(x)\right\|^2}_{U_\Phi\left(x^{(k)},y,\Omega\right)}$$

Gradient-based solver (adjoint/Euler-Lagrange method):

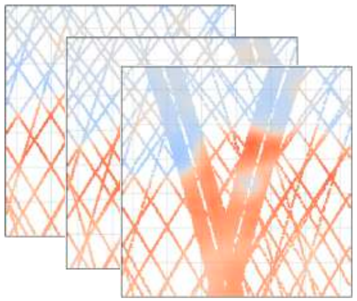
$$x^{(k+1)} = x^{(k)} - \alpha \nabla_x U_\Phi\left(x^{(k)},y,\Omega\right)$$

No control on the reconstruction error $$x^{true} \neq \arg\min_x U_\Phi\left(x^{(k)},y,\Omega\right)$$

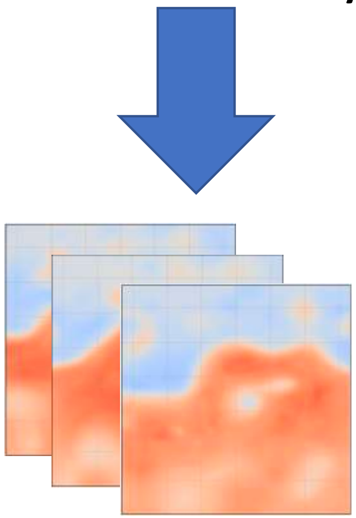


Variational cost for the true state

End-to-end learning for inverse problems (Fablet et al., 2020)



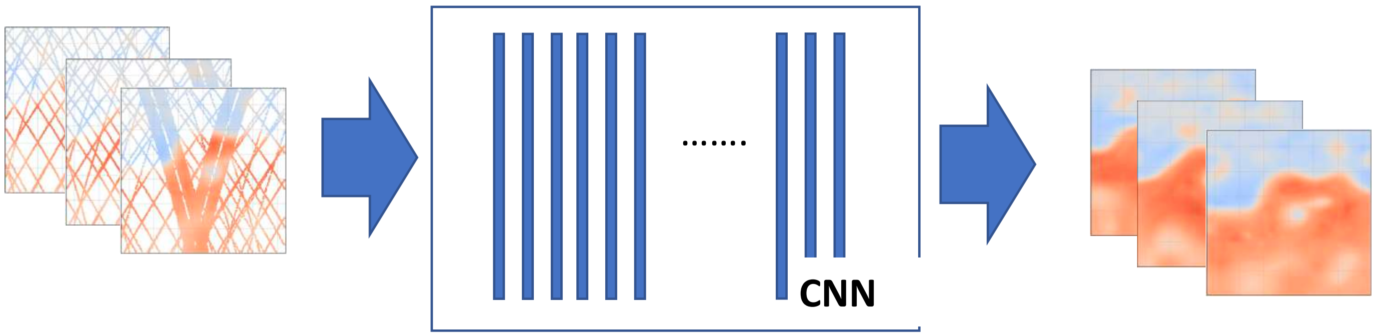
Partial observations y

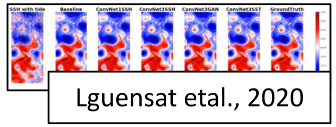


True states x

Model-driven schemes: $\widehat{x} = \arg\min_x \lambda_1 \|x - y\|_\Omega^2 + \lambda_2 \Phi(x)$

Direct learning for inverse problems: $\widehat{x} = \Psi(y)$ $\quad y \rightarrow \boxed{\text{CNN}} \rightarrow x$

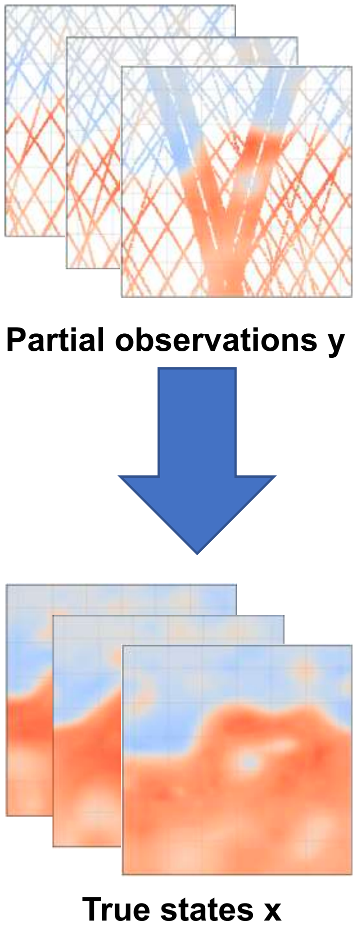




Examples of CNN architectures: Reaction-Diffusion architectures, ADMM-inspired architectures,...

Good performance but possibly weak interpretability/generaiization capacities of the solution byeond the training cases

End-to-end learning for inverse problems (Fablet et al., 2020)



Model-driven schemes: $\widehat{x} = \arg\min_x \lambda_1 \|x - y\|_\Omega^2 + \lambda_2 \Phi(x)$

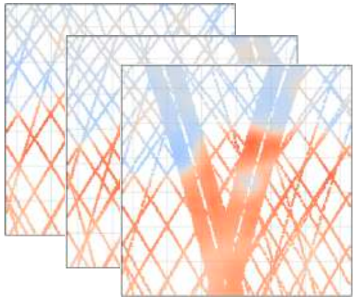
Direct learning for inverse problems: $\widehat{x} = \Psi(y)$ $\quad y \rightarrow \boxed{\textbf{CNN}} \rightarrow x$

Proposed scheme: joint learning of the variational model and solver

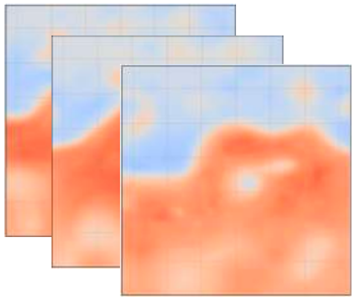
- **Theoretical bi-level optimization**

$$\arg\min_\Phi \sum_n \|x_n - \tilde{x}_n\|^2 \text{ s.t. } \tilde{x}_n = \arg\min_{x_n} U_\Phi(x_n, y_n, \Omega_n)$$

End-to-end learning for inverse problems (Fablet et al., 2020)



Partial observations y



True states x

Model-driven schemes: $\widehat{x} = \arg\min_x \lambda_1 \left\| x - y \right\|_\Omega^2 + \lambda_2 \Phi(x)$

Direct learning for inverse problems: $\widehat{x} = \Psi(y)$ $\quad y \rightarrow \boxed{\text{CNN}} \rightarrow x$

Proposed scheme: joint learning of the variational model and solver

- **Theoretical bi-level optimization**

$$\arg\min_\Phi \sum_n \left\| x_n - \tilde{x}_n \right\|^2 \text{ s.t. } \tilde{x}_n = \arg\min_{x_n} U_\Phi(x_n, y_n, \Omega_n)$$

