Machine learning and physical modelling-3

julien.brajard@nersc.no

21-25 January 2018

NERSC/Sorbonne University

`https://github.com/brajard/Geilo-Winter-school`

AI Art?

Edmond de Bellamy by Obvious(collective)

Generated using a Generative Adversarial Network.

Selling prince (Oct. 2018): $432,000

Table of contents

1. Machine learning to reveal complex connexions

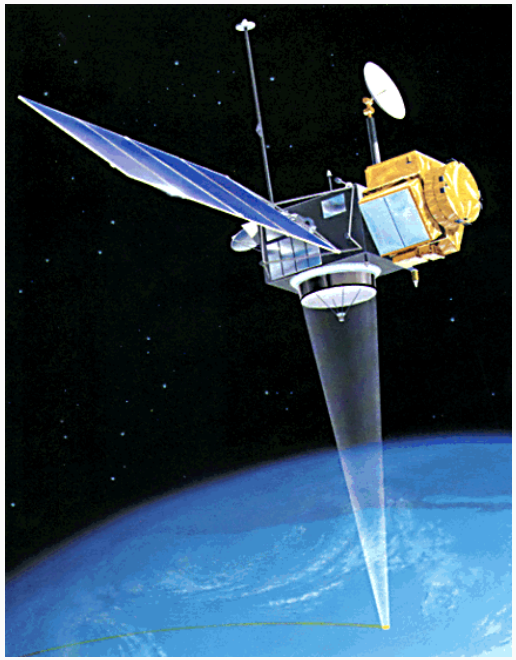
2. Machine learning and differential equations

3. Merge approaches: between physics and machine learning

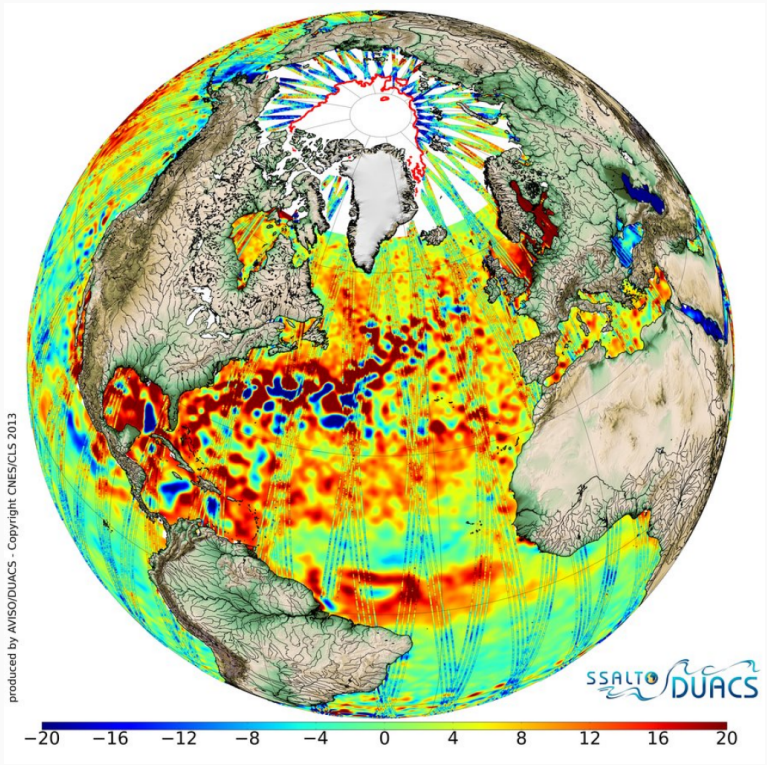
Machine learning to reveal complex connexions

Altimeter data

Satellite altimeter



Sea surface height anomaly

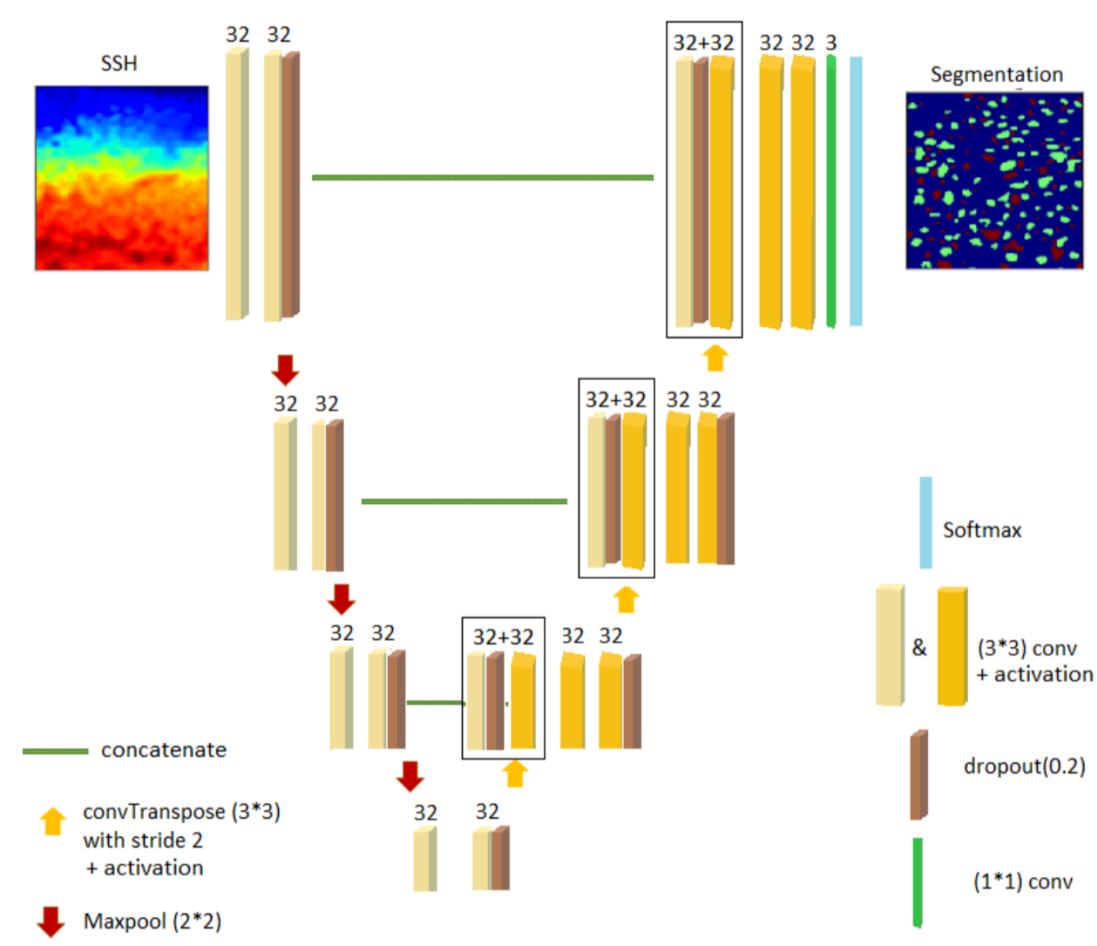


Structure detection

Can we detect and charaterize eddies?

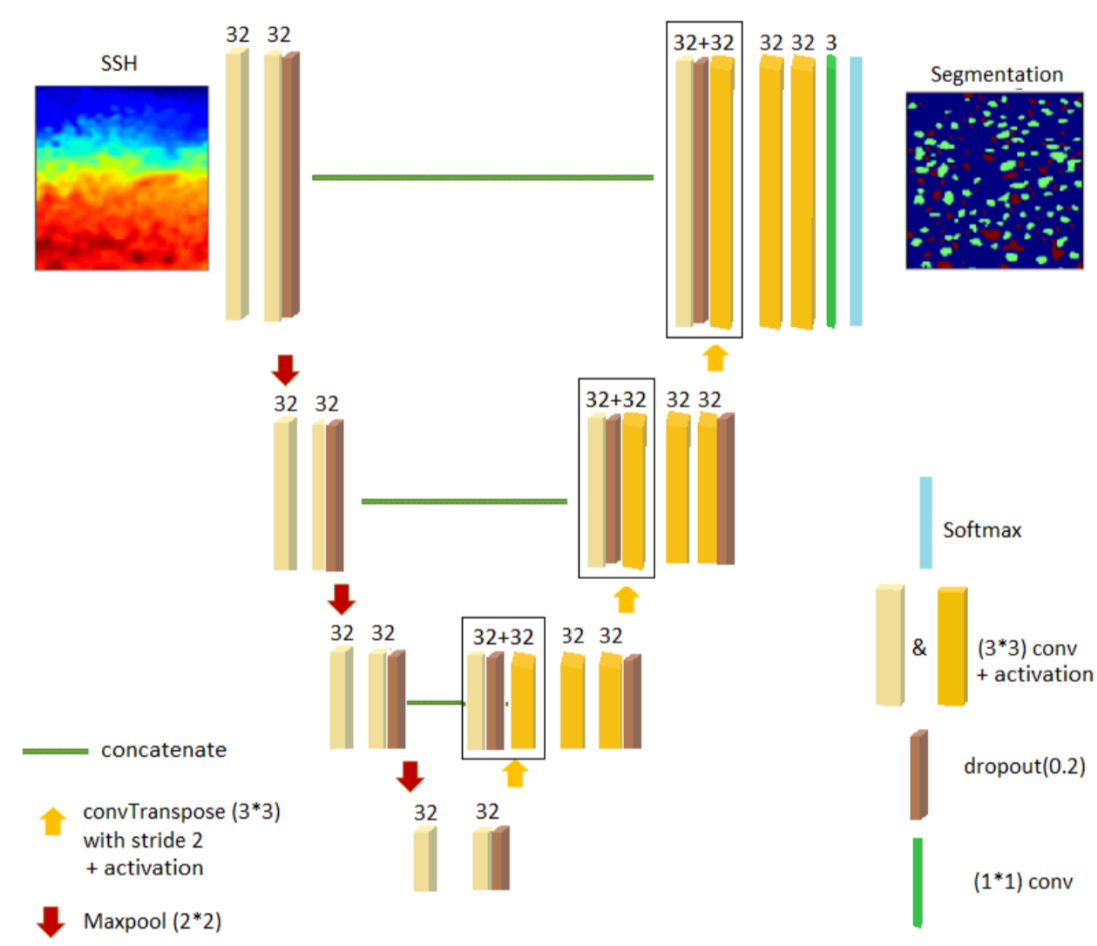
Eddy detection

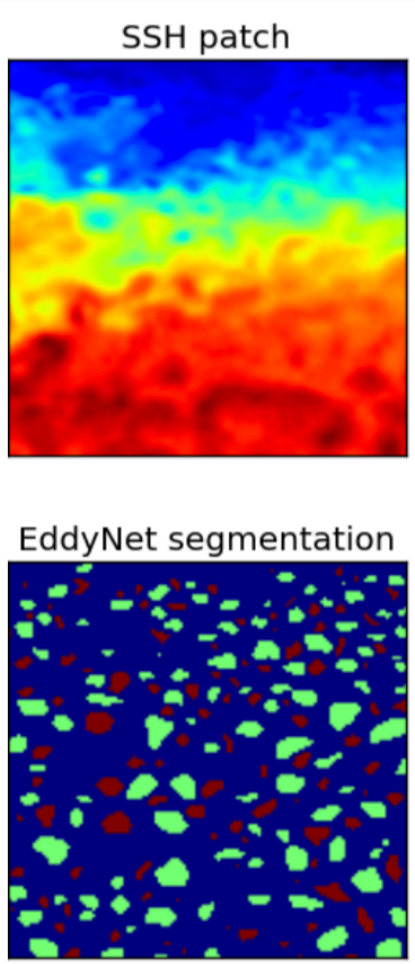
Lguensat et al. (2017)



