New Perspectives on Balancing Physics with Data-Driven Models: The Case for Physics-Informed Neural Networks in Environmental Statistics

Stefano Castruccio

University of Notre Dame

Research supported by:

Lucy Family Institute for Data & Society

NASA

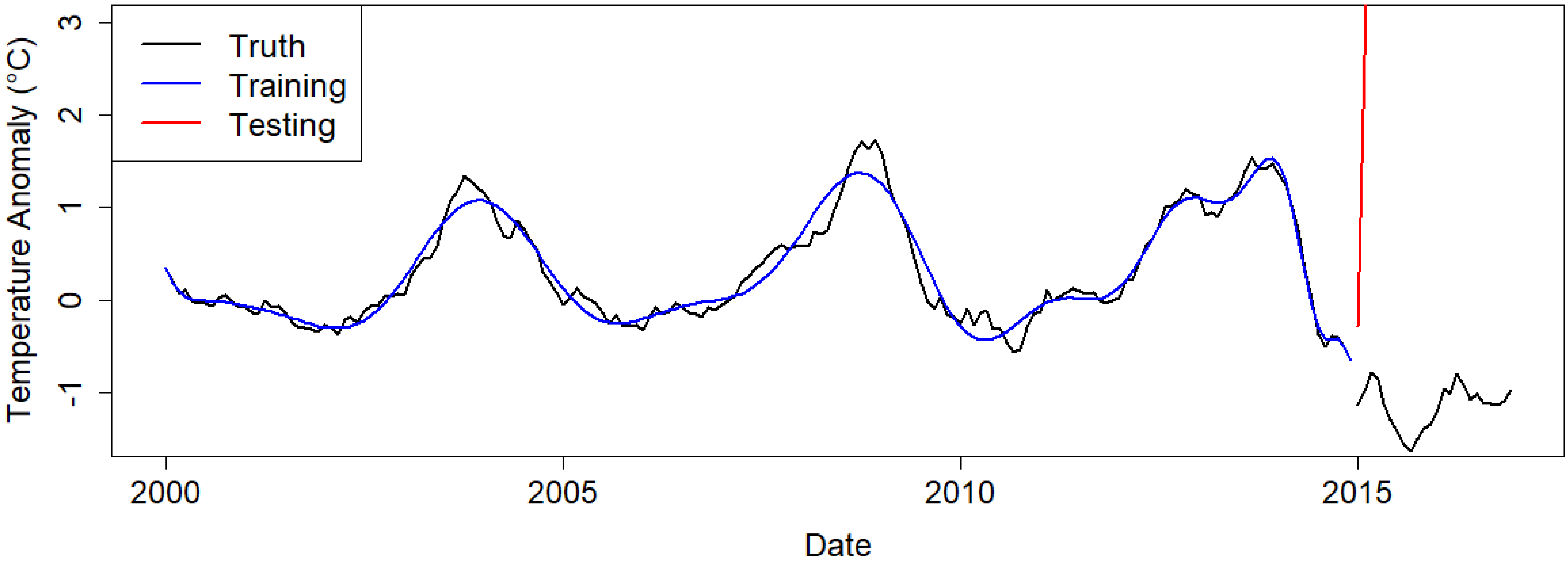
NSF

KAUST

Problem- or data-driven?

- Let’s start with a super-generic question
- **For a particular scientific problem, do we need more context, or should the data “speak for themselves”?**
- Data-driven: rely on **non-parametric constructs (neural networks, NNs)** have been very popular for many applications in STEM and beyond
- Shall we just ignore the context?

Why data driven methods could fail

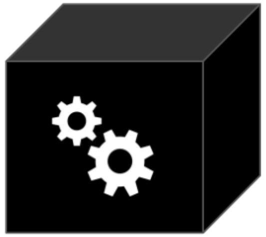


$$\frac{\partial W}{\partial t} + \nabla \cdot Q = E - P$$

$$F_\lambda = \int_{l_{West}} Q_{\lambda_{West}} \cdot dl \text{ and } F_\phi = \int_{l_{East}} Q_{\phi_{East}} \cdot dl$$

$$\frac{\partial \rho F^{(x)}}{\partial x} + \frac{\partial \rho F^{(y)}}{\partial y} = E - \rho P$$

A fork in the road?



- Well known equations
- Few data (hard/expensive)
- E.g., astronomy, paleoclimate

- Unknown or nonexistent equations
- A lot of data (cheap and ubiquitous)
- E.g., social networks, large language models, foundation models

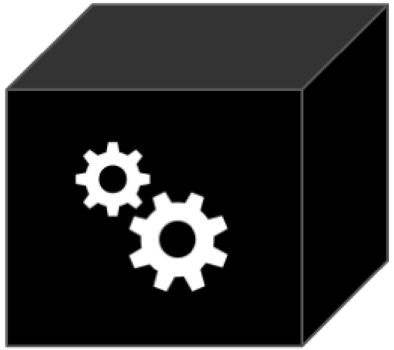
Do we even have to choose?

- The problem of merging (assimilating, fusing, etc.) data with physical information is a very old exploit in Statistics
- These models are generally termed **physical-statistical** (PS) models
- Mark Berliner's 2003 JRG paper was the first one to formalize it (to my knowledge!)

Spatio-temporal statistical models are not at odds with deterministic ones. Indeed, the most powerful models are constructed based on physical mechanisms

(Wikle, Zammit-Magion and Cressie, Spatio-temporal statistics with R)

How PS models work



$$\frac{\partial W}{\partial t} + \nabla \cdot Q = E - P$$

$$F_\lambda = \int_{l_{West}} Q_{\lambda_{West}} \cdot dl \text{ and } F_\phi = \int_{l_{East}} Q_{\phi_{East}} \cdot dl$$

$$\frac{\partial \rho F^{(x)}}{\partial x} + \frac{\partial \rho F^{(y)}}{\partial y} = E - \rho P$$

PS models

- PS model fits naturally in a hierarchical (Bayesian) framework
- Stage 1 (data): Suppose we observe $Z_t(\boldsymbol{s})$, we assume

$$Z_t(\boldsymbol{s}) = \mathcal{H}\left(Y_t(\boldsymbol{s}), \boldsymbol{\theta}^{\text{data}}, \varepsilon_t(\boldsymbol{s})\right)$$

- $Y_t(\boldsymbol{s})$: latent process
- $\mathcal{H}$: (linear/nonlinear) mapping
- $\boldsymbol{\theta}^{\text{data}}$: data-model parameters
- $\varepsilon_t(\boldsymbol{s})$: data-model error
- Easy example $Z_t(\boldsymbol{s}) \sim N\left(Y_t(\boldsymbol{s}), \sigma^2\right)$

PS models

- Stage 2: The latent process $Y_t(\boldsymbol{s})$ is modeled

$$Y_t(\boldsymbol{s}) = \mathcal{M}(\{Y_{t-1}(\boldsymbol{s}), \ldots, Y_{t-k}(\boldsymbol{s})\}, \boldsymbol{\theta}^{\text{process}}, \eta_t(\boldsymbol{s}))$$