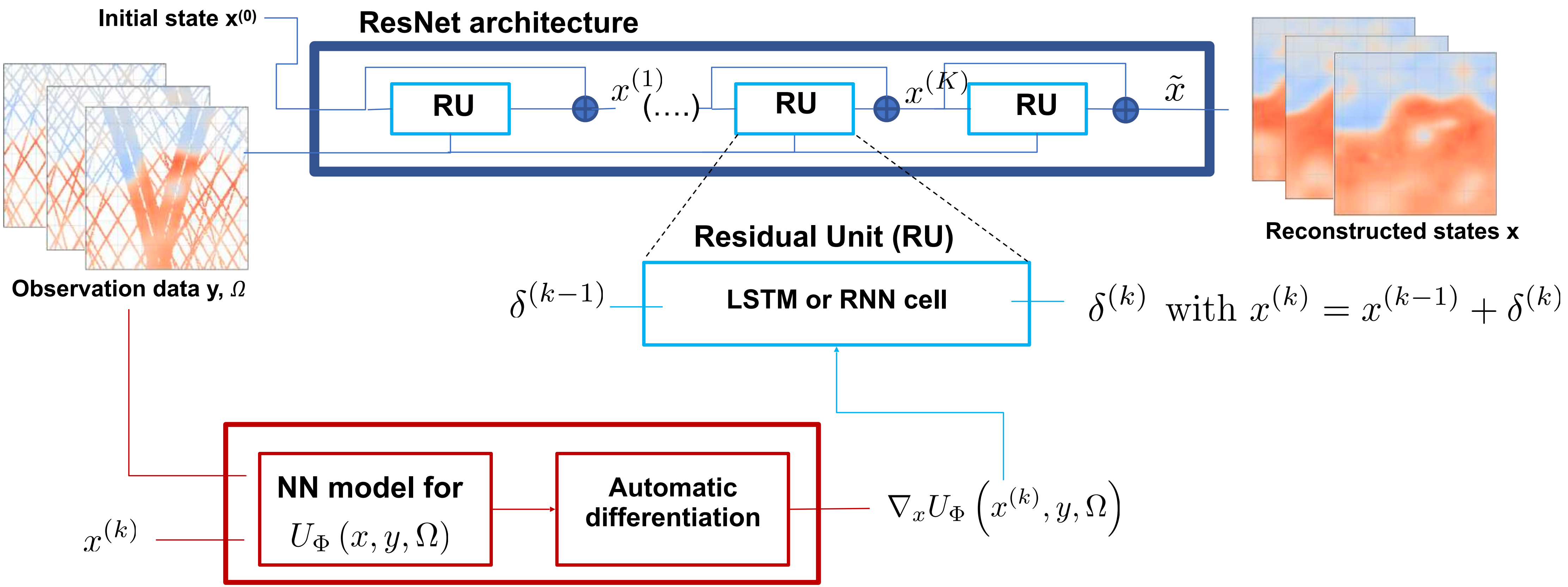
- **Restated with a gradient-based NN solver for inner minimization**

$$\arg\min_{\Phi,\Gamma} \sum_n \left\| x_n - \tilde{x}_n \right\|^2 \text{ s.t. } \tilde{x}_n = \Psi_{\Phi,\Gamma}(x_n^{(0)}, y_n, \Omega_n)$$

Iterative NN solver using automatic differentiation to compute gradient $\nabla_x U_\Phi \left(x^{(k)}, y, \Omega \right)$

End-to-end learning for inverse problems (Fablet et al., 2020)