Eddy detection

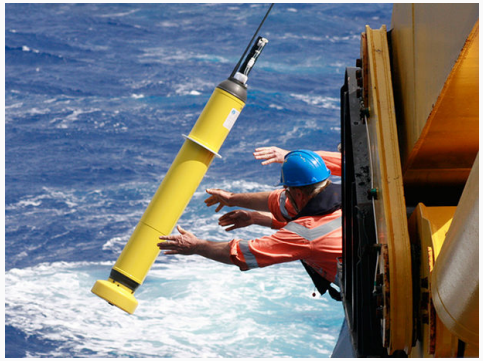
Lguensat et al. (2017)





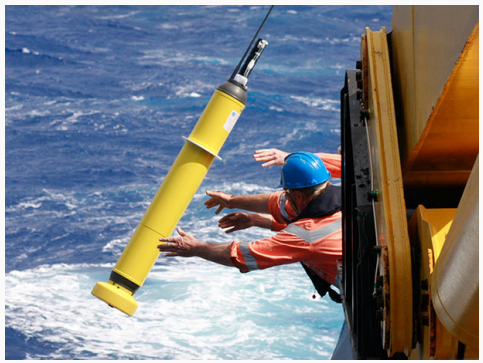
Observations of sea velocity

ARGO floats are deployed in the oceans

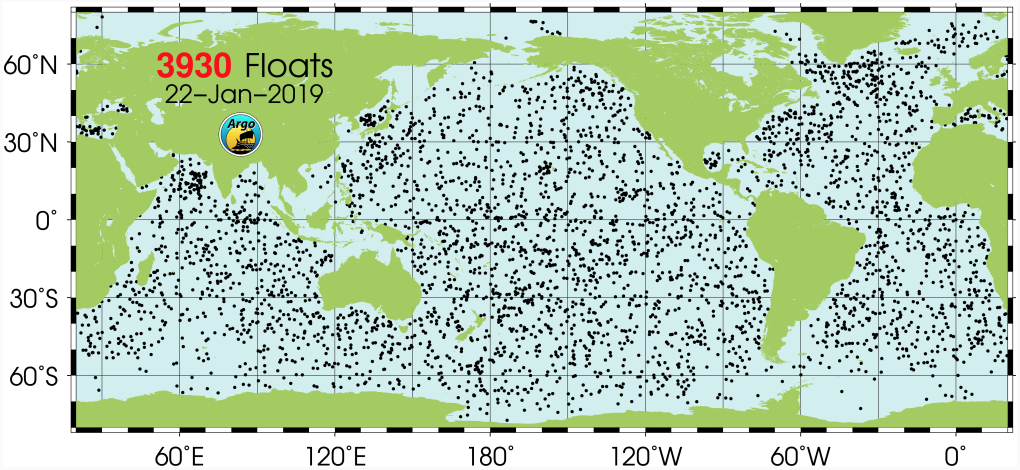


Observations of sea velocity

ARGO floats are deployed in the oceans

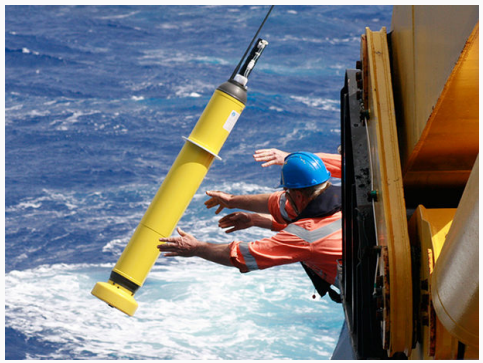


It measures several *in-situ* oceanic parameters

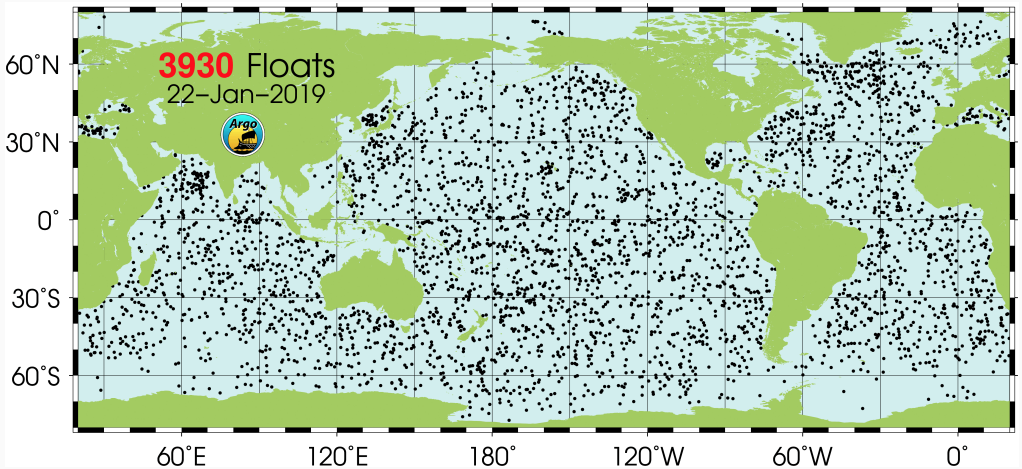


Observations of sea velocity

ARGO floats are deployed in the oceans

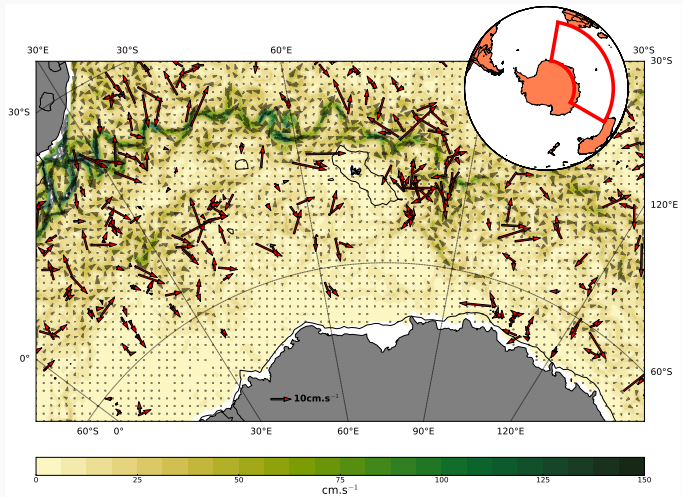


It measures several *in-situ* oceanic parameters



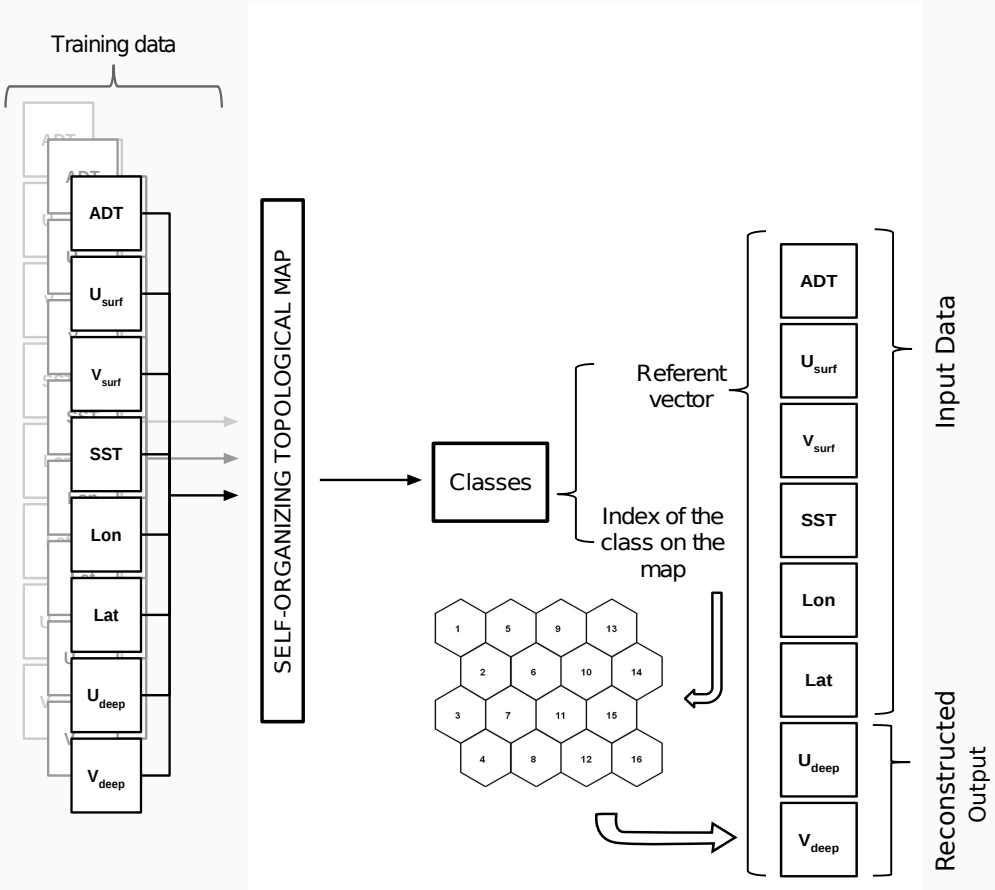
Matching between:

- Sparse *in-situ* velocity at 1000m
- global surface velocity from satellite Sea-Surface heights (SSH).