- $Y_t(\boldsymbol{s})$: latent process
- $\mathcal{M}$: dynamic process
- $\boldsymbol{\theta}^{\text{process}}$: data-model parameters
- $\eta_t(\boldsymbol{s})$: process-model error
- (Stage 3: Priors)

$$r: \quad \rho\left(\frac{\partial u_r}{\partial t} + u_r\frac{\partial u_r}{\partial r} + \frac{u_\phi}{r\sin(\theta)}\frac{\partial u_r}{\partial \phi} + \frac{u_\theta}{r}\frac{\partial u_r}{\partial \theta} - \frac{u_\phi^2 + u_\theta^2}{r}\right) = -\frac{\partial p}{\partial r} + \rho g_r +$$

$$\mu\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial u_r}{\partial r}\right) + \frac{1}{r^2\sin(\theta)^2}\frac{\partial^2 u_r}{\partial \phi^2} + \frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial \theta}\left(\sin(\theta)\frac{\partial u_r}{\partial \theta}\right) - 2\frac{u_r + \frac{\partial u_\theta}{\partial \theta} + u_\theta\cot(\theta)}{r^2} - \frac{2}{r^2\sin(\theta)}\frac{\partial u_\phi}{\partial \phi}\right]$$

$$\phi: \quad \rho\left(\frac{\partial u_\phi}{\partial t} + u_r\frac{\partial u_\phi}{\partial r} + \frac{u_\phi}{r\sin(\theta)}\frac{\partial u_\phi}{\partial \phi} + \frac{u_\theta}{r}\frac{\partial u_\phi}{\partial \theta} + \frac{u_r u_\phi + u_\phi u_\theta\cot(\theta)}{r}\right) = -\frac{1}{r\sin(\theta)}\frac{\partial p}{\partial \phi} + \rho g_\phi +$$

$$\mu\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial u_\phi}{\partial r}\right) + \frac{1}{r^2\sin(\theta)^2}\frac{\partial^2 u_\phi}{\partial \phi^2} + \frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial \theta}\left(\sin(\theta)\frac{\partial u_\phi}{\partial \theta}\right) + \frac{2\sin(\theta)\frac{\partial u_r}{\partial \phi} + 2\cos(\theta)\frac{\partial u_\theta}{\partial \phi} - u_\phi}{r^2\sin(\theta)^2}\right]$$

$$\theta: \quad \rho\left(\frac{\partial u_\theta}{\partial t} + u_r\frac{\partial u_\theta}{\partial r} + \frac{u_\phi}{r\sin(\theta)}\frac{\partial u_\theta}{\partial \phi} + \frac{u_\theta}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{u_r u_\theta - u_\phi^2\cot(\theta)}{r}\right) = -\frac{1}{r}\frac{\partial p}{\partial \theta} + \rho g_\theta +$$

$$\mu\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial u_\theta}{\partial r}\right) + \frac{1}{r^2\sin(\theta)^2}\frac{\partial^2 u_\theta}{\partial \phi^2} + \frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial \theta}\left(\sin(\theta)\frac{\partial u_\theta}{\partial \theta}\right) + \frac{2}{r^2}\frac{\partial u_r}{\partial \theta} - \frac{u_\theta + 2\cos(\theta)\frac{\partial u_\phi}{\partial \phi}}{r^2\sin(\theta)^2}\right].$$

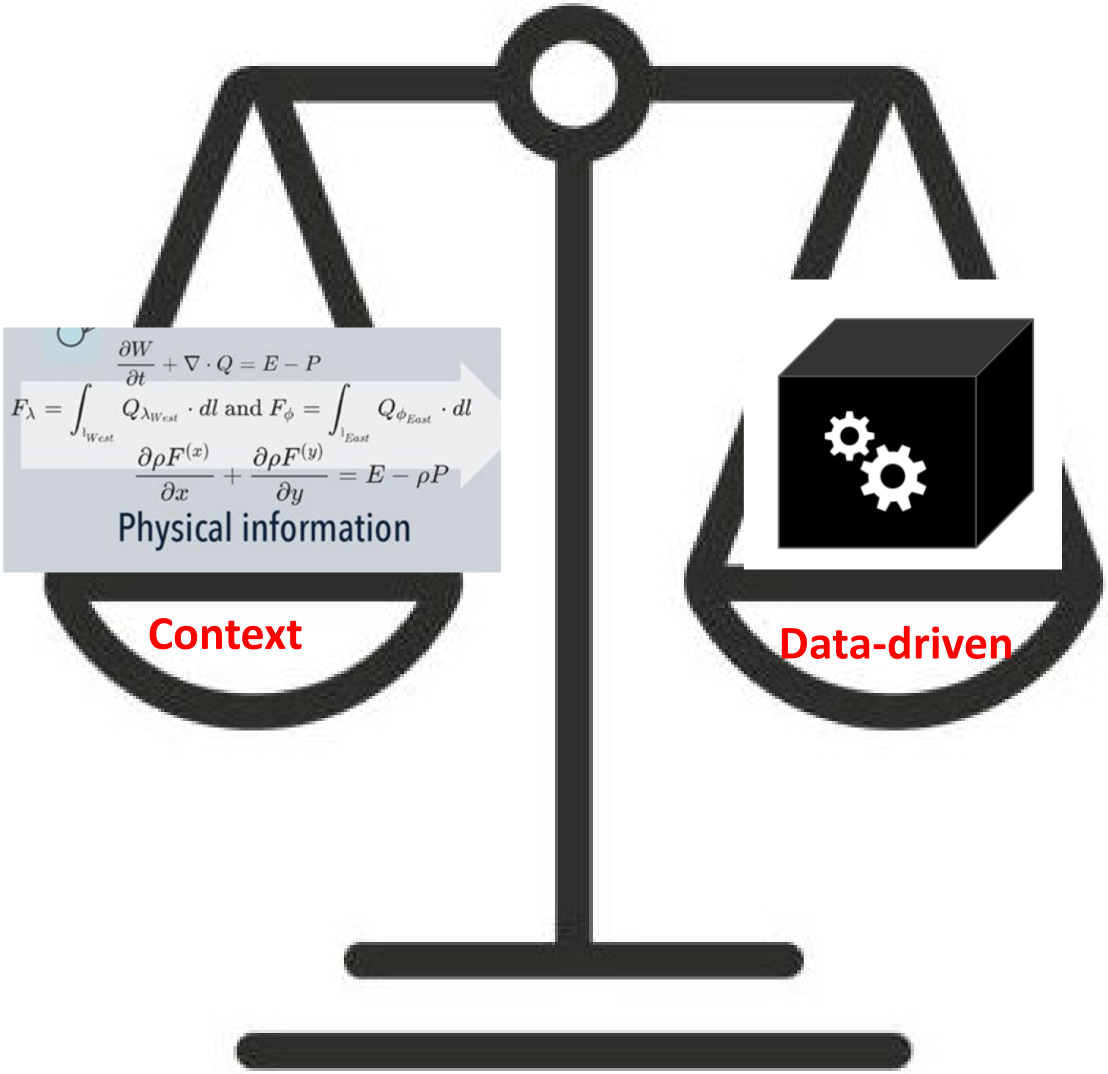
Relative merits of PS models

- PS models
$$Y_t(\boldsymbol{s}) = \mathcal{M}(\{Y_{t-1}(\boldsymbol{s}), \ldots, Y_{t-k}(\boldsymbol{s})\}, \boldsymbol{\theta}^{\text{process}}, \eta_t(\boldsymbol{s}))$$
- PS models require a definition of $\mathcal{M}$, so an **exact knowledge of the physics**
- If something goes wrong, we hope that $\eta_t(\boldsymbol{s})$ takes case of any misspecification