Proposed scheme: associated NN architecture

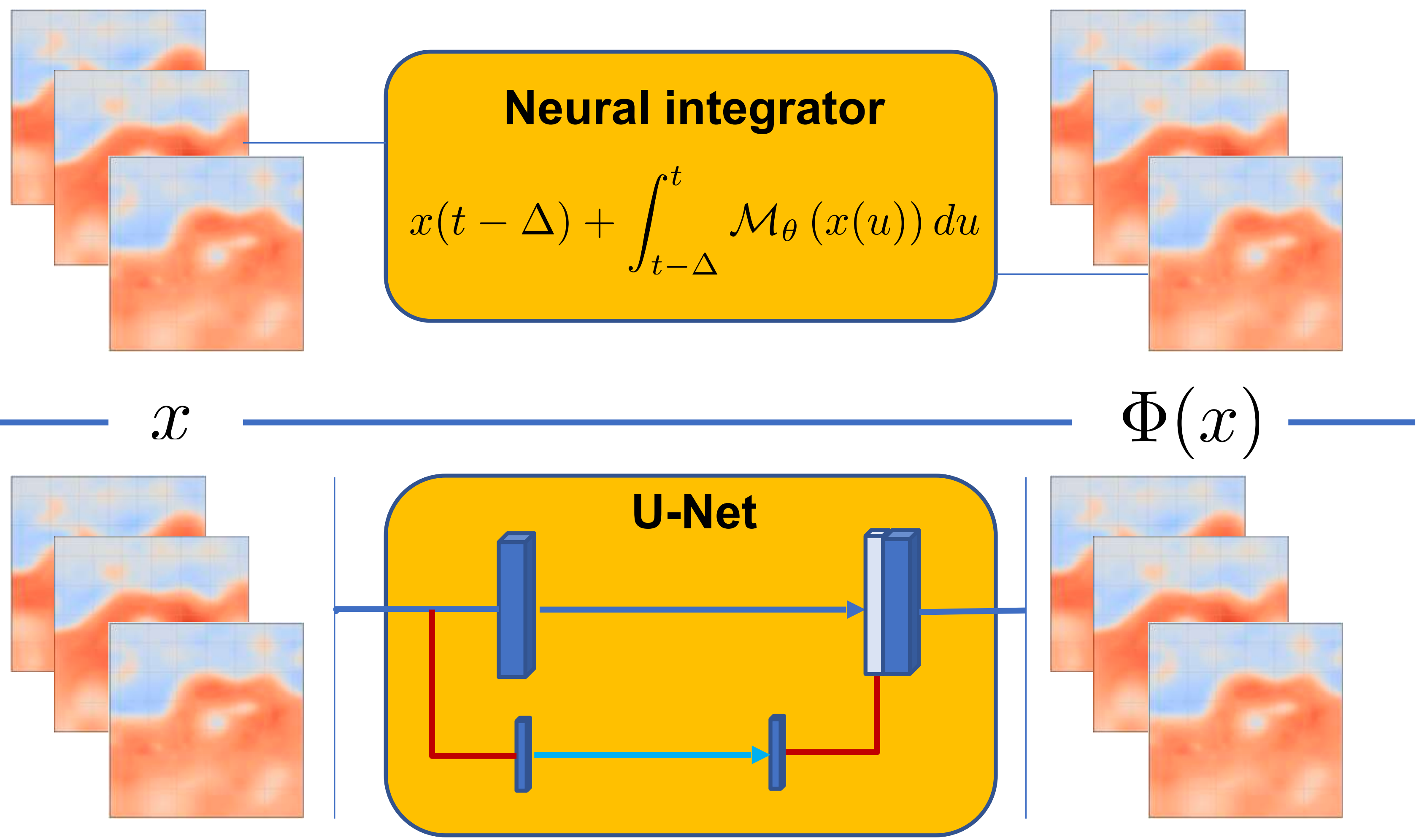


End-to-end learning for 4DVar DA: projection operator Φ

Parameterization using (learnable) ODE operator

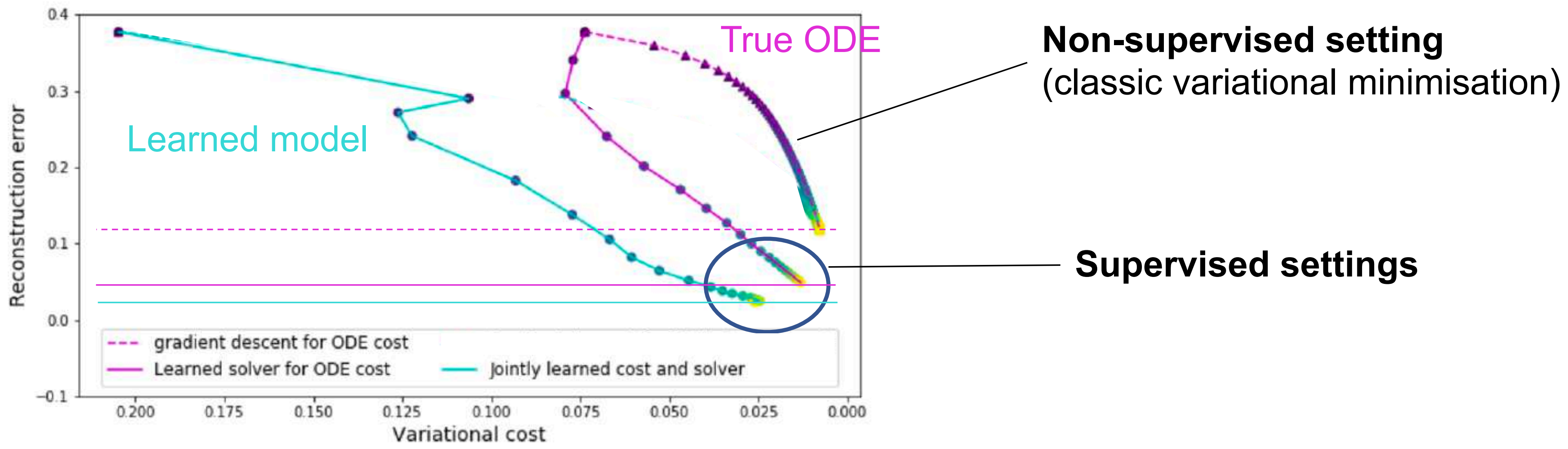
$$\frac{\partial x(t)}{\partial t} = \mathcal{M}_\theta\left(x(t)\right)$$

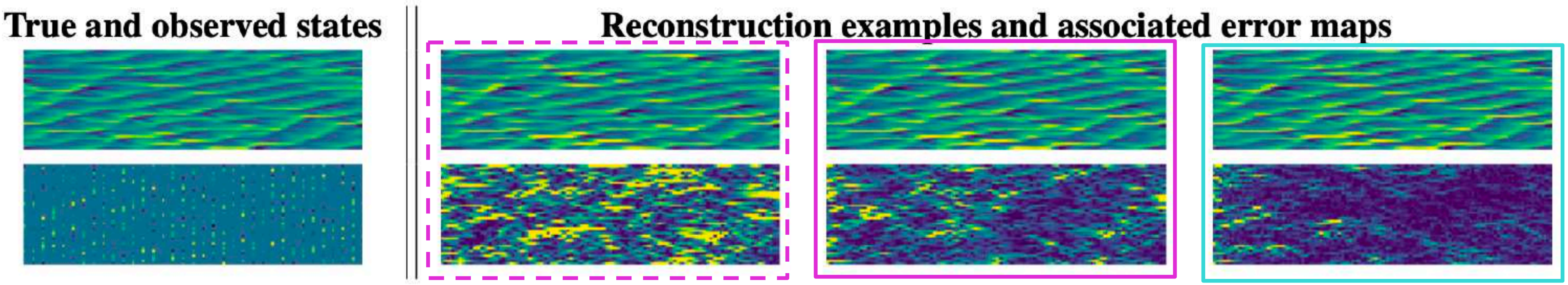
Two-scale U-Net-like Parameterization (Gibbs Field)



End-to-end learning for inverse problems (Fablet etal., 2020)

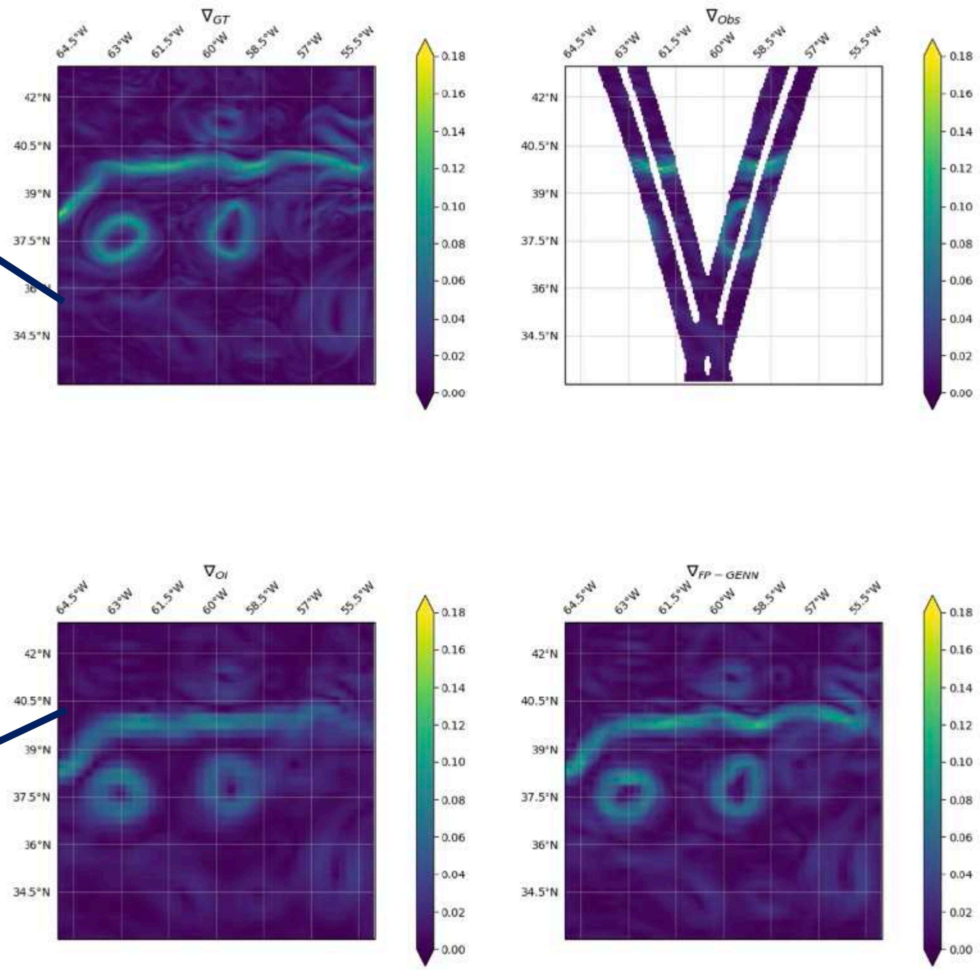
Illustration on Lorenz-96 dynamics (Bilinear ODE)