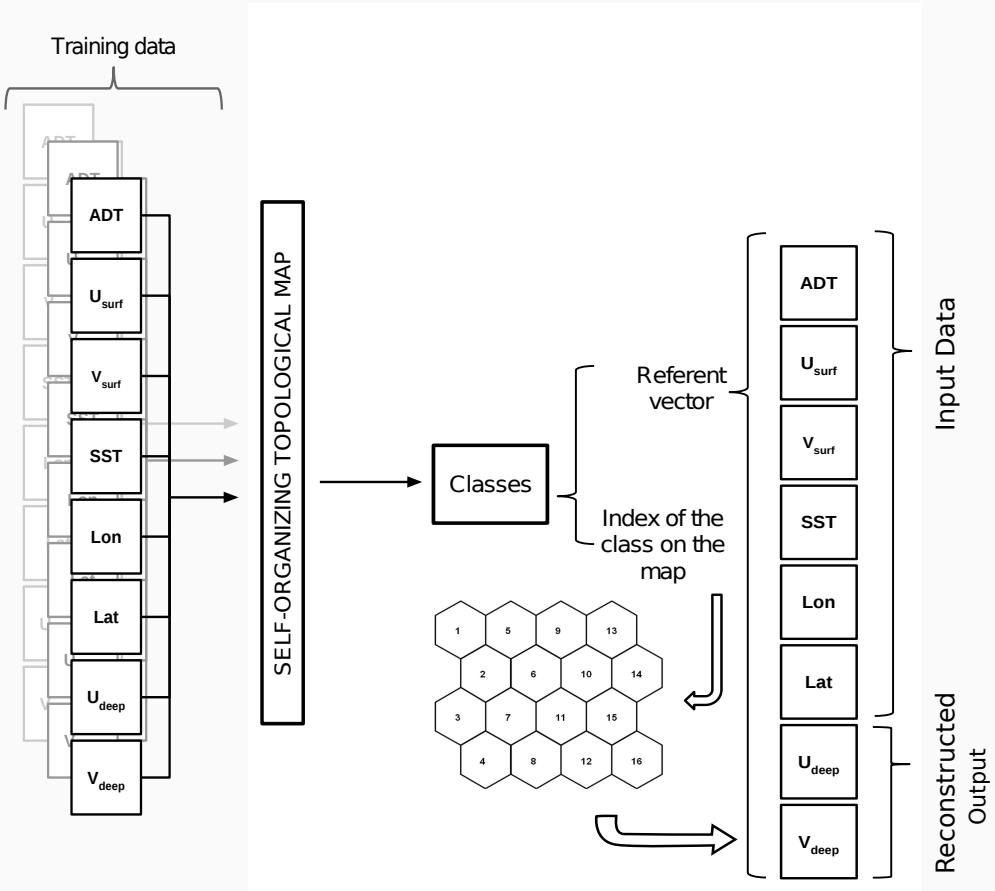
Inference of unknown parameters

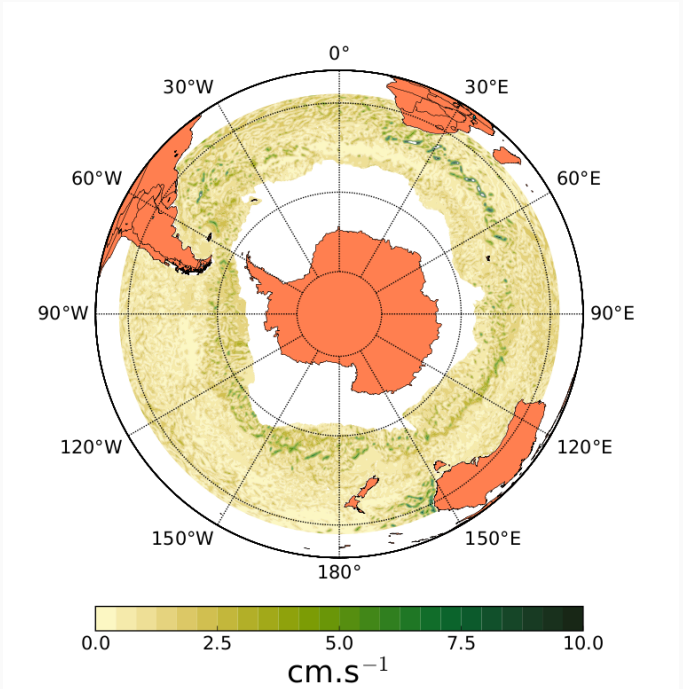
Chapman and Charantonis (2017)

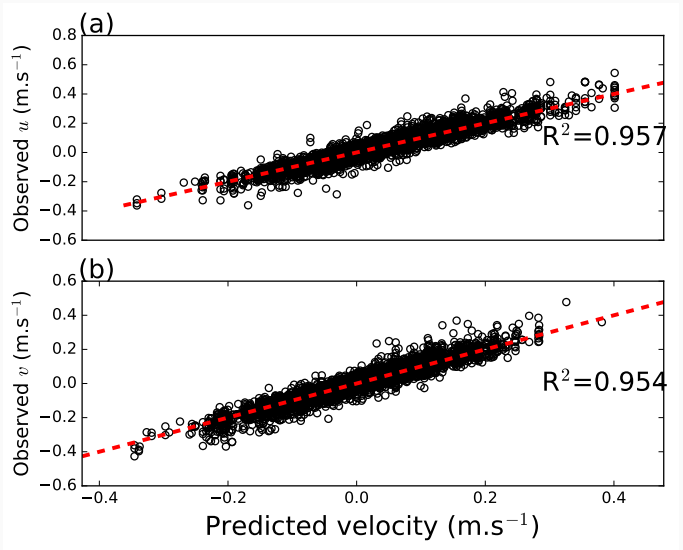


Inference of unknown parameters

Chapman and Charantonis (2017)







Machine learning and differential equations

phyical system and differential equations

For example, Motion fluid can be represented using Navier-Stokes equations (PDE):

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \gamma \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{F}$$

phyical system and differential equations

For example, Motion fluid can be represented using Navier-Stokes equations (PDE):

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \gamma \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{F}$$

After spatial discretization, can be represented as an ODE:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(\mathbf{t}))$$

ResNET and ODE

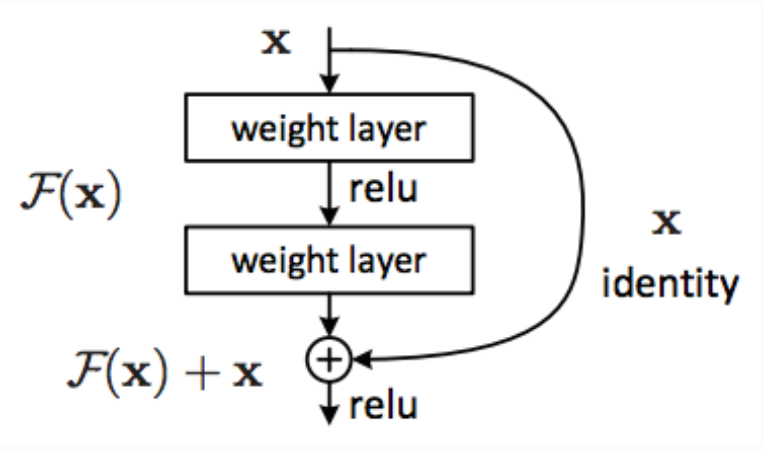
Euler discretization scheme:

$$x(t + \Delta t) = x + \Delta t.f(x(t))$$

ResNET and ODE

Euler discretization scheme:

$$x(t + \Delta t) = x + \Delta t.f(x(t))$$



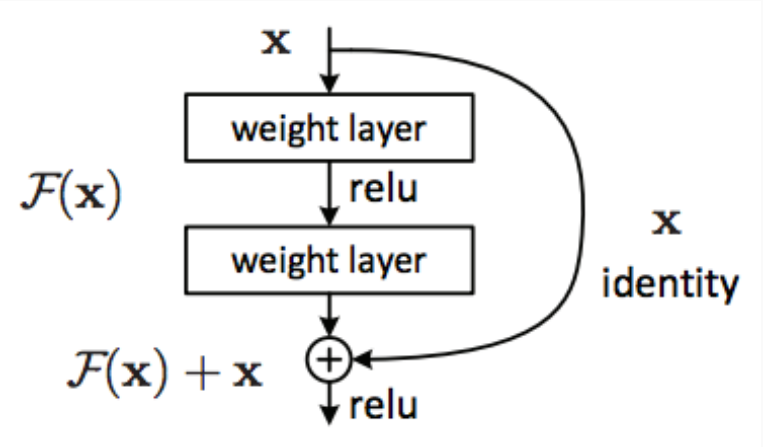
input: $x(t)$
output:$x(t + \Delta t)$

$$x(t + \Delta t) = x(t) + \Delta t.\mathcal{F}(x(t))$$

ResNET and ODE

Euler discretization scheme:

$$x(t + \Delta t) = x + \Delta t.f(x(t))$$



input: $x(t)$
output:$x(t + \Delta t)$

$$x(t + \Delta t) = x(t) + \Delta t.\mathcal{F}(x(t))$$

Training a ResNet is equivalent to training the underlying model in a Euler discretization scheme

Other schemes?

Fablet et al. (2017)

Runge-Kutta (order 4) integration scheme:

$$x(t+\Delta t) = x(t) + \sum_{i=1}^{4} \alpha_i k_i$$

where $k_i = f(x(t+\beta_i k_{i-1} \Delta t))$
$k_0 = 0$, $\alpha_1 = \alpha_4 = 1/6$,
$\alpha_2 = \alpha_3 = 2/6$, $\beta_1 = \beta_4 = 1$ and
$\beta_2 = \beta_3 = 1/2$.

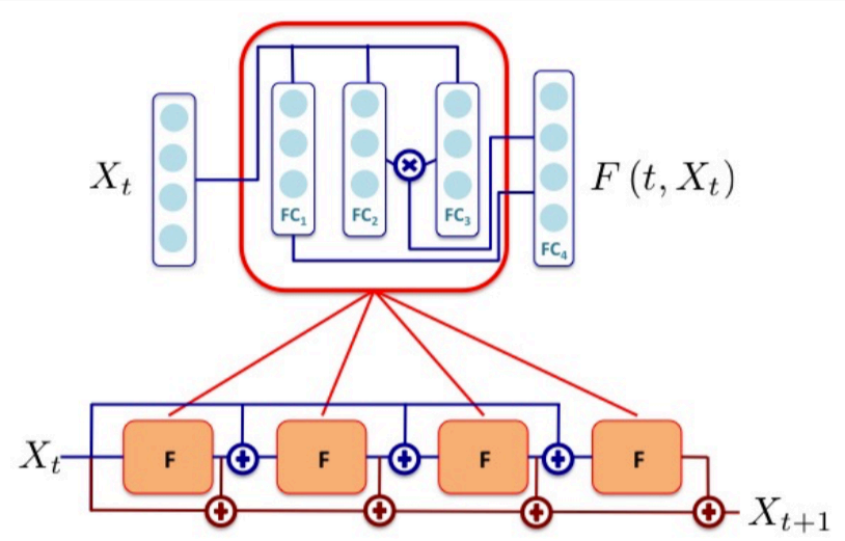
Other schemes?

Fablet et al. (2017)

Runge-Kutta (order 4) integration scheme:

$$x(t+\Delta t) = x(t) + \sum_{i=1}^{4} \alpha_i k_i$$

where $k_i = f(x(t+\beta_i k_{i-1}\Delta t))$
$k_0 = 0$, $\alpha_1 = \alpha_4 = 1/6$,
$\alpha_2 = \alpha_3 = 2/6$, $\beta_1 = \beta_4 = 1$ and
$\beta_2 = \beta_3 = 1/2$.



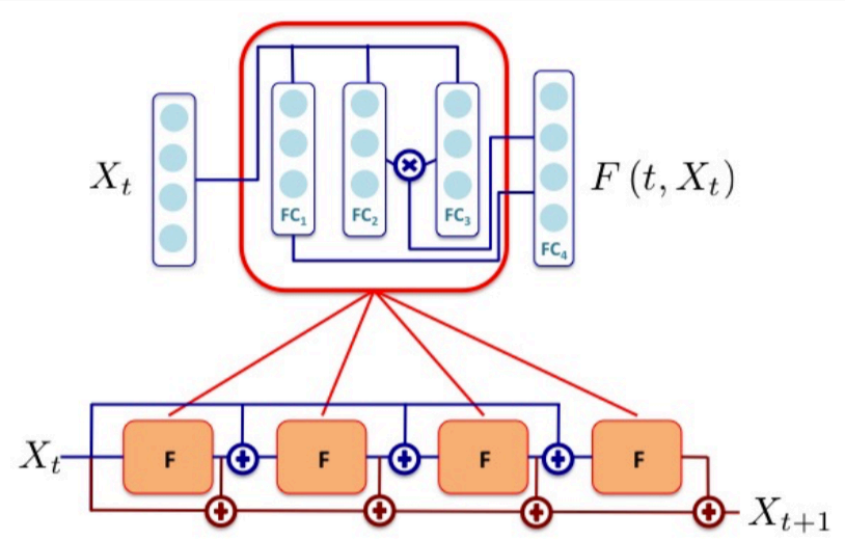
Other schemes?