Physics-informed Neural networks

- In the last few years, the ML literature has developed a separate area of research on how to merge physics with data: **physics informed neural networks (PINNs)**
- The idea is quite different, but there is a link with PS models which was somehow lost

The main idea: a scale



PINNs in a nutshell

- **This is my “stats” reframing of PINN**
- The canonical PINN definition is more algorithmical
- Stage 1: We observe $Z_t(\boldsymbol{s})$ and we assume
$$Z_t(\boldsymbol{s}) = \mathcal{H}\left(Y_t(\boldsymbol{s}), \boldsymbol{\theta}^{\text{data}}, \varepsilon_t(\boldsymbol{s})\right)$$
- $Y_t(\boldsymbol{s})$: latent process
- $\mathcal{H}$: (linear/nonlinear) mapping
- $\boldsymbol{\theta}^{\text{data}}$: data-model parameters
- $\varepsilon_t(\boldsymbol{s})$: data-model error (noise)

PINNs as hierarchical models

- Stage 2 (process): The latent process $Y_t(\boldsymbol{s})$ is modeled
$$Y_t(\boldsymbol{s}) = \mathcal{M}(\{Y_{t-1}(\boldsymbol{s}), \ldots, Y_{t-k}(\boldsymbol{s})\}, \boldsymbol{\theta}^{\text{process}}, \eta_t(\boldsymbol{s}))$$
- $Y_t(\boldsymbol{s})$: latent process
- $\mathcal{M}$: dynamic process, **highly nonparametric NN**
- $\boldsymbol{\theta}^{\text{process}}$: data-model parameters (**huge space!**)
- $\eta_t(\boldsymbol{s})$: process-model error (noise)
- (Stage 3: Priors)

The main idea behind PINNs

- Assume $\boldsymbol{Y}_{\mathrm{t}} = \{Y_t(\boldsymbol{s})\}$ is informed by a PDE

$$\frac{\partial \boldsymbol{Y}_{\mathrm{t}}}{\partial t} - \mathcal{N}[\boldsymbol{Y}_{\mathrm{t}}] = 0$$

- **We are <u>not</u> solving the PDE**, we just want $\boldsymbol{Y}_{\mathrm{t}}$ to be "loosely compliant"
- So, $\mathrm{g}(\boldsymbol{Y}_{\mathrm{t}}) = \frac{\partial \boldsymbol{Y}_{\mathrm{t}}}{\partial t} - \mathcal{N}[\boldsymbol{Y}_{\mathrm{t}}]$ should be small
- Physics-based models:

$$\sum_{t=1}^{T} C\left(\boldsymbol{Z}_{\mathrm{t}}, \widehat{\boldsymbol{Z}}_{\mathrm{t}}(\boldsymbol{\theta}^{\mathrm{process}})\right) + \lambda \mathrm{g}\left(\widehat{\boldsymbol{Y}}_{\mathrm{t}}(\boldsymbol{\theta}^{\mathrm{process}})\right)^2$$

- For some cost function C

The main idea behind PINNs

- We minimize

$$\sum_{t=1}^{T} C\left(\boldsymbol{Z}_{\mathrm{t}}, \widehat{\boldsymbol{Z}}_{\mathrm{t}}(\boldsymbol{\theta}^{\mathrm{process}})\right) + \lambda \mathrm{g}\left(\widehat{\boldsymbol{Y}}_{\mathrm{t}}(\boldsymbol{\theta}^{\mathrm{process}})\right)^{2}$$

- If we have a Gaussian process ("traditional" PINN) $\widehat{\boldsymbol{Z}}_{\mathrm{t}} = \widehat{\boldsymbol{Y}}_{\mathrm{t}}$ and we have

$$\sum_{t=1}^{T} \left(\boldsymbol{Z}_{\mathrm{t}} - \widehat{\boldsymbol{Z}}_{\mathrm{t}}(\boldsymbol{\theta}^{\mathrm{process}})\right)^{2} + \lambda \mathrm{g}\left(\widehat{\boldsymbol{Z}}_{\mathrm{t}}(\boldsymbol{\theta}^{\mathrm{process}})\right)^{2}$$

- We penalize values that departs from the PDE
- Good computational news: **this is not an inverse problem, it's a forward problem**
- Bad computational news: **no closed form, requires gradient descent, backprop, etc.**

PINNs and penalized functional regression

- Recall **PINN**

$$\text{argmin}_{\boldsymbol{\theta}} \sum_{t=1}^{T} \left(\boldsymbol{Z}_{\mathrm{t}} - \widehat{\boldsymbol{Z}}_{\mathrm{t}}(\boldsymbol{\theta}^{\text{process}}) \right)^2 + \lambda \mathrm{g} \left(\widehat{\boldsymbol{Z}}_{\mathrm{t}}(\boldsymbol{\theta}^{\text{process}}) \right)^2$$

- Even this penalized approach is actually not new in statistics
- Sangalli and collaborators at Politecnico di Milano cast this as a (spatial) functional problem

$$\text{argmin}_{\widehat{\boldsymbol{Y}}} \sum_{i=1}^{n} \left(\boldsymbol{Z}(s_i) - \widehat{\boldsymbol{Z}}(s_i) \right)^2 + \lambda \int \mathrm{g} \left(\widehat{\boldsymbol{Z}}(s) \right)^2 \mathrm{d}s$$

- There is no NN, but the idea is very similar and in a continuous setting

Work 1

- We model the process

$$Z_t(\boldsymbol{s}) = \mathcal{H}\left(Y_t(\boldsymbol{s}), \boldsymbol{\theta}^{\text{data}}, \varepsilon_t(\boldsymbol{s})\right)$$

$$Y_t(\boldsymbol{s}) = \mathcal{M}(\{Y_{t-1}(\boldsymbol{s}), \dots, Y_{t-k}(\boldsymbol{s})\}, \boldsymbol{\theta}^{\text{process}}, \eta_t(\boldsymbol{s}))$$

- We model $\mathcal{M}$ in two different ways
- 1) PS: $\mathcal{M}$ is derived from the PDE
- 2) $\mathcal{M}$ is some very large NN and is “nudged” towards the PDE solution

Contribution 1 (Bonas et al., JASA): develop a flexible temporal model NN for $\mathcal{M}$ (Gaussian $\mathcal{H}$) and inform it with a PDE in an experimental fluid dynamic problem (Navier-Stokes)