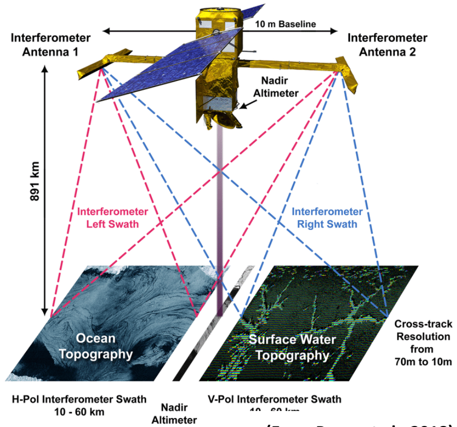




An application for upcoming SWOT mission

Groundtruth





(From Perez et al., 2018)

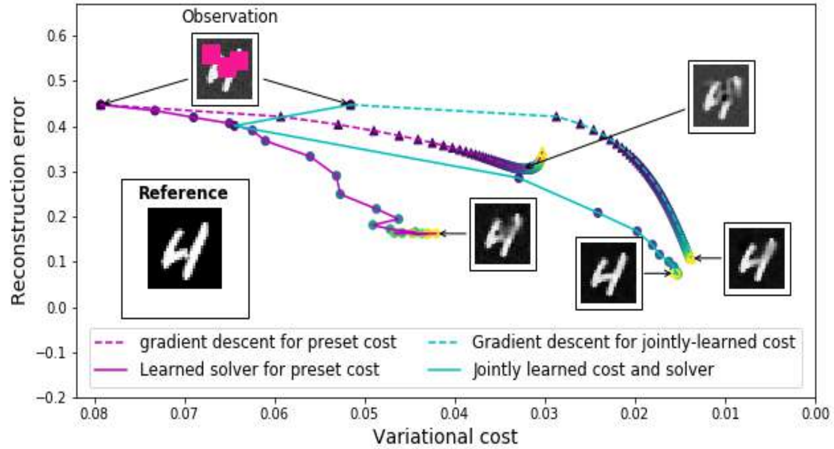
Proposed NN framework (Fablet et al., 2019)

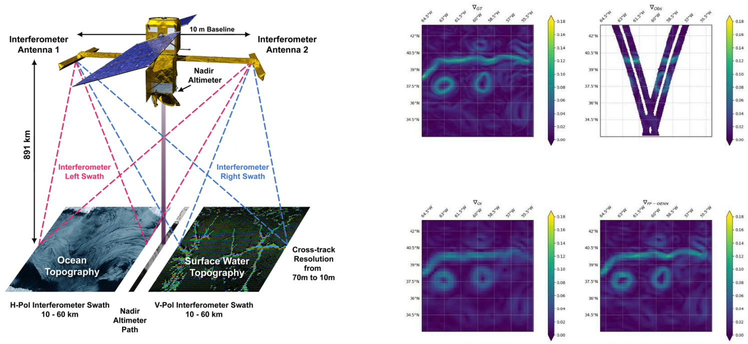
State-of-the-art operational processing

End-to-end learning for inverse problems (Fablet etal., 2020)

Key messages

- **We can bridge DNN and variational models to solve inverse problems**
- **Learning both variational priors and solvers using groundtruthed (simulation) or observation-only data**
- **The best model may not be the TRUE one for inverse problems**
- **Generic formulation/architecture beyond space-time dynamics**



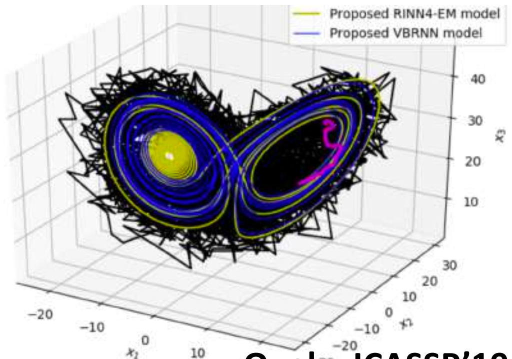


Preprint: https://arxiv.org/abs/2006.03653

Code: https://github.com/CIA-Oceanix

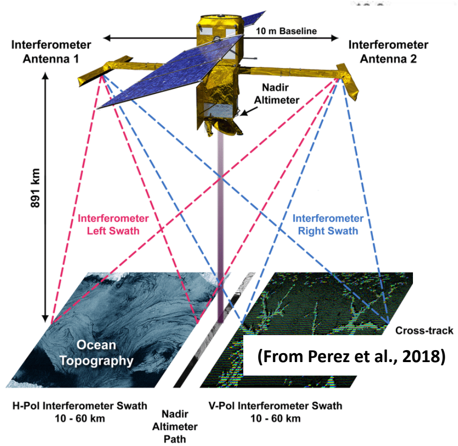
End-to-end learning from real observation data ?

Scarce time sampling

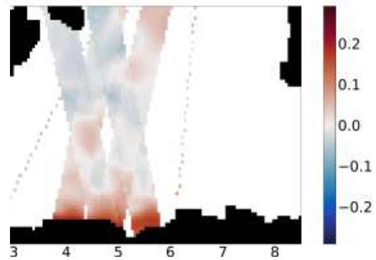


Ouala, ICASSP'19

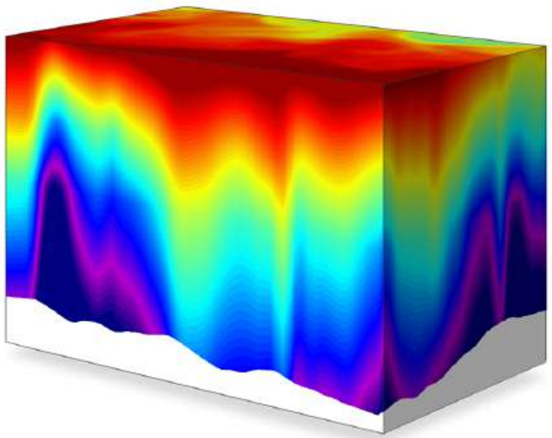
Noisy and irregular sampling



(From Perez et al., 2018)