Fablet et al. (2017)

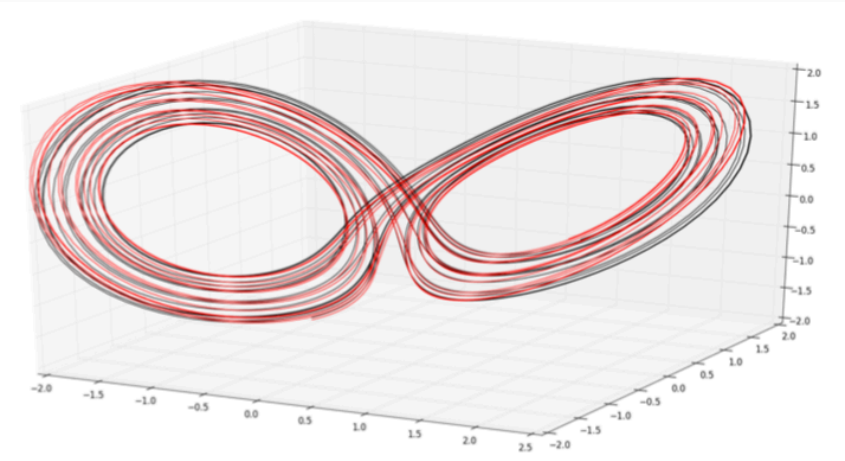
Runge-Kutta (order 4) integration scheme:

$$x(t+\Delta t) = x(t) + \sum_{i=1}^{4} \alpha_i k_i$$

where $k_i = f(x(t+\beta_i k_{i-1}\Delta t))$
$k_0 = 0$, $\alpha_1 = \alpha_4 = 1/6$,
$\alpha_2 = \alpha_3 = 2/6$, $\beta_1 = \beta_4 = 1$ and
$\beta_2 = \beta_3 = 1/2$.



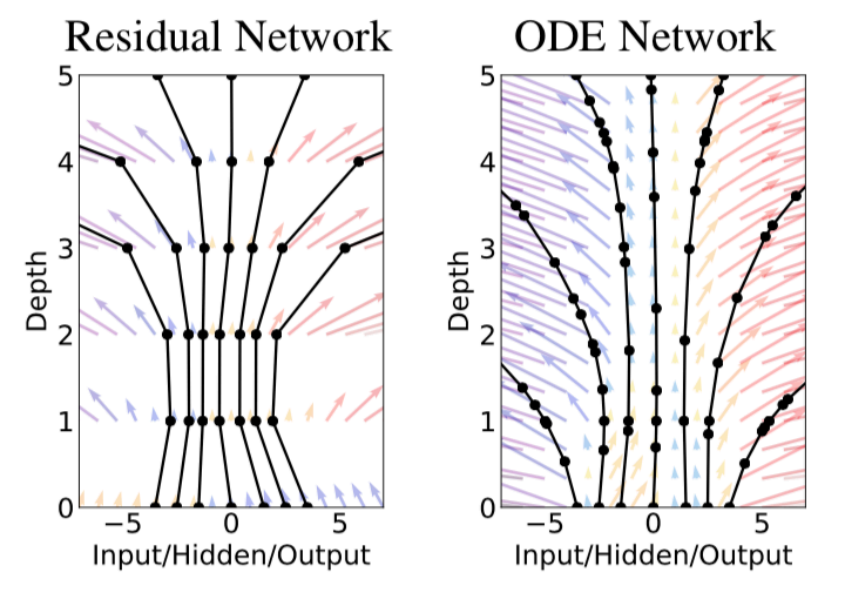
Example on data simulated with a Lorenz 63 system

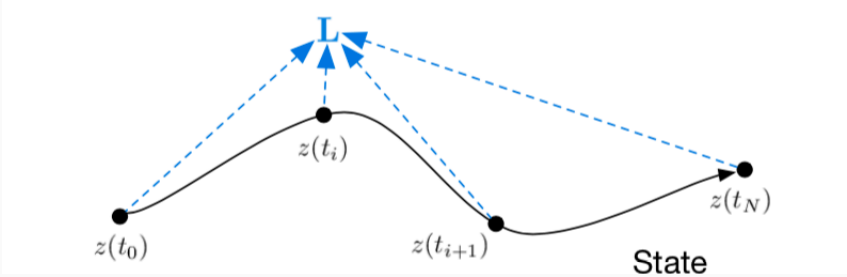


Toward general framework: Neural ODE

Chen et al. (2018): Best paper award NeurIPS, Dec. 2018

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta)\, dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)$$

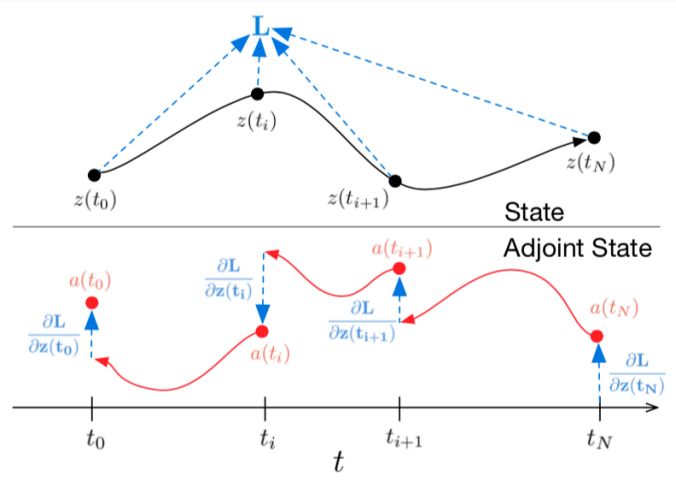




Gradient backpropagation

Objective

Computing $\partial L / \partial \theta$



$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta)\, dt$$

Define the adjoint:

$$\mathbf{a}(t) = \partial L / d\mathbf{z}(t)$$

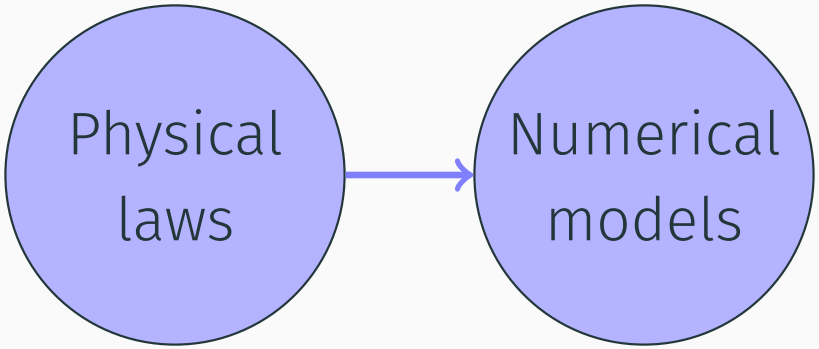
$$\mathbf{a}(t_0) = \mathbf{a}(t_1) - \int_{t_1}^{t_0} a(t)^T \frac{\partial df(\mathbf{z}(\mathbf{t}), \theta)}{\partial \mathbf{z}}\, dt$$

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} a(t)^T \frac{\partial df(\mathbf{z}(\mathbf{t}), \theta)}{\partial \theta}\, dt$$

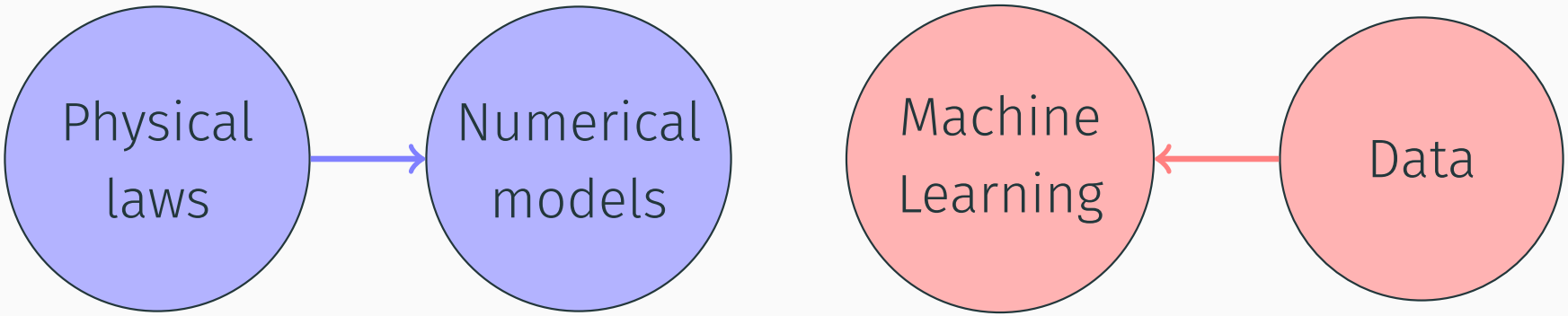
$a(t)$ and $\frac{dL}{d\theta} =$ can be computed using the Same ODE solver backward in time.

Merge approaches: between physics and machine learning

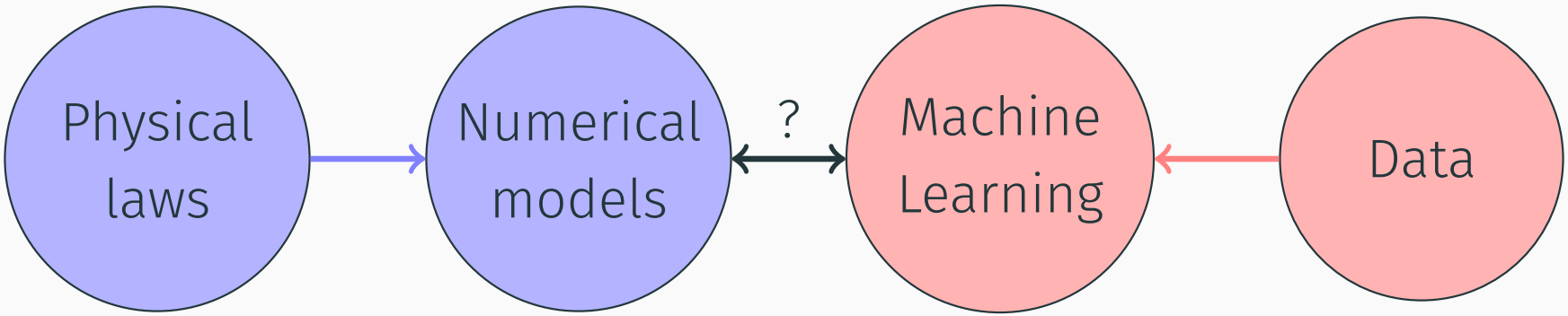
What is this about?



What is this about?



What is this about?