Echo State Networks

- Temporal model, call $\mathbf{Z}_\mathrm{t} = \{Z_t(\boldsymbol{s})\}$

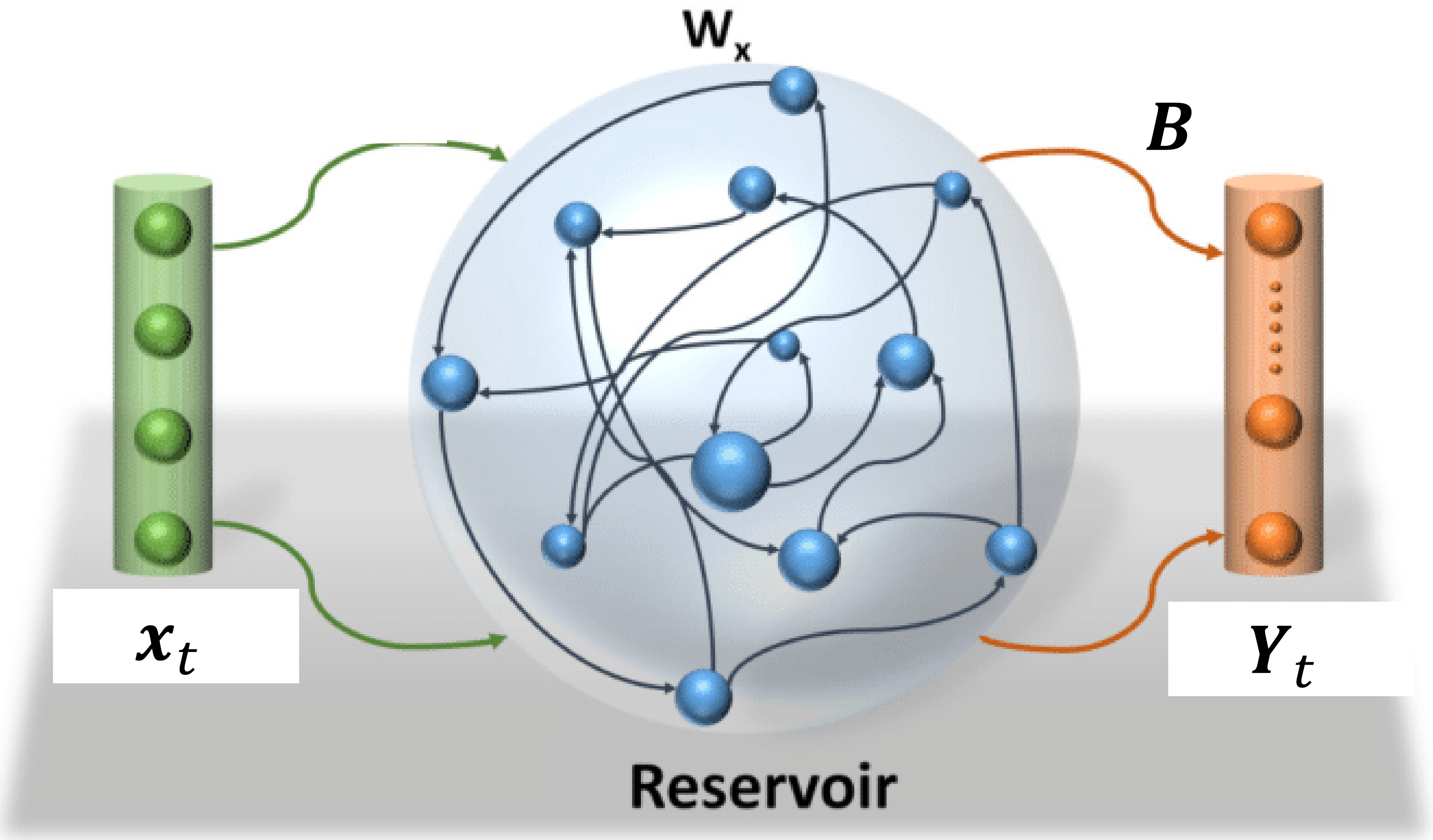
$$\mathbf{Z}_\mathrm{t} = \boldsymbol{B}\boldsymbol{Y}_t + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathrm{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$
$$\boldsymbol{Y}_t = (1-\alpha)\boldsymbol{Y}_{t-1} + \alpha\boldsymbol{\omega}_t$$
$$\boldsymbol{\omega}_t = \mathrm{g}(\boldsymbol{W}\boldsymbol{Y}_{t-1} + \boldsymbol{W}^x\boldsymbol{x}_t)$$

- **Sparse random matrices with spike and slab prior (reservoir)**

$$W_{ij} = p_{ij}^W f(\eta_W) + \left(1 - p_{ij}^W\right)\delta_0, \qquad p_{ij}^W \sim Be(\pi_W)$$
$$W_{ij}^x = p_{ij}^{W^x} f(\eta_{W^x}) + \left(1 - p_{ij}^{W^x}\right)\delta_0, \qquad p_{ij}^{W^x} \sim Be(\pi_{W^x})$$

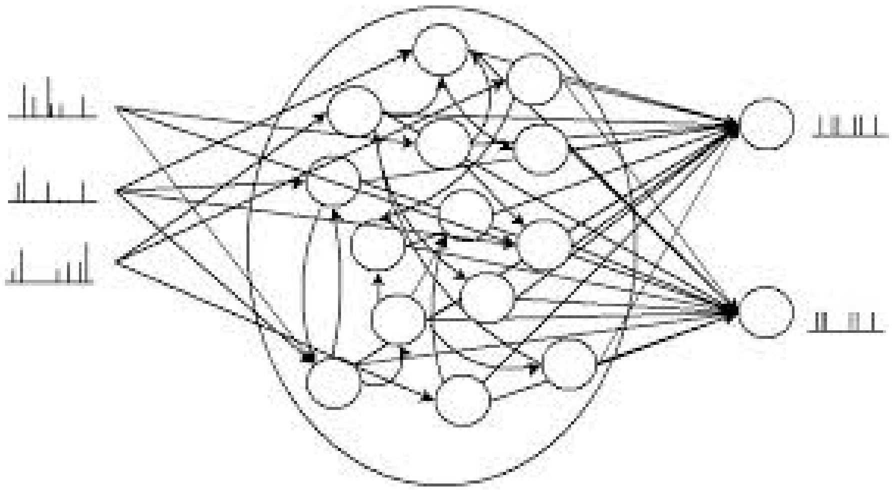
- This is called an **Echo State Network (ESN)**
- This is the shallow version, but **I will use a deep network (DESN)**

ESN (simplified) in a picture



Liquid State Machines

- We also use Deep Liquid State Machines (DLSM) for time series
- This NN **mimics more closely how the brain processes information**
- The input (past data) are transformed in a **subprocess with a firing rate proportional to the data** (*spike train*)
- **Spike trains is used as input** and multiplied by synapse weights to increase (decrease) the stored energy of a neuron