Nguyen, ICASSP'20

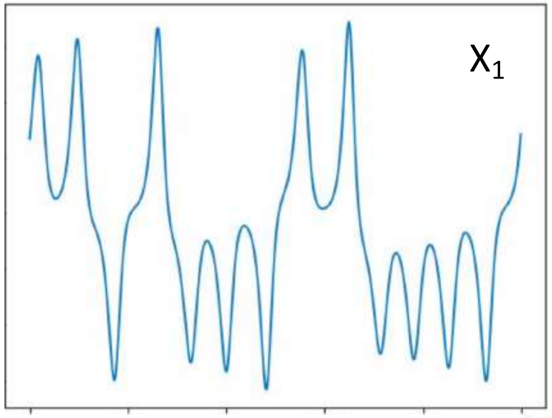


Partially-observed system

Ouala, preprint 2019

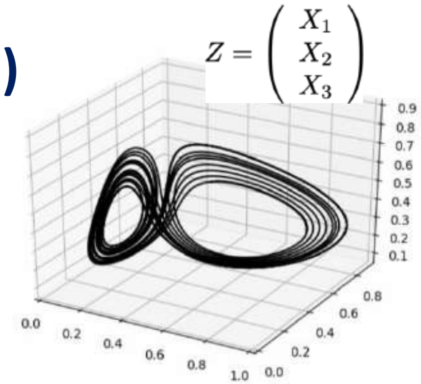
Neural ODE for partially-observed systems [Ouala et al., 2020]

Illustration for L63 assuming only the first components is observed

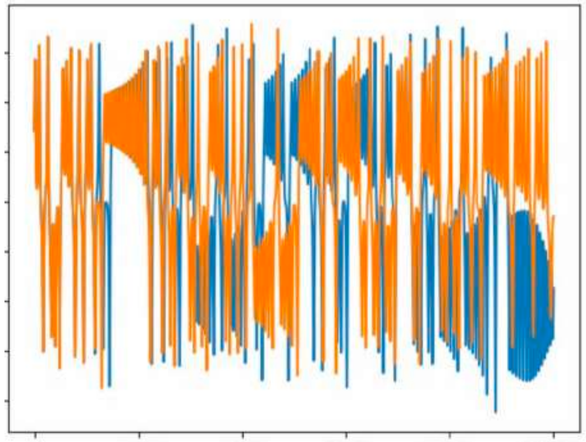


Learning Latent (unobserved) dynamics

$$d_t Z_t = \Phi_\theta(Z_t)$$



Objectives: acccurate short-term forecast and realistic « long-term » patterns for X_1



Approach: trainable variational formulation with latent dynamics

Neural ODE for partially-observed systems [Ouala et al., 2019]

Problem statement: **end-to-end learning of the latent (augmented) space and of the associated dynamics**

$$X_t = \begin{pmatrix} x_t \\ z_t \end{pmatrix}$$

Observed variables

Unknown variables

Dynamical model in the latent space

$$\partial_t X_t = f_\theta \left(X_t \right)$$

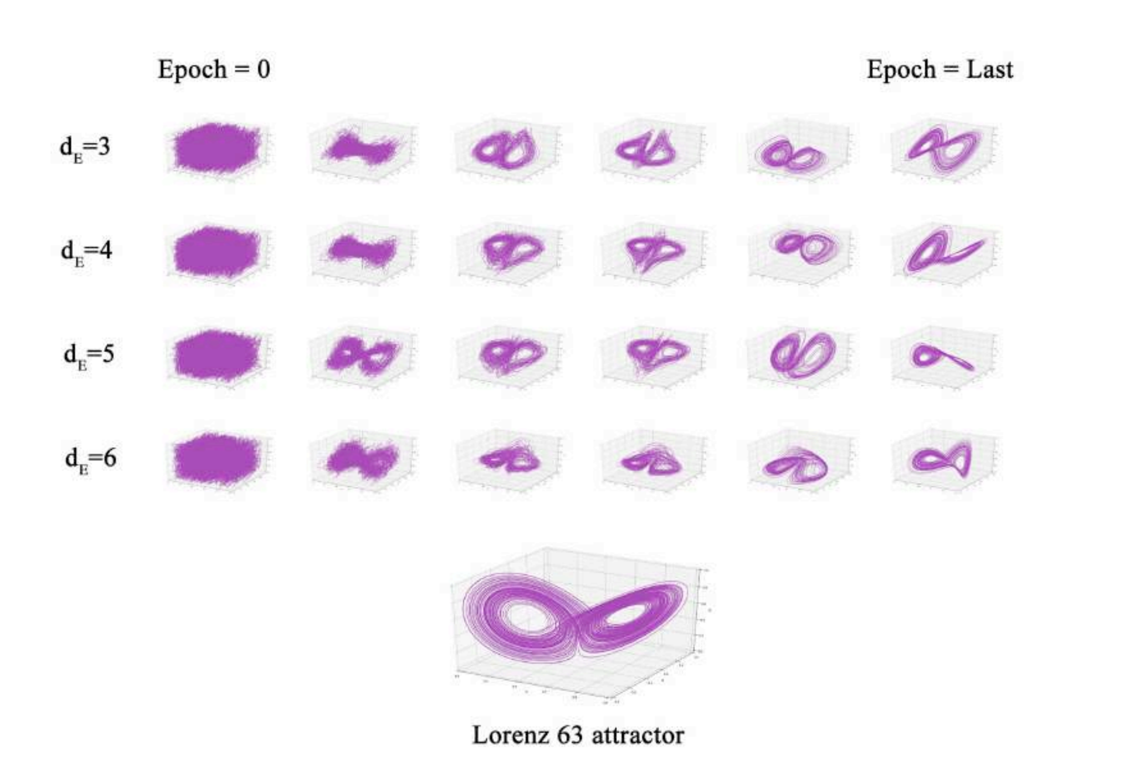
Goals:

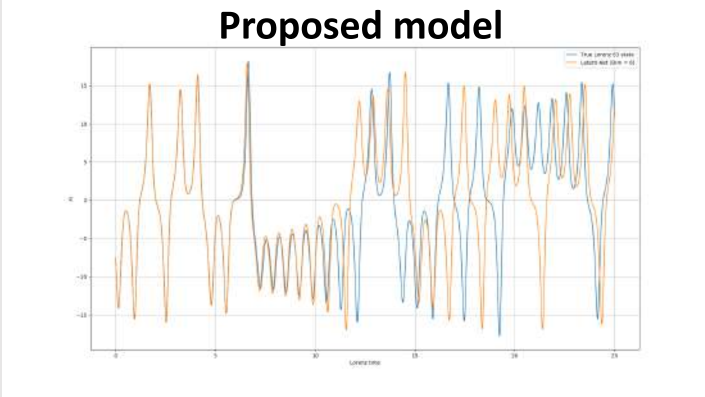
1. **Learn model parameters θ from observed time series**
2. **Forecast future observed states given previous ones**