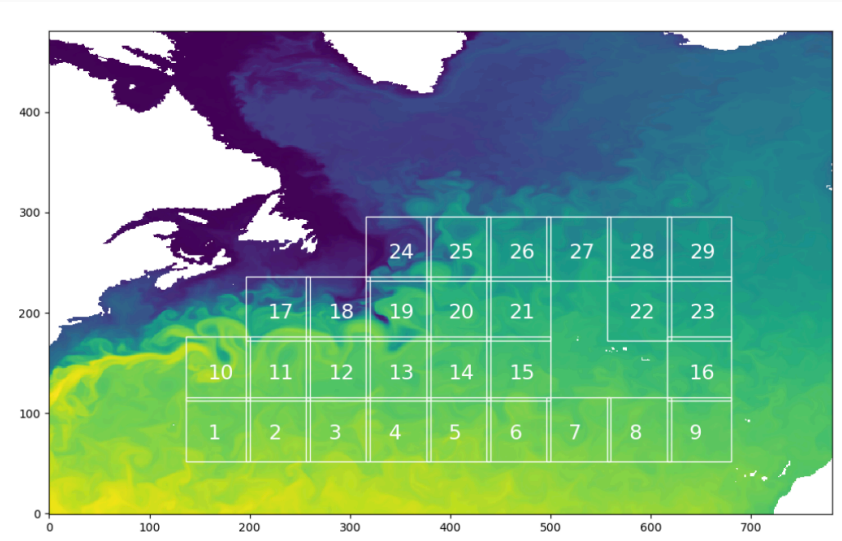


SST dynamic

De Bézenac et al. (2017)

Objective

Given $SST(t-h), \cdots, SST(t)$, predict $SST(t+\tau)$

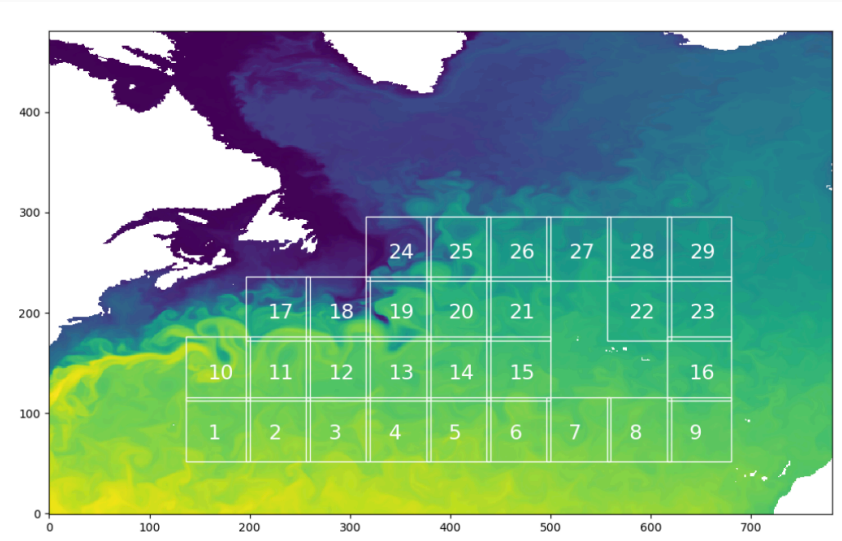


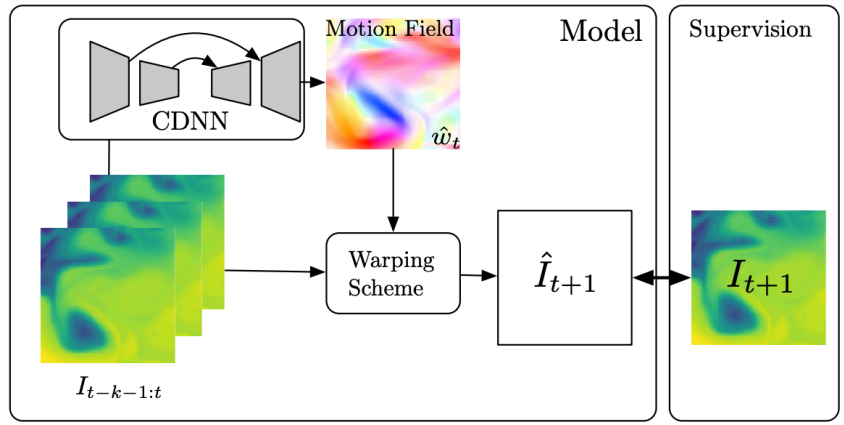
SST dynamic

De Bézenac et al. (2017)

Objective

Given $SST(t-h), \cdots, SST(t)$, predict $SST(t+\tau)$



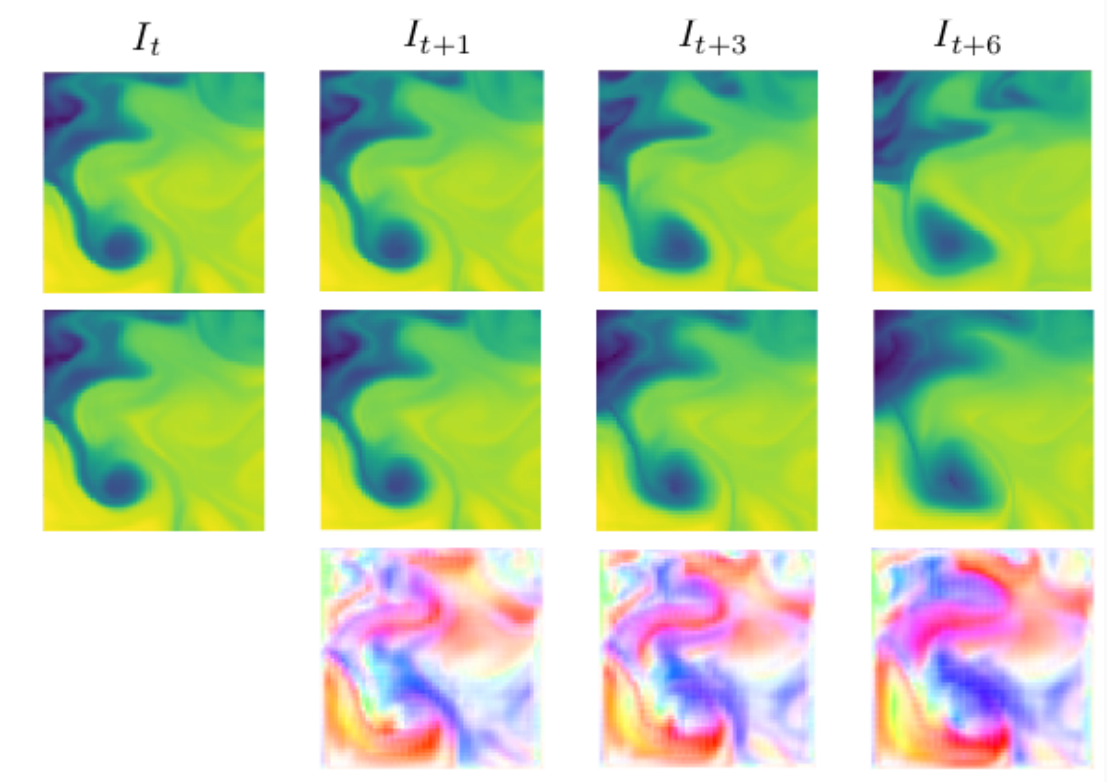


Illustration

Reanalysis (truth)

Deep learning forecast

Motion estimate



Other problem

Using a numerical model $\mathcal{M}$, we can simulate the evolution in time of a state vector (t):

$$\frac{d}{dt} = \mathcal{M}_0((t))$$

Some processes cannot be exactly resolved using only physical principles, because of:

- Imperfect physical knowledge
- Numerical schemes
- Discretization errors
- Unknown factors

In practice, the unresolved/unknown processes can be represented by a θ:

$$\frac{d}{dt} = \mathcal{M}((t), \theta)$$

Determination of θ

$$\frac{d}{dt} = \mathcal{M}((t), \theta(t))$$

θ must be determined to produce a fair simulation.

Determination of θ

$$\frac{d}{dt} = \mathcal{M}((t), \theta(t))$$

θ must be determined to produce a fair simulation. Method: θ is determined using a data-driven approach.

Determination of θ

$$\frac{d}{dt} = \mathcal{M}((t), \theta(t))$$

θ must be determined to produce a fair simulation. Method: θ is determined using a data-driven approach.

Two approaches:

- θ is estimated directly by fitting some observations (Data assimilation)

Determination of θ

$$\frac{d}{dt} = \mathcal{M}((t), \theta(t))$$

θ must be determined to produce a fair simulation. Method: θ is determined using a data-driven approach.

Two approaches:

- θ is estimated directly by fitting some observations (Data assimilation)
- θ is calculated by using an empirical model f:

$$\theta(t) = f((t))$$

Determination of θ

$$\frac{d}{dt} = \mathcal{M}((t), \theta(t))$$

θ must be determined to produce a fair simulation. Method: θ is determined using a data-driven approach.

Two approaches:

- θ is estimated directly by fitting some observations (Data assimilation)
- θ is calculated by using an empirical model f:

$$\theta(t) = f((t))$$

Question:

Can $f((t))$ be represented by a neural network ?

Proof of concept using a shallow-water model

$$
\begin{aligned}
\partial_t u &= +(f+\zeta).v - \partial_x\left(\frac{u^2+v^2}{2} + g^*.h\right) + \theta_u \\
\partial_t v &= -(f+\zeta).u - \partial_y\left(\frac{u^2+v^2}{2} + g^*.h\right) + \theta_v \\
\partial_t h &= -\partial_x(u(H+h)) - \partial_y(v(H+h))
\end{aligned}
$$

Proof of concept using a shallow-water model

$$
\begin{aligned}
\partial_t u &= +(f+\zeta).v - \partial_x(\frac{u^2+v^2}{2} + g^*.h) + \theta_u \\
\partial_t v &= -(f+\zeta).u - \partial_y(\frac{u^2+v^2}{2} + g^*.h) + \theta_v \\
\partial_t h &= -\partial_x(u(H+h)) - \partial_y(v(H+h))
\end{aligned}
$$

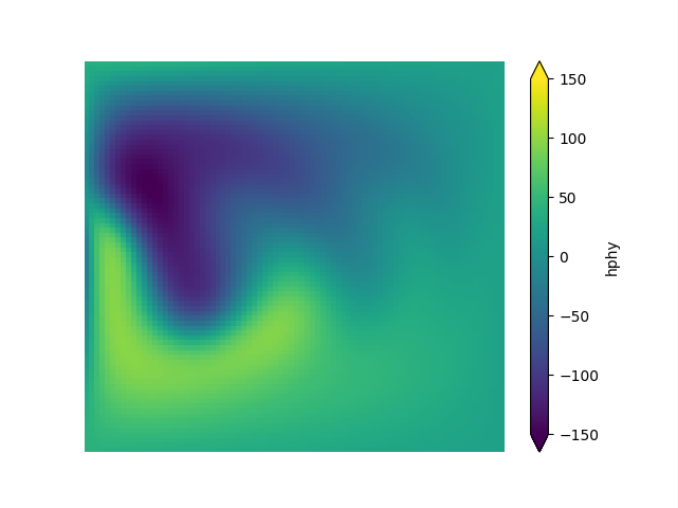
Validation of the approach

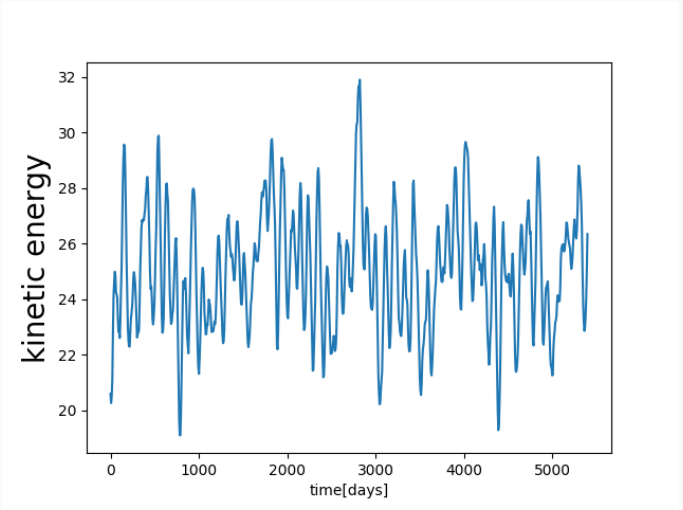
To validate our approach, we use reference synthetic data produced by a fully-specified shallow-water model :