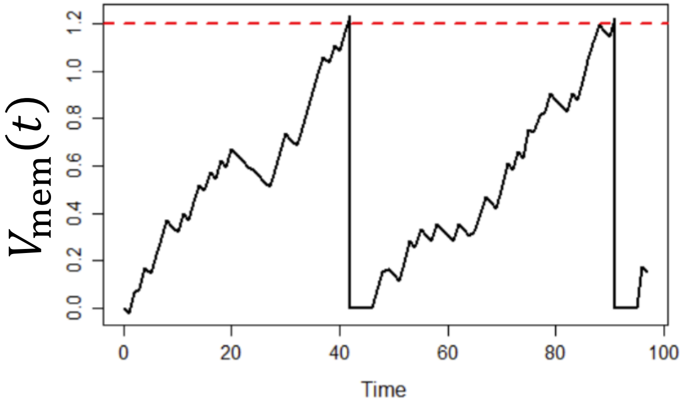
Liquid State Machines

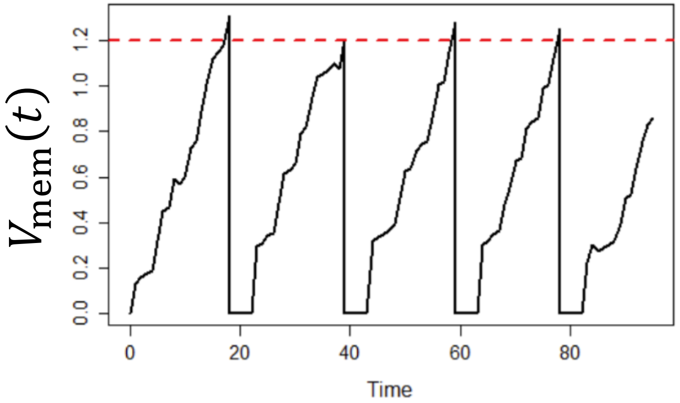
- Neurons 'spike' once their energy crosses a threshold
- The signal is then sent deeper into the network (like a brain synapse)
- Evolution of each neuron's i potential energy at time t

$$V_{\text{mem}}(t) = V_{\text{mem}}(t-1) + \sum_{j \in \text{synapse}} w_{j;i} I_t^{(j)} - V_{\text{leak}}$$

$$V_{\text{mem}}(t) = V_{\text{res}} \ \text{ if } V_{\text{mem}}(t) \geq V_{\text{thr}}$$

$$s_t = \begin{cases} 1, & \text{if} \quad V_{\text{mem}}(t) \geq V_{\text{thr}} \\ 0, & \text{if} \quad V_{\text{mem}}(t) < V_{\text{thr}} \end{cases}$$





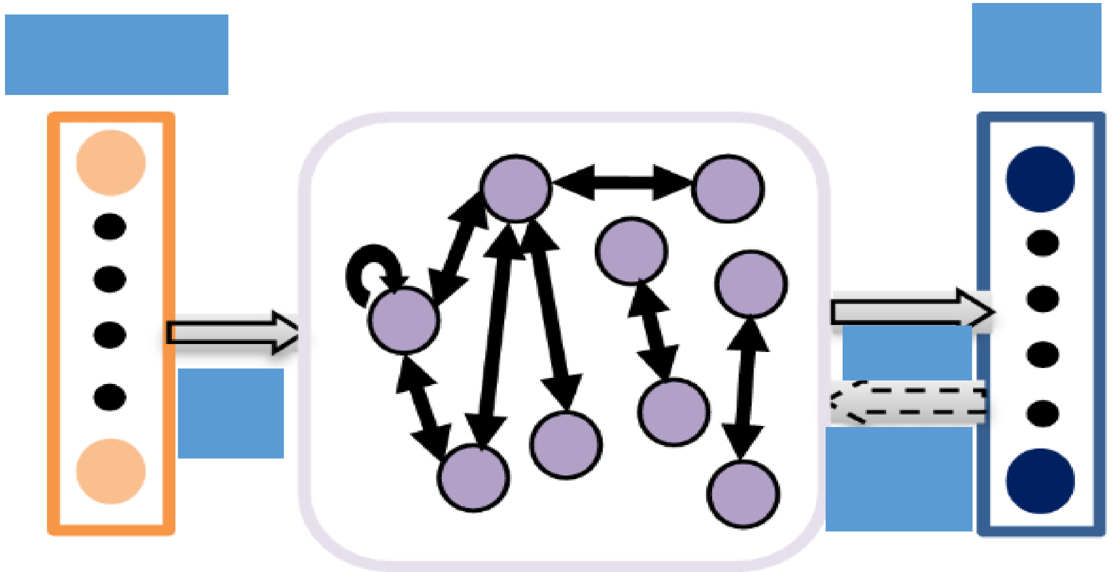
Liquid State Machines

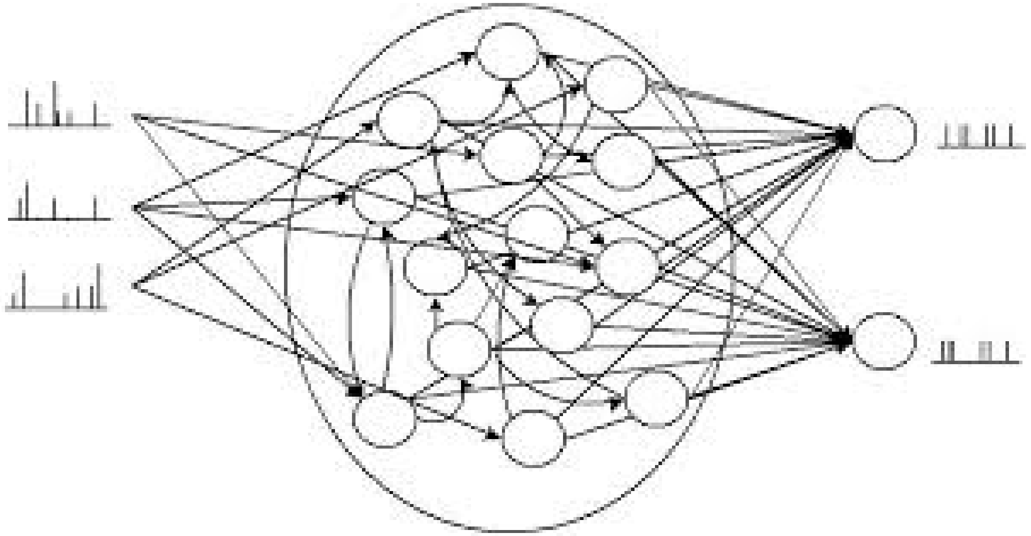
- We finally count how many spikes

$$\omega_t = \sum_{t^*=1}^{T^*} s_{t^*}$$

- And feed this into a reservoir
- It's actually more complicated than this
- There are two more processes to mimic synapses
- 1) **latent period** after firing
- 2) **lateral inhibition** (once a synapse fires, nearby synapses can't)