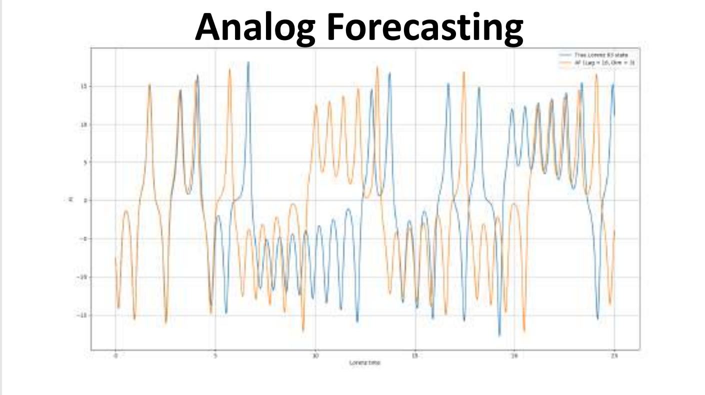
Proposed approach: WC 4DVar formulation with an unknown dynamical model

Neural ODE for partially-observed systems [Ouala et al., 2020]

Illustration on Lorenz-63 dynamics







Summary

- ***NNs as numerical schemes for ODE/PDE/energy-based representations of geophysical flows***

- ***Embedding geophysical priors in NN representations*** (e.g., Lguensat et al., 2019; Ouala et al., 2019)

- ***End-to-end architecture for jointly learning a representation (**eg, ODE**) and a solver*** (e.g., Fablet et al., 2020)

- ***Towards stochastic representations embedded in NN architectures*** (e.g., Pannekoucke et al., 2020, Nguyen etal., 2020)

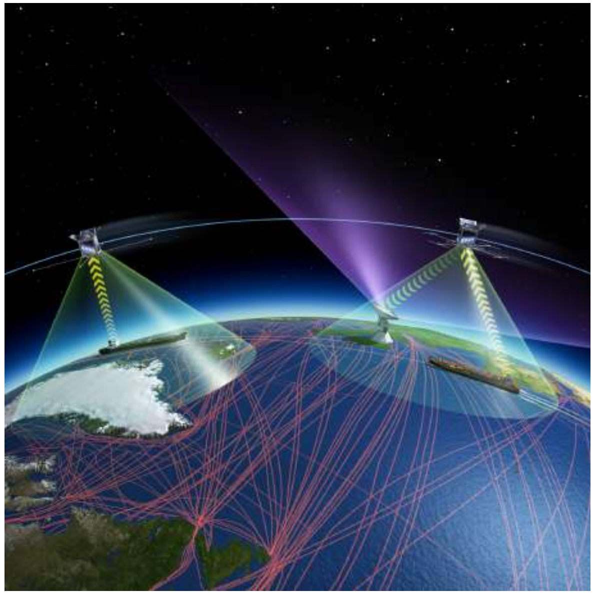
Beyond Ocean Dynamics

Learning stochastic hidden dynamics

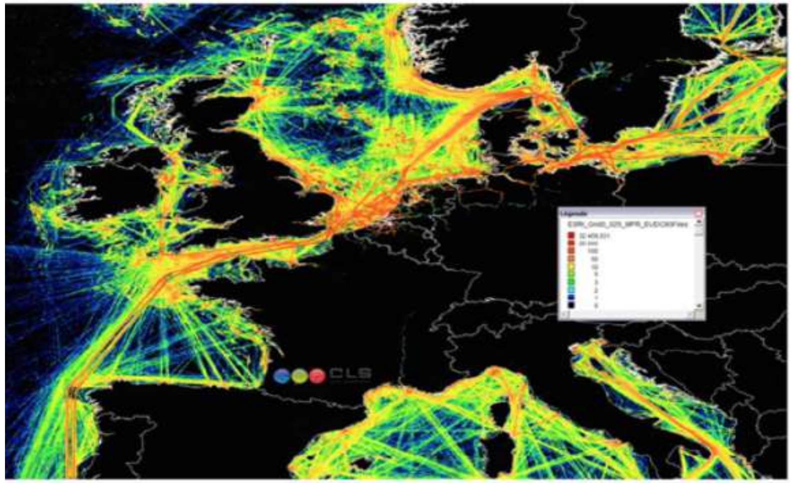
Lab-STICC

Learning stochastic hidden dynamics [Nguyen et al., 2018]

The example of AIS Vessel trajectory data

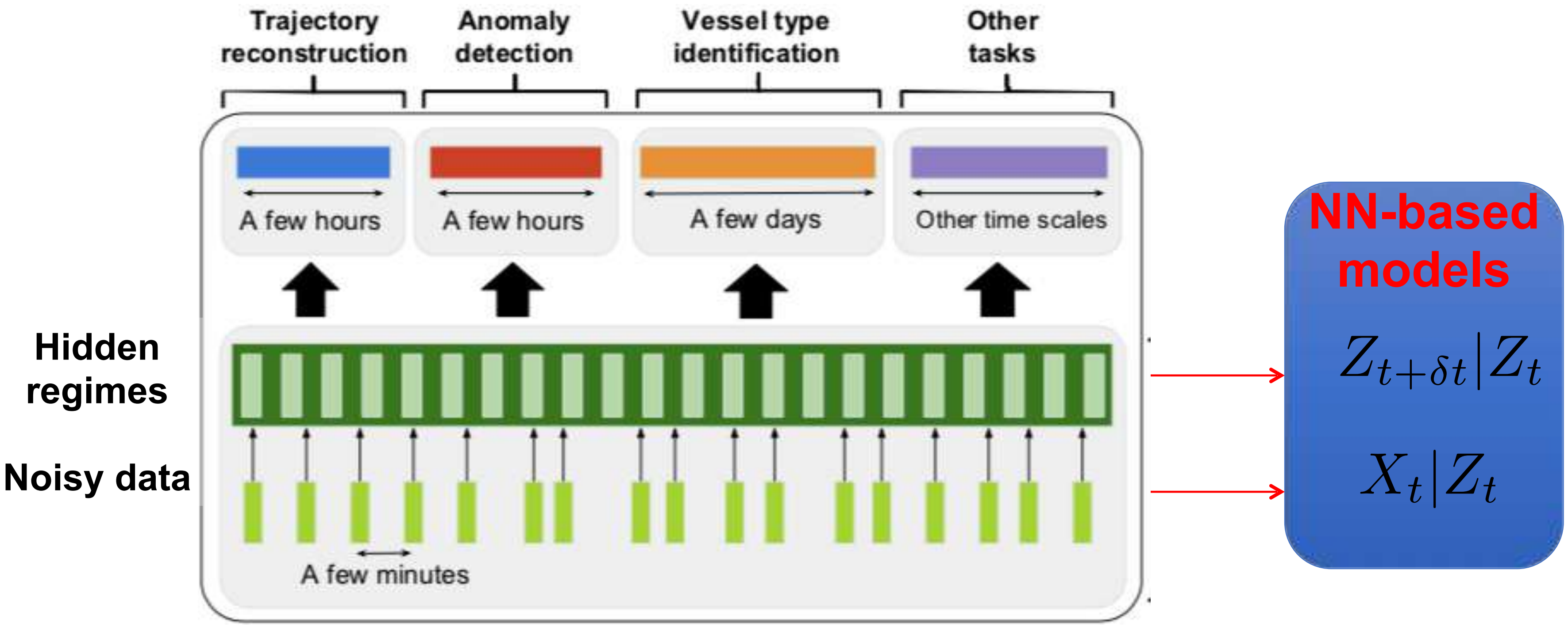


- Millions of AIS positions daily
- Noisy data: irregular sampling, corrupted data