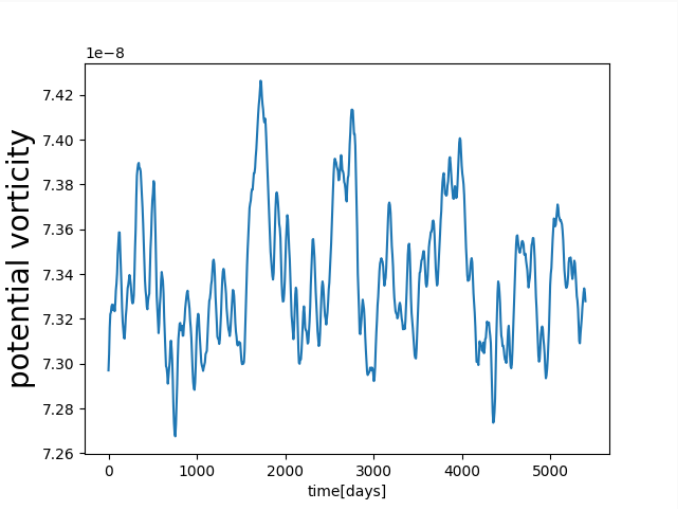
$$
\begin{aligned}
\theta_u &= f_u(u,h,\tau_x) = \frac{\tau_x}{\rho_0(H+h)} - \gamma.u + \nu\Delta u \\
\theta_v &= f_v(v,h,\tau_y) = \frac{\tau_y}{\rho_0(H+h)} - \gamma.v + \nu\Delta v
\end{aligned}
$$

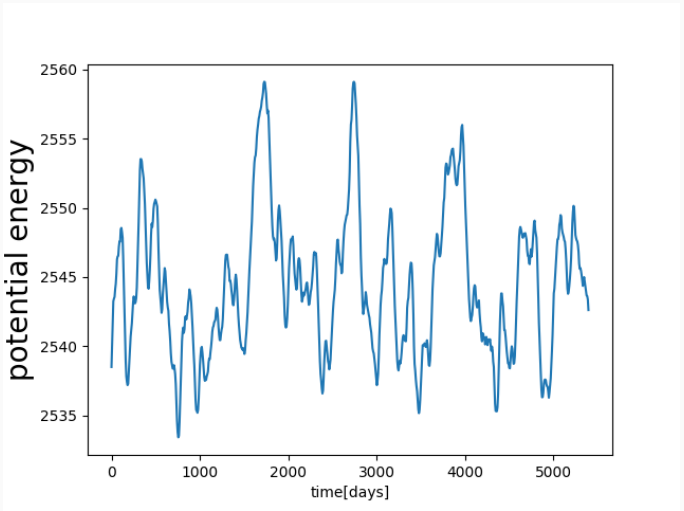
Diagnostic quantities of the reference simulation

mean h



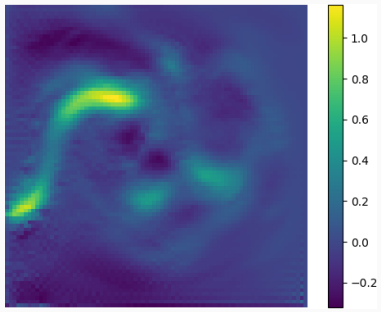
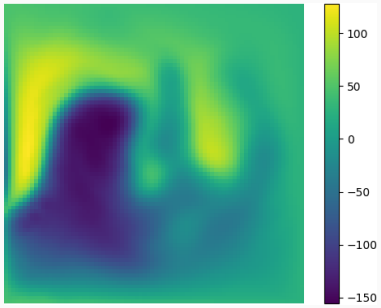
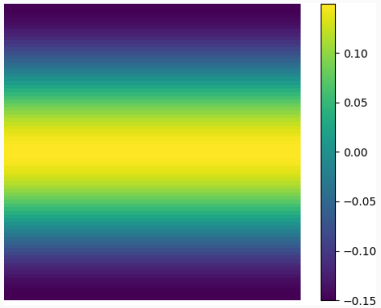
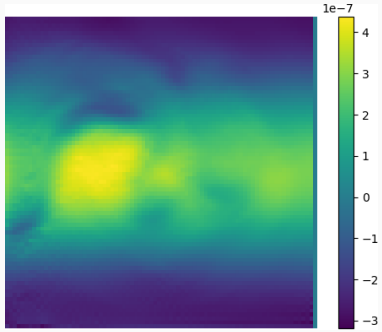




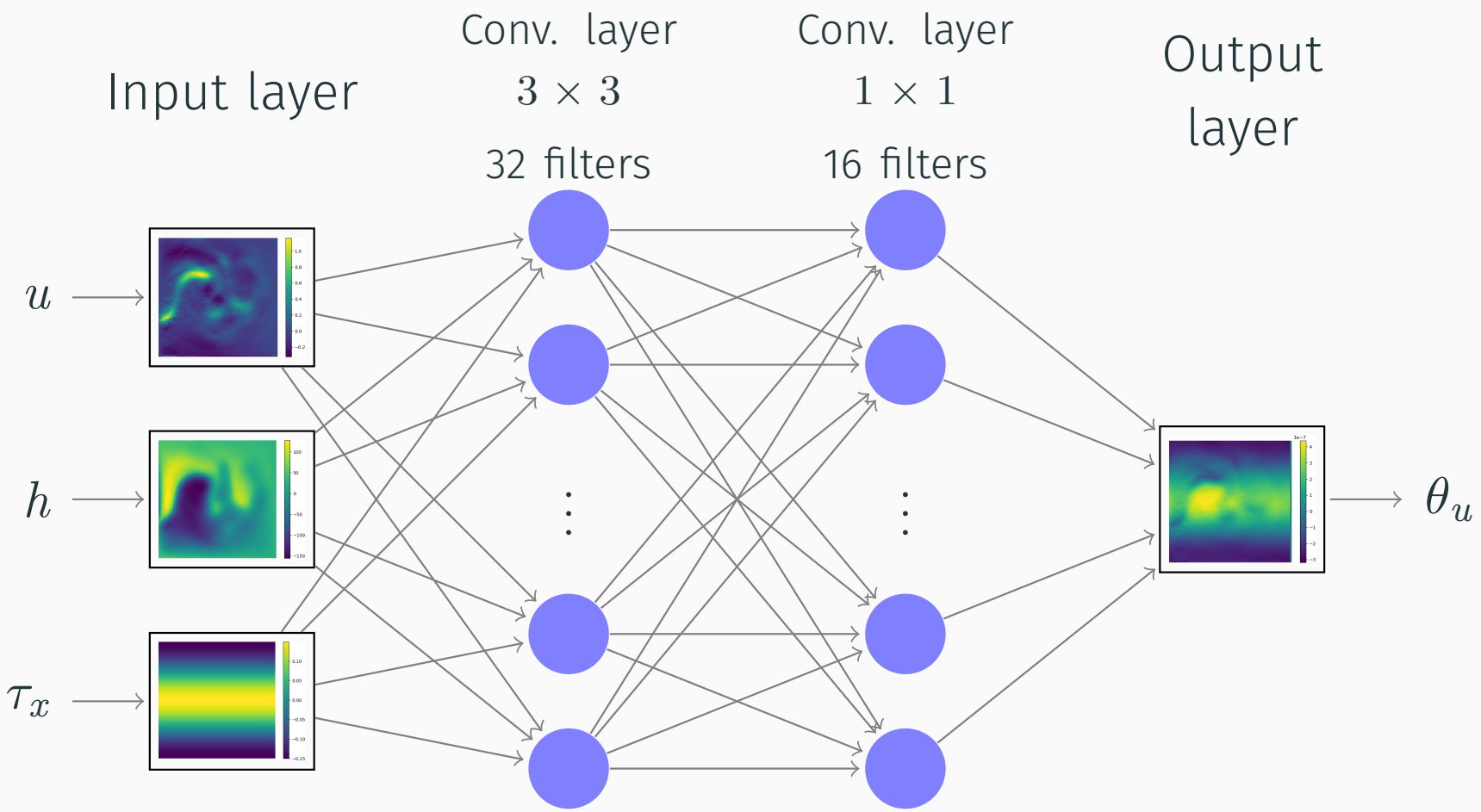


Training dataset

After a simulation of the reference model $\mathcal{M}_T$, we have a set of ~ 300 examples of mapping between $= (u, v, h)$ and θ :

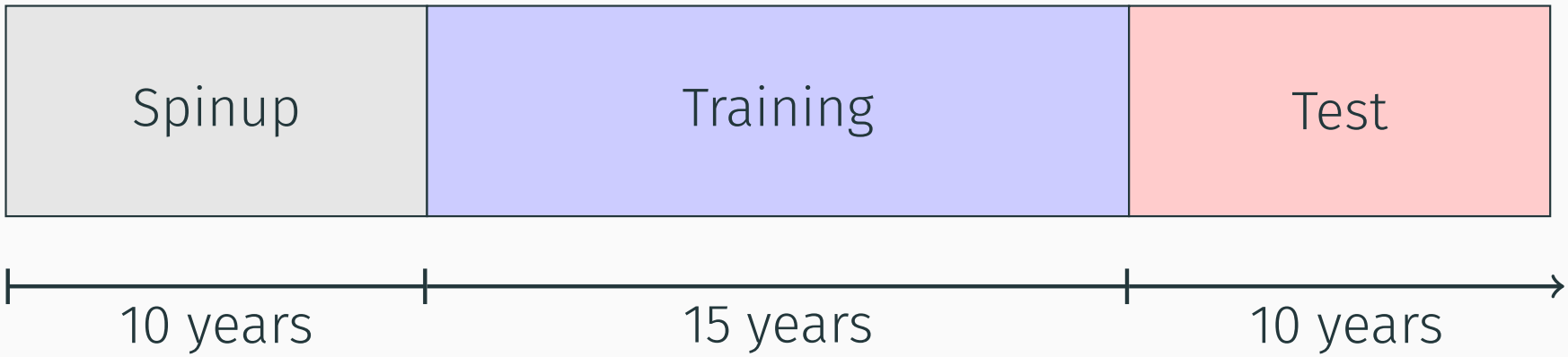
Input			Output
u	h	τ_x	θ_u
			
v	h	τ_y	θ_v

Description of the neural networks



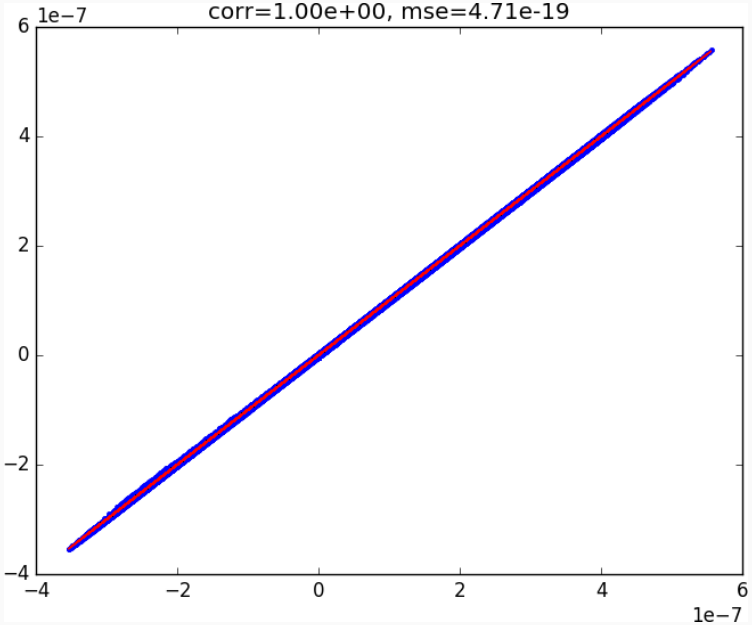
Number of weights to optimize in the neural network : ~ 1700