DESN

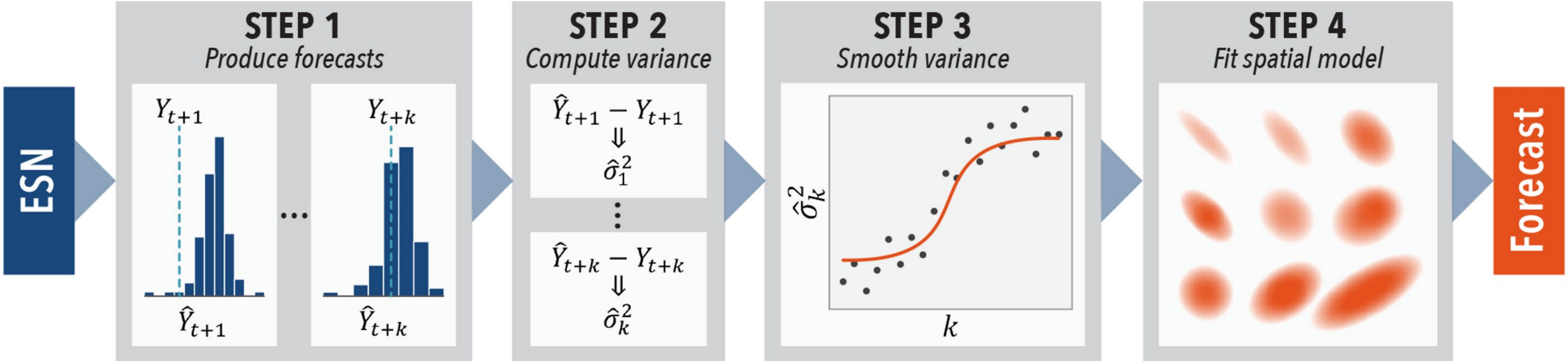




DLSM

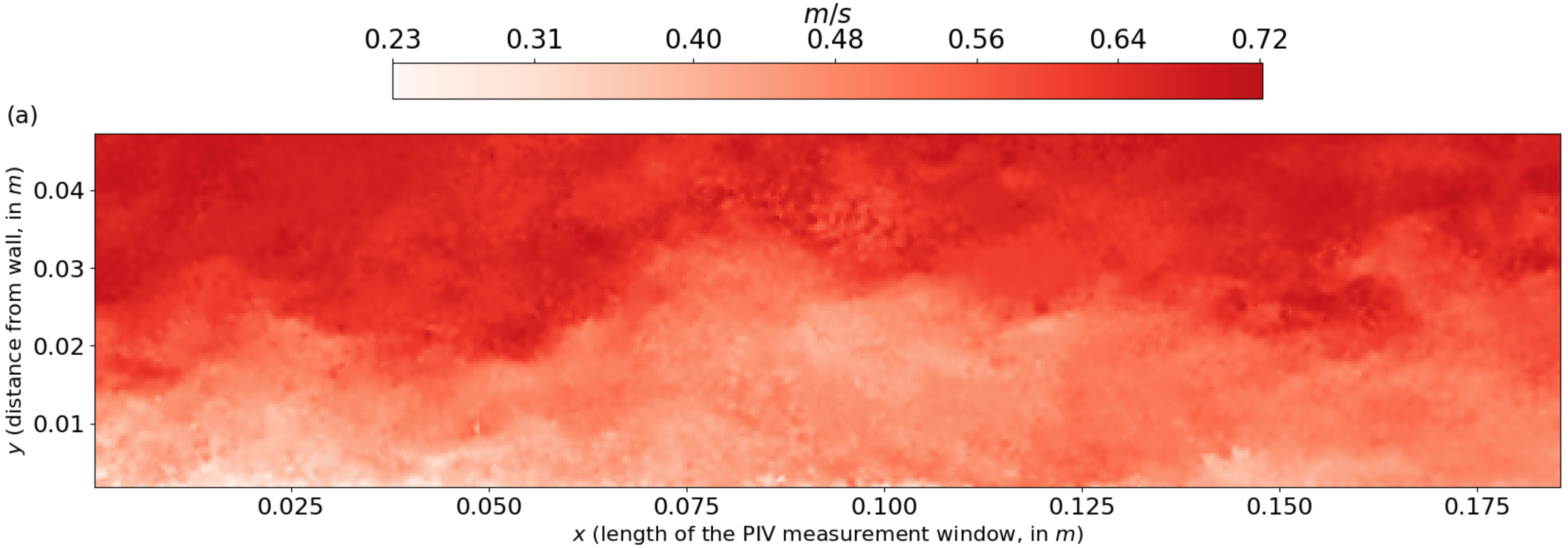
ENSEMBLE
RESERVOIR

What about uncertainty?

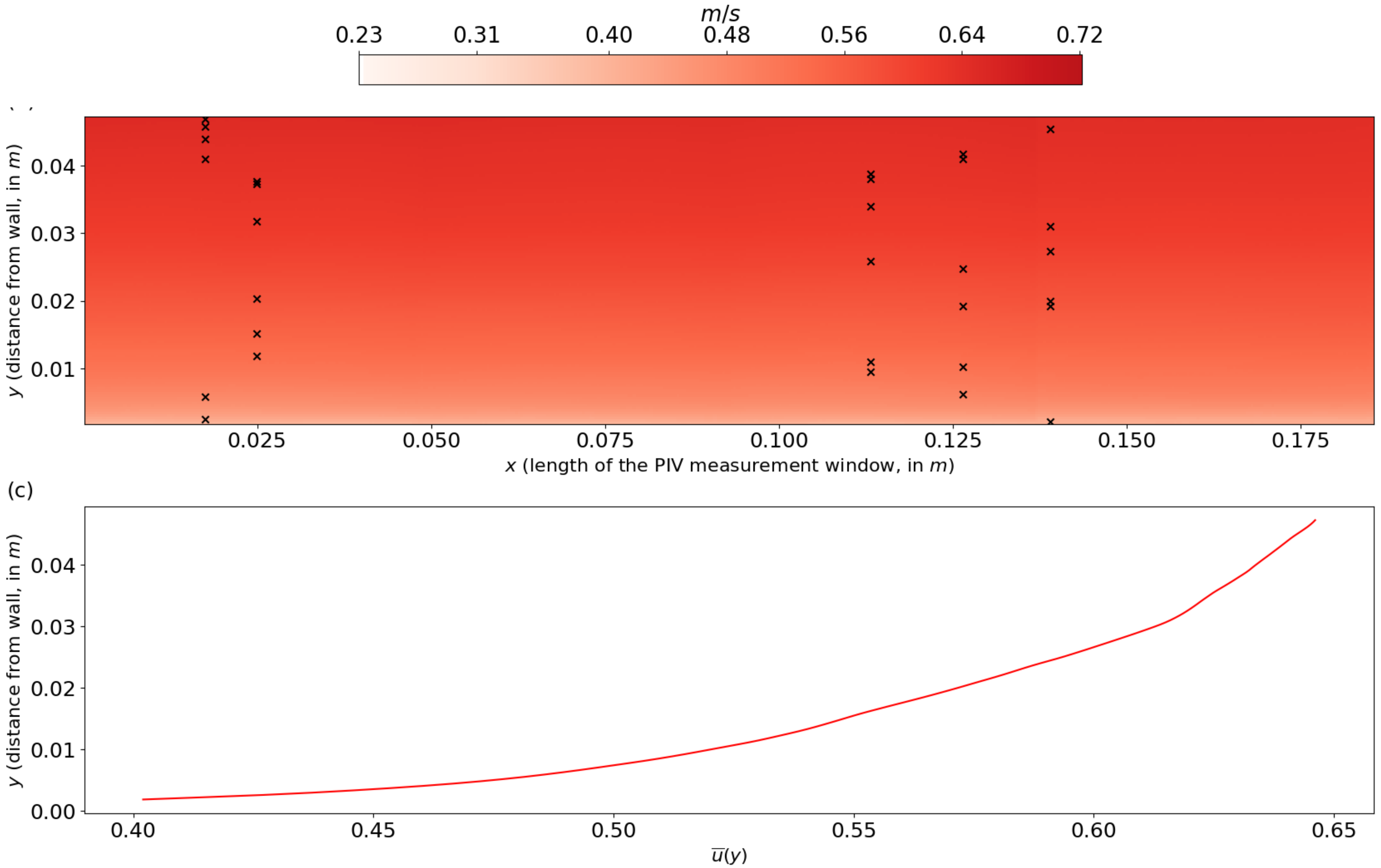


Data

- **Velocity data of water flow in a tunnel**
- Bottom: wall, top: free flow
- 2,500 time steps at frequency of 1,000 Hz: total time 2.5s
- **Goal: provide physically consistent forecasting**