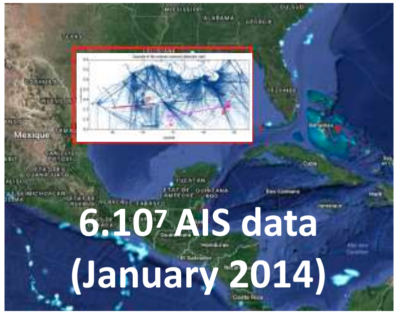
How can we learn from AIS data streams ?

Learning stochastic hidden dynamics [Nguyen et al., 2018]



Model training from noisy AIS streams using variational Bayesian approximation

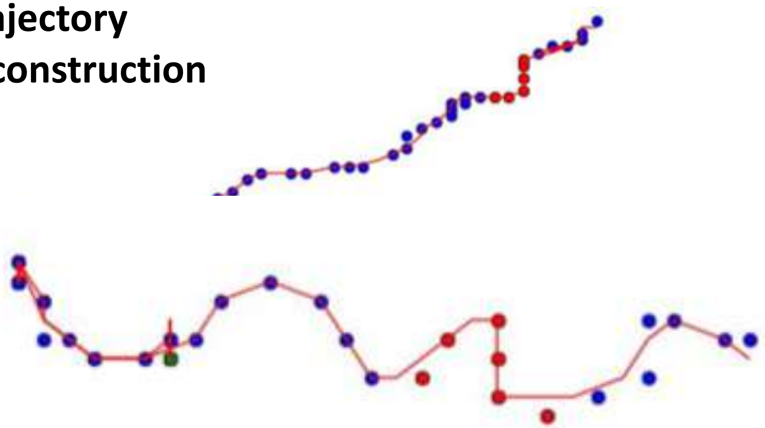
Learning stochastic hidden dynamics [Nguyen et al., 2018]