Spinup/Training/Validation

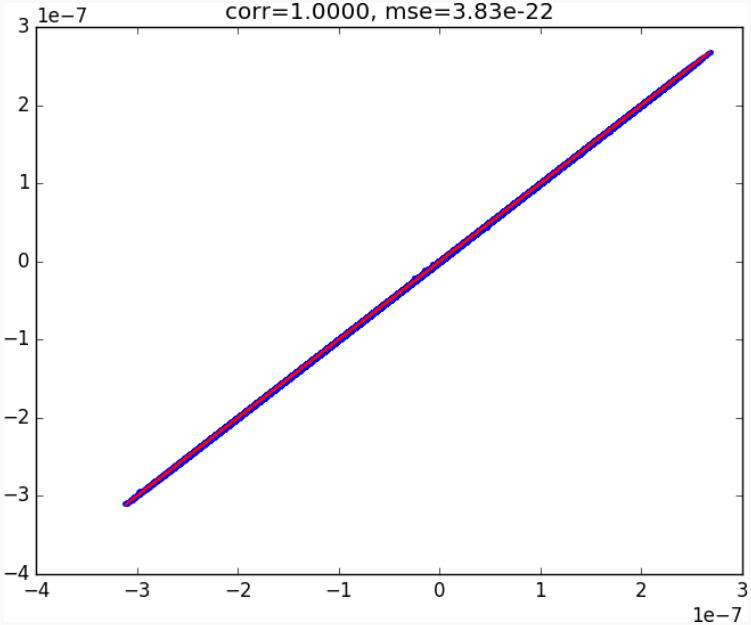


Performance of the neural net

Neural Netwotk output



True θ_u



True θ_v

Description of the neural net simulation

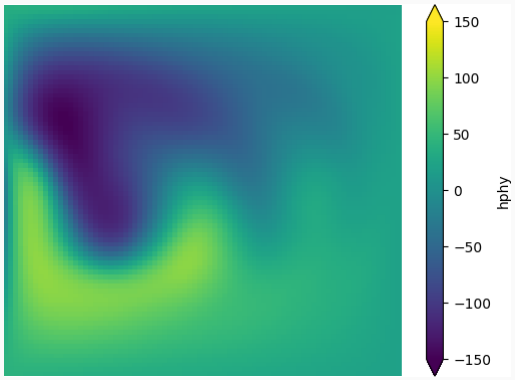
Neural nets are inserted into the shallow-water model:

$$\begin{aligned}
\partial_t u &= +(f+\zeta).v - \partial_x(\frac{u^2+v^2}{2} + g^*.h) + f_u(h, u, \tau_x) \\
\partial_t v &= -(f+\zeta).u - \partial_y(\frac{u^2+v^2}{2} + g^*.h) + f_v(h, v, \tau_y) \\
\partial_t h &= -\partial_x(u(H+h)) - \partial_y(v(H+h))
\end{aligned}$$

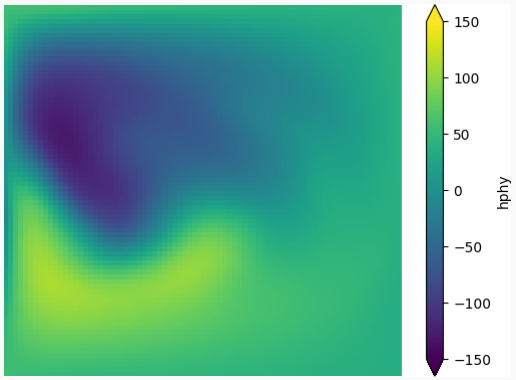
$f_u(h, u, \tau_x)$ and $f_v(h, v, \tau_y)$ are the neural nets used to estimate θ_u and θ_v at each time step.

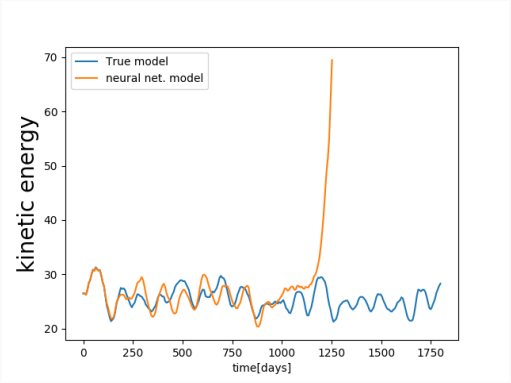
A first simulation

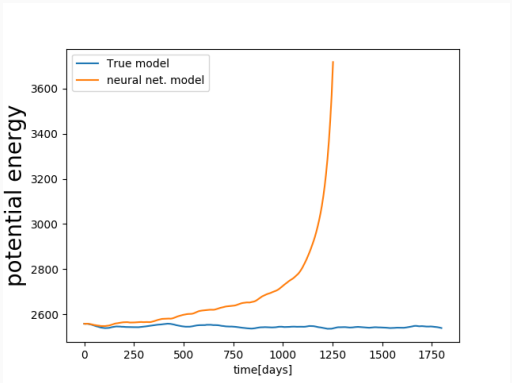
mean h for true model



mean h for nn model

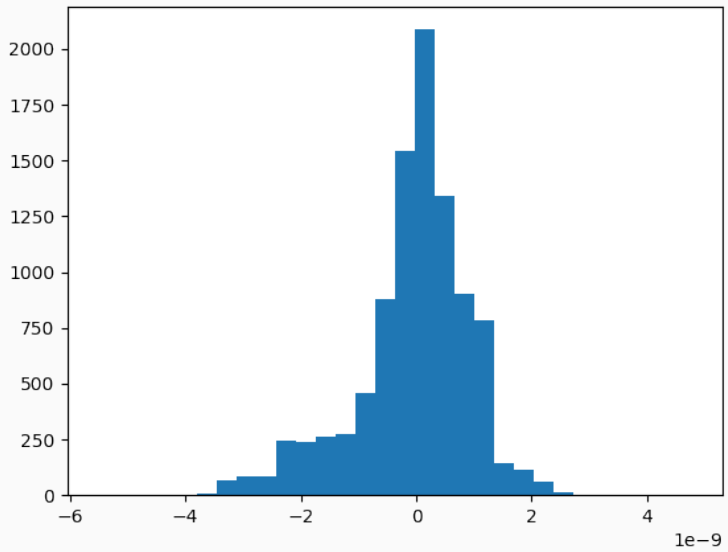






Are the boundary conditions was correctly represented ?

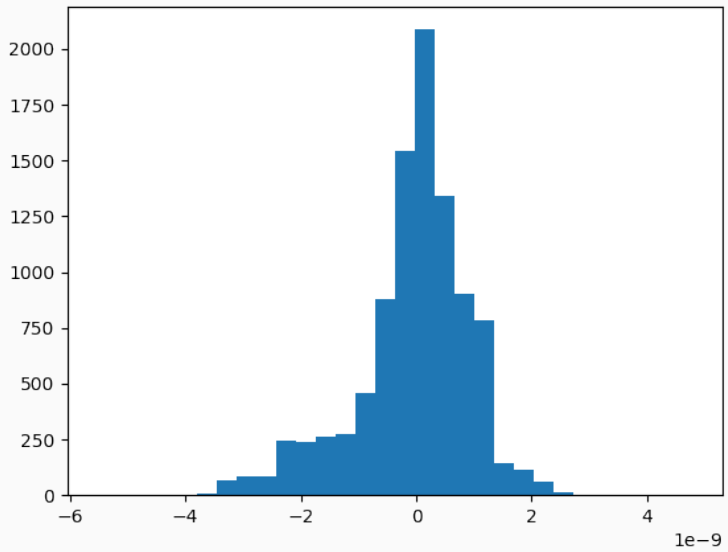
Has the neural net learned $u = 0$ velocity at the East boundary ?



histogram of u values near the Eastearn boundary (RMS = $9.9e-10$)

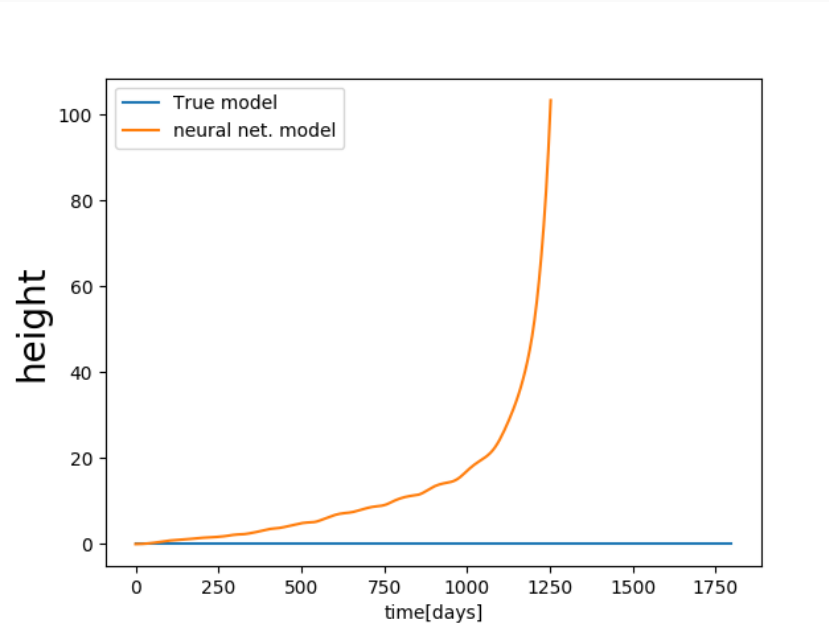
Are the boundary conditions was correctly represented ?

Has the neural net learned $u = 0$ velocity at the East boundary ?



histogram of u values near the Eastearn boundary (RMS = $9.9e-10$)

- Consequence : The NN model is not conservative :

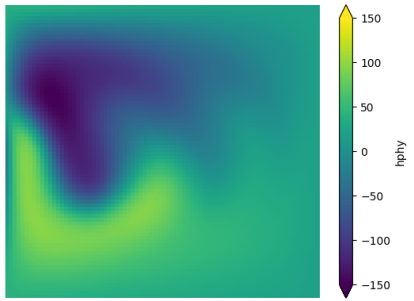


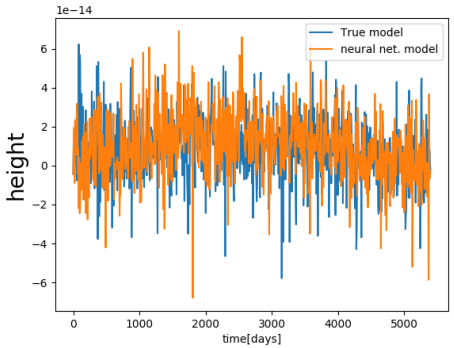
Evolution of mean depth

- The boundary conditions have to be imposed.