Average velocity profile



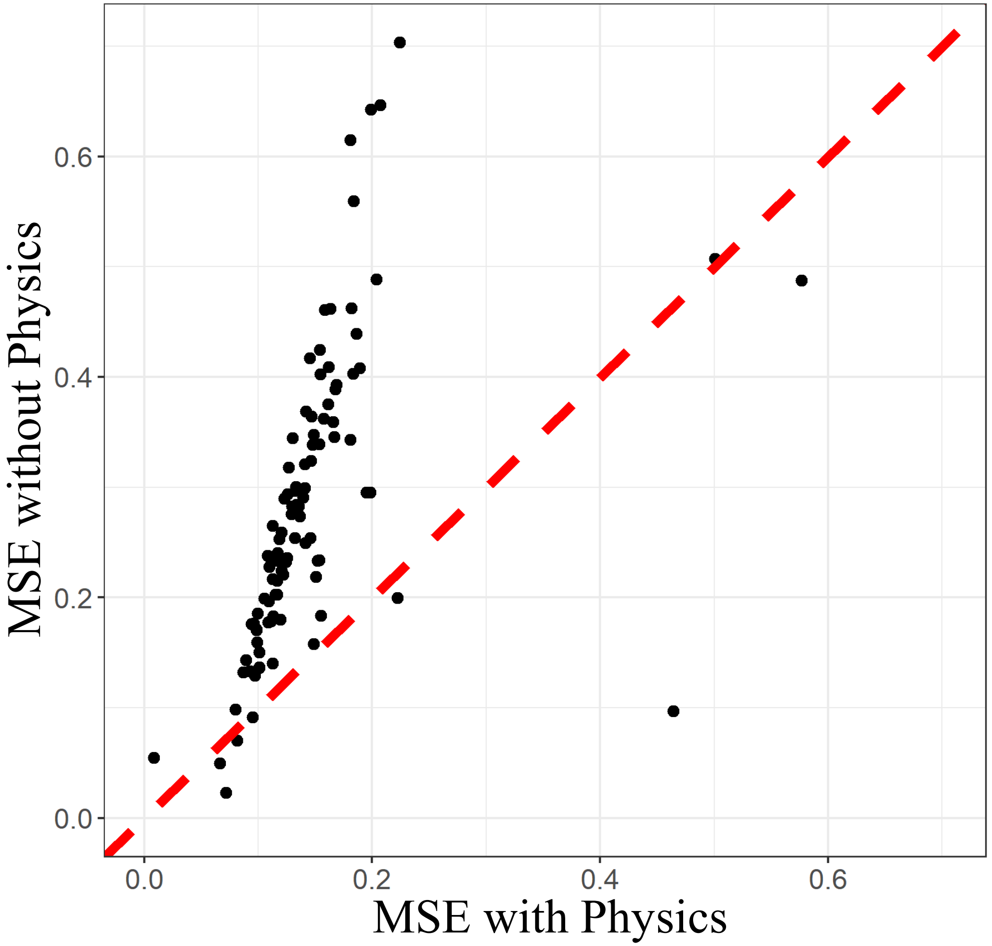
PDE derivation

- Controlled experiment: 2D fluid at equilibrium
- We use time averaged (RANS) solution to Navier Stokes, which reduces to

$$\frac{u_\tau}{\delta} + \frac{u_\tau \delta}{\mathrm{RE}_\tau} \frac{\partial^2 \bar{Y}}{\partial y^2} + \left\{ (\kappa y)^2 \left| \frac{\partial \bar{Y}}{\partial y} \right| \right\} \frac{\partial \bar{Y}}{\partial y} = 0$$

- $\bar{Y}(x, y)$: average water velocity
- δ: boundary layer thickness (known)
- RE_τ: Reynolds number (known)
- u_τ: friction velocity (known)

Method	MSE
With Physics	0.13 (0.05)
No Physics	0.26 (0.17)



Uncertainty quantification

Prediction Interval	Uncalibrated	Calibrated
95%	100.0 (0.0)	**95.0 (1.0)**
90%	100.0 (0.0)	**90.0 (2.0)**
80%	90.0 (3.0)	**80.0 (2.0)**

Wrap up work 1

- **A PINN is just a penalized statistical model**
- It works really well in cases where the physics is well known
- No free lunch: you need the right approximation of the Navier Stokes equation
- We also performed uncertainty quantification and calibrated the forecast
- If the PDE penalty has some unknown physical parameters (e.g., **Reynolds number and viscosity**), they can be estimated as well!