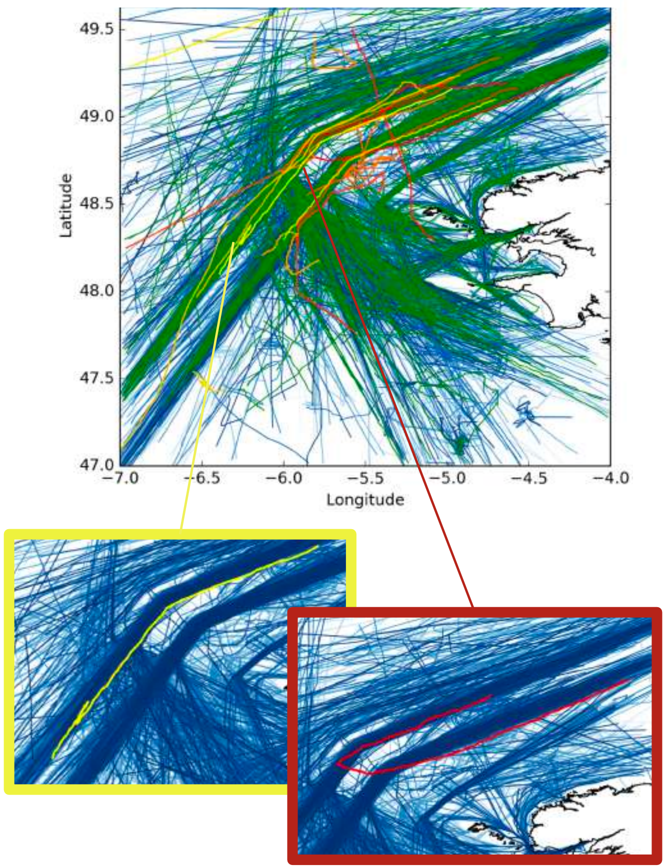
Vessel type recognition

~88% of correct recognition

Trajectory reconstruction



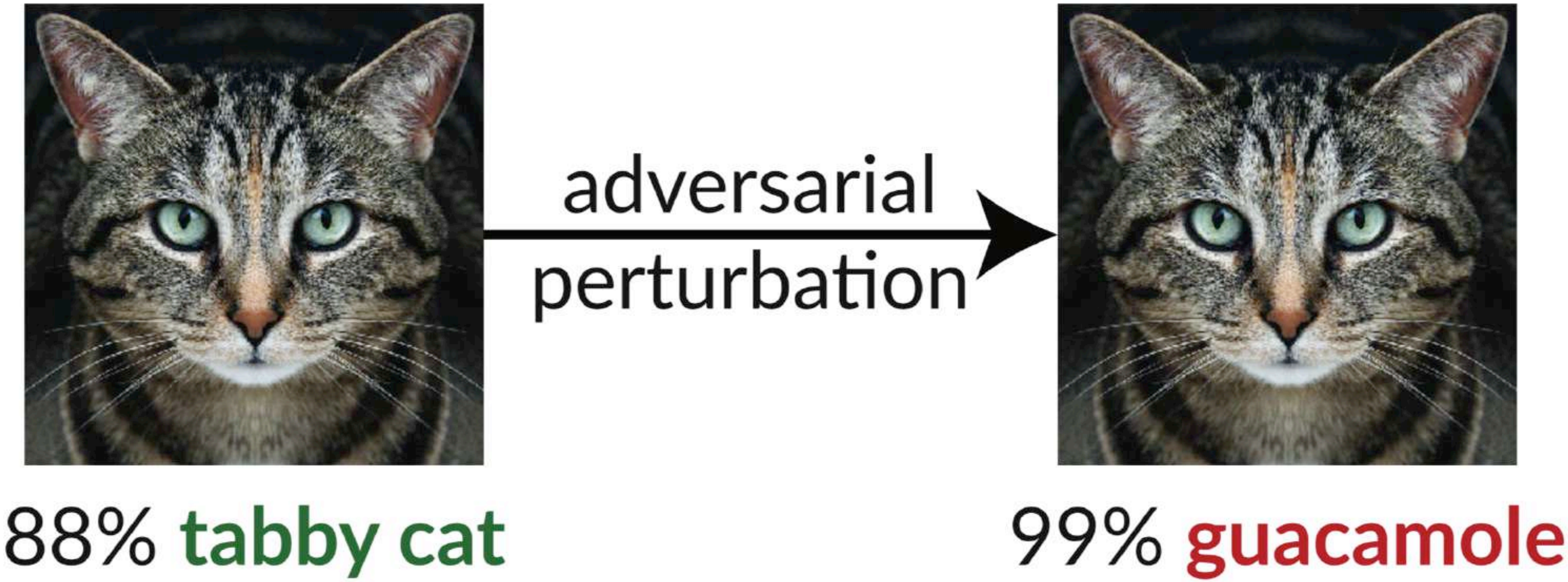
Abnormal behaviour detection



Beyond Ocean Dynamics

Dynamical System Theory for Deep Learning

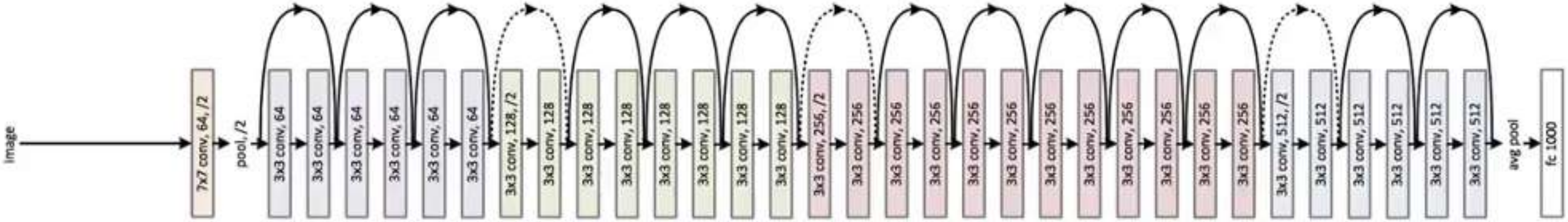
Understanding DL models ?



Szegedy et al., 2015

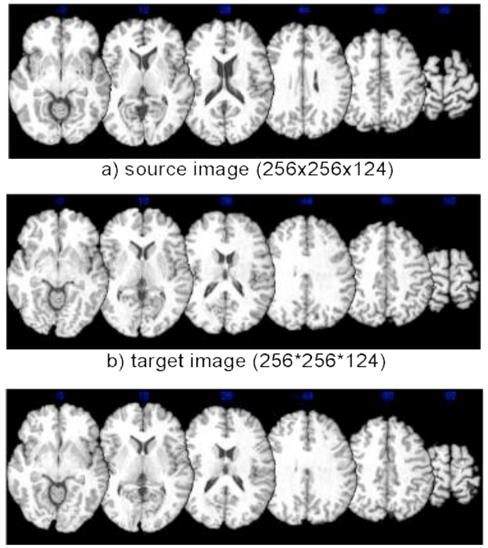
Understanding ResNets [Rousseau et al., 2019]

ResNet [He et al., 2015] regarded as space registration machines



- **Image registration examples**

[Matlab tutorial]



a) source image (256x256x124)

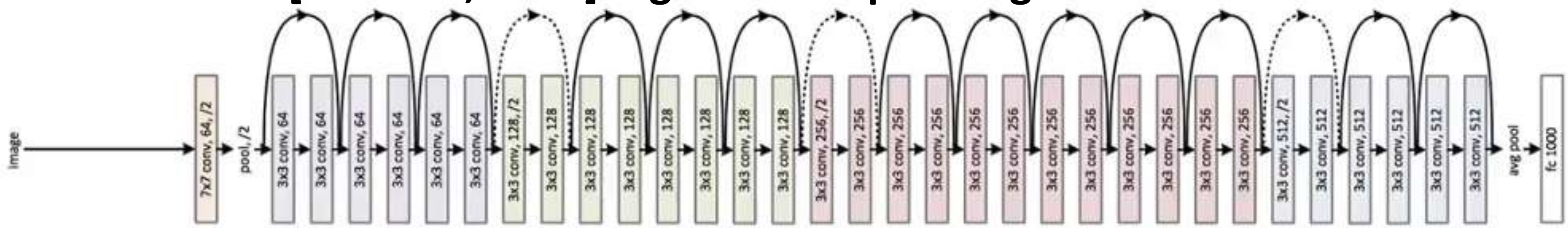
b) target image (256*256*124)

c) registered image (source to target)

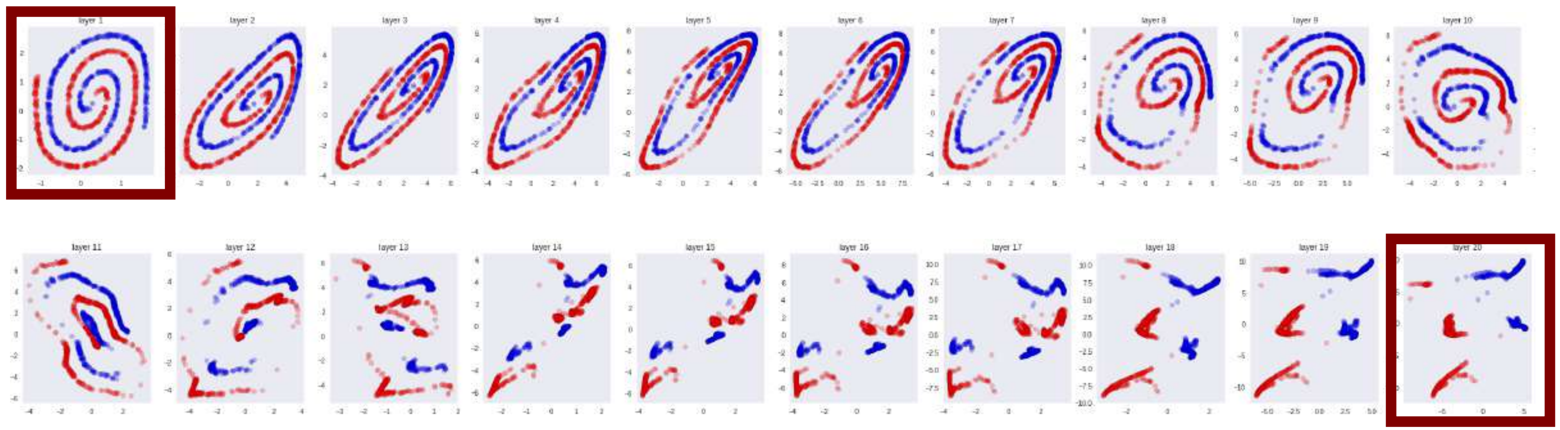
[Dramms tutorial]

Understanding ResNets [Rousseau et al., 2019]

ResNet [He et al., 2015] regarded as space registration machines



Original feature space



Registered space to make feasible a linear separation between classes

AI Chair OceaniX 2020-2024

Physics-informed AI for Observation-Driven Ocean AnalytiX

PI: R. Fablet, Prof. IMT Atlantique, Brest
Web: https://cia-oceanix.github.io/

Thank you.

Internship, PhD and postdoc opportunities