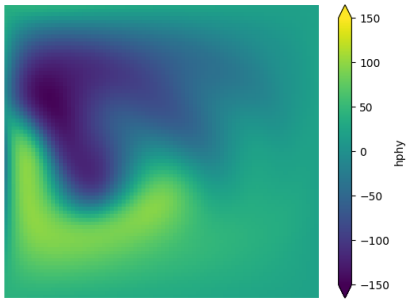
Simulation with imposed boundary conditions

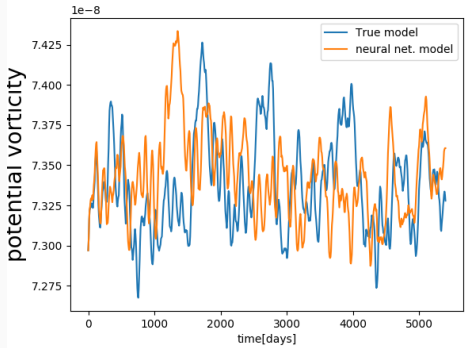
mean h for true model

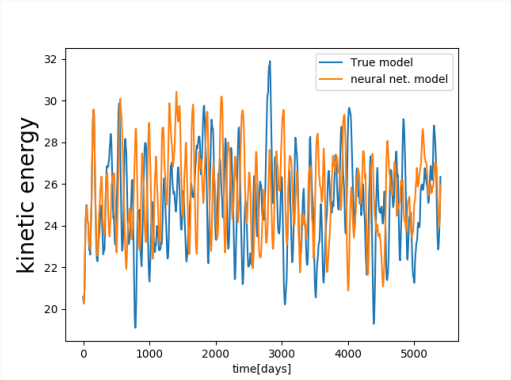


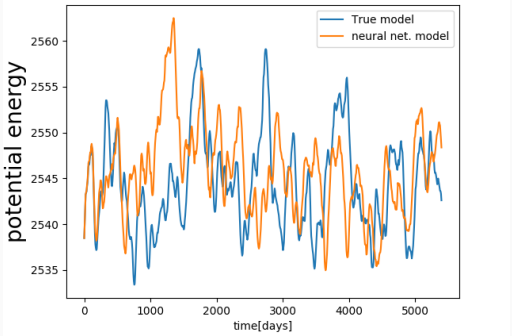


mean h for nn model









Simulation with a different wind forcing

Objective

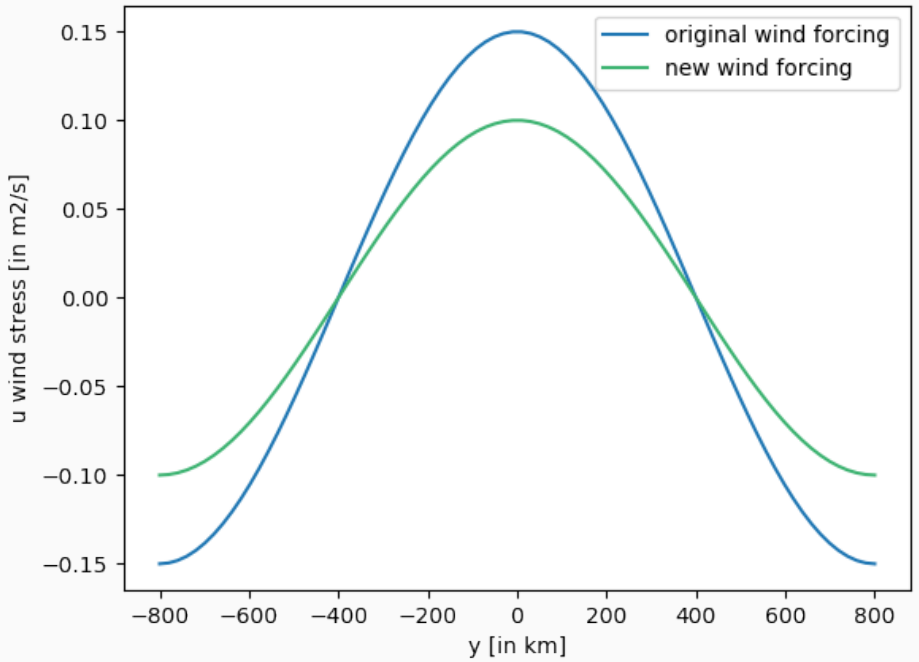
Has the neural net learned systematic features or can it simulate something (relatively) new ?

Simulation with a different wind forcing

Objective

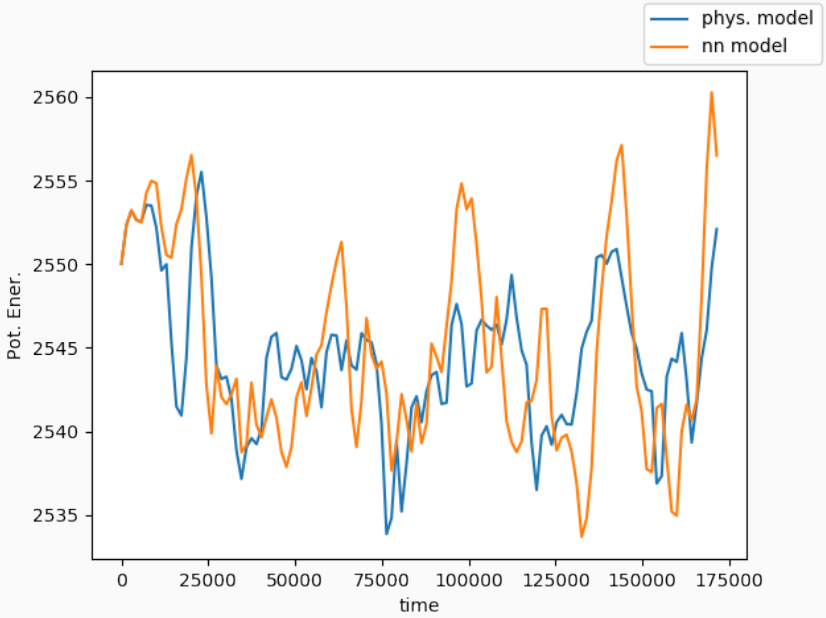
Has the neural net learned systematic features or can it simulate something (relatively) new ?

Idea : Try a neural net simulation with a new wind forcing.

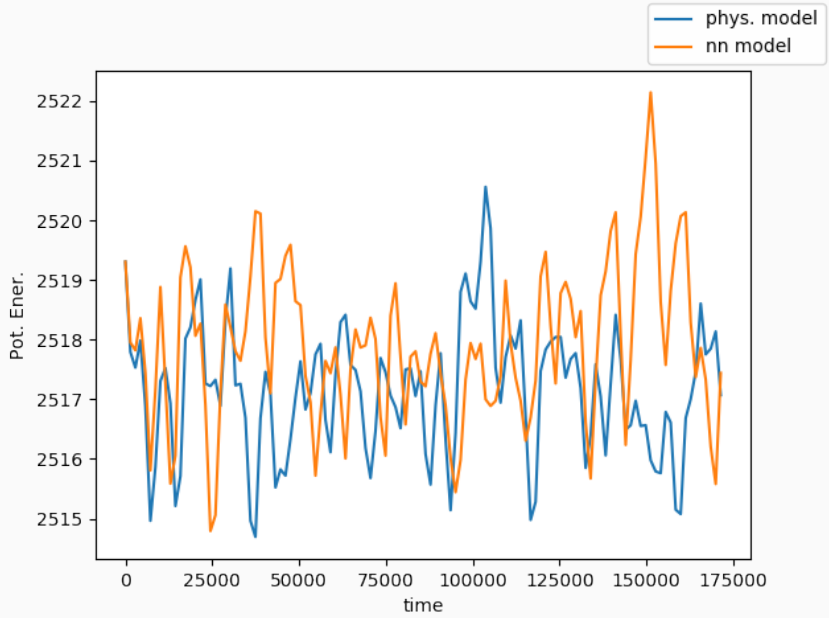


Simulation with a different wind-forcing

With the original wind-forcing

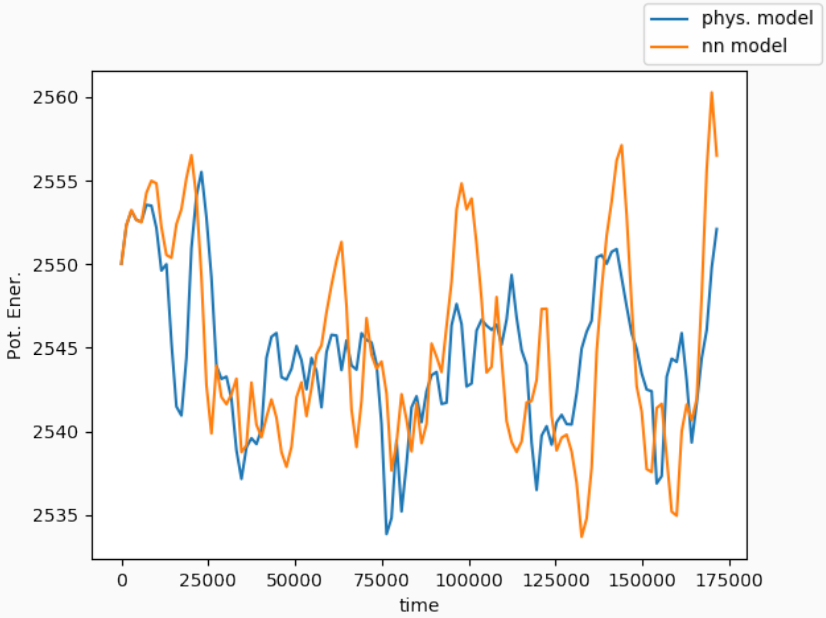


With the low wind-forcing

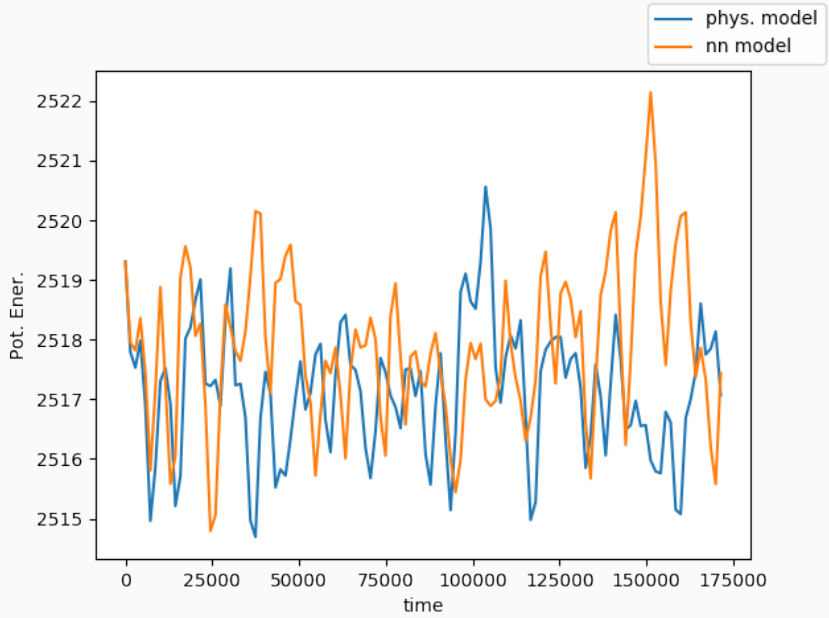


Simulation with a different wind-forcing

With the original wind-forcing



With the low wind-forcing



	ref wind		low wind	
P. E.	2545 ± 2	2545 ± 3	2517 ± 1	2518 ± 1
K. E.	25 ± 1	25 ± 1	11.8 ± 0.3	12.0 ± 0.4
P. V.	7.332 ± 0.01	7.339 ± 0.02	7.159 ± 0.004	7.165 ± 0.005

References

Chapman, C. and Charantonis, A. A. (2017). Reconstruction of Subsurface Velocities From Satellite Observations Using Iterative Self-Organizing Maps. *IEEE Geoscience and Remote Sensing Letters*, 14(5):617–620.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural Ordinary Differential Equations.

De Bézenac, E., Pajot, A., and Lip6, P. G. (2017). Towards a Hybrid Approach to Physical Process Modeling.

Fablet, R., Ouala, S., and Herzet, C. (2017). Bilinear residual Neural Network for the identification and forecasting of dynamical systems.

Lguensat, R., Sun, M., Fablet, R., Mason, E., Tandeo, P., and Chen, G. (2017). EddyNet: A Deep Neural Network For Pixel-Wise Classification of Oceanic Eddies. pages 1–5.