Work 2

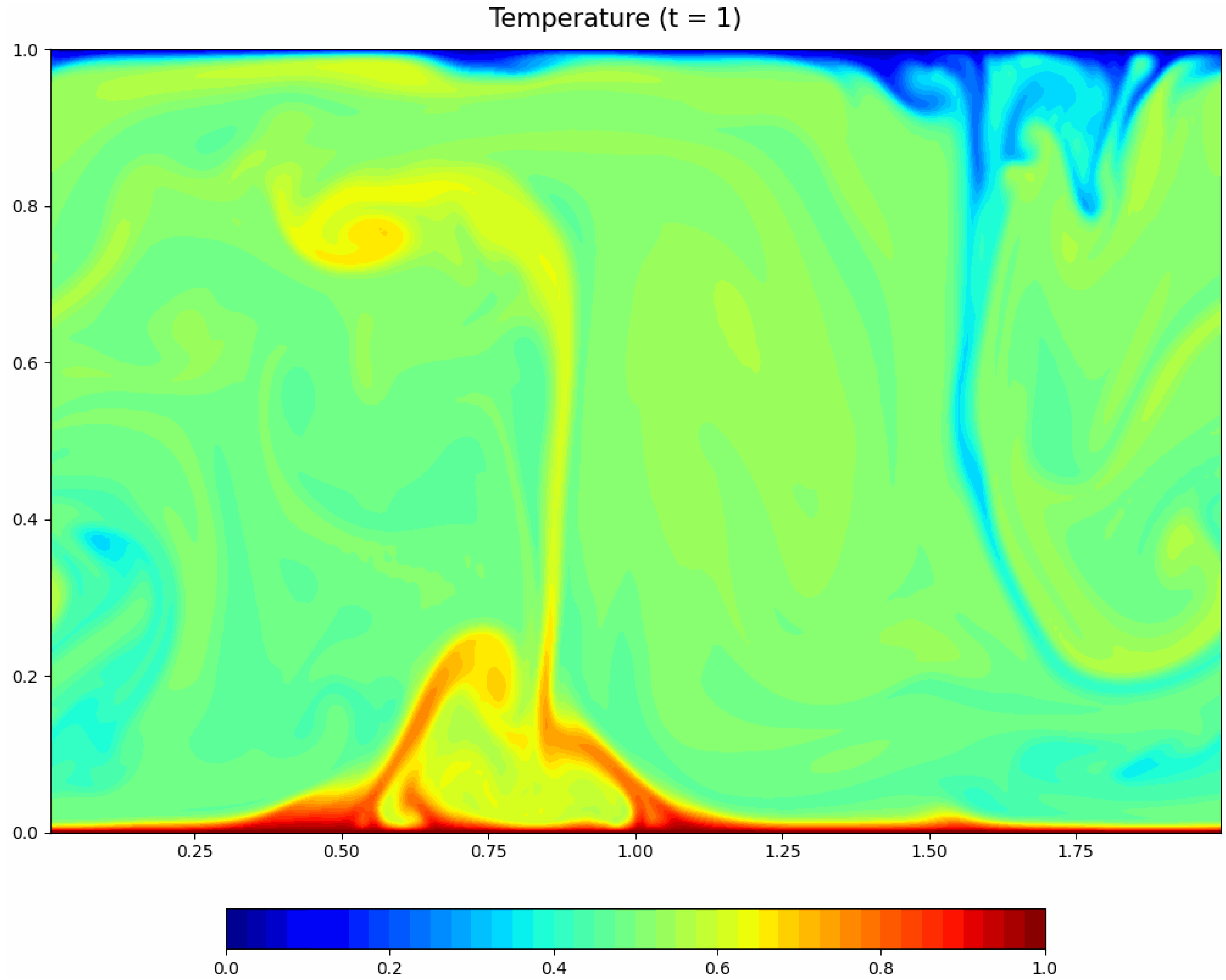
- We model the process

$$Z_t(\boldsymbol{s}) = \mathcal{H}\left(Y_t(\boldsymbol{s}), \boldsymbol{\theta}^{\text{data}}, \varepsilon_t(\boldsymbol{s})\right)$$
$$Y_t(\boldsymbol{s}) = \mathcal{M}(\{Y_{t-1}(\boldsymbol{s}), \dots, Y_{t-k}(\boldsymbol{s})\}, \boldsymbol{\theta}^{\text{process}}, \eta_t(\boldsymbol{s}))$$

- $\mathcal{M}$ is some very large NN and is "nudged" towards the PDE solution

Contribution 2 (Menicali et al., under review): A PINN to generate more realization from a numerical model (digital twin). Model: autoencoder for $\mathcal{H}$, an a recurrent NN for $\mathcal{M}$, informed with Navier-Stokes

The simulation: Rayleigh–Bénard convection (RCB)



Data and problem

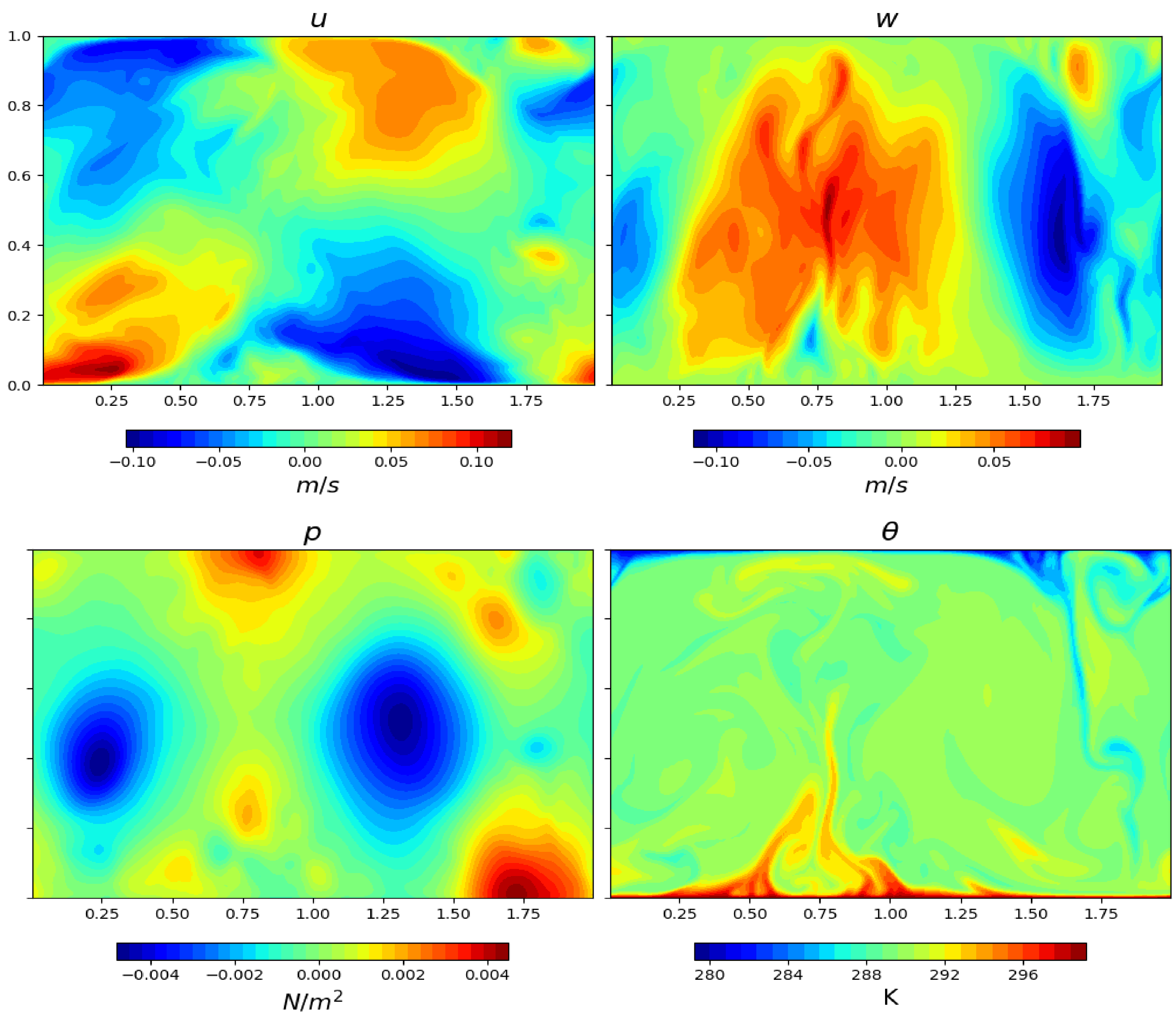
- Two dimensional fluid
- Variables: velocity $\boldsymbol{u}_t = (u_t, w_t)$, temperature θ_t and pressure p_t
- Navier Stokes becomes:

$$
\nabla \cdot \boldsymbol{u}_t = 0,
$$

$$
\frac{\partial \boldsymbol{u}_t}{\partial t} + (\boldsymbol{u}_t \cdot \nabla)\boldsymbol{u}_t = -\boldsymbol{\nabla} p_t + \sqrt{\frac{\mathrm{Pr}}{Ra}}\, \nabla^2 \boldsymbol{u}_t + \theta_t \boldsymbol{e}_z,
$$

$$
\frac{\partial \theta_t}{\partial t} + (\boldsymbol{u}_t \cdot \nabla)\theta_t = \sqrt{\frac{1}{\mathrm{PrRa}}}\, \nabla^2 \theta_t,
$$

- $\boldsymbol{e}_z$: (0,1) vector
- Pr and Ra: Prandtl and Rayleigh constants
- The data are simulated on a 256×256 grid and $1{,}000$ time points



u
w
p
θ
1.0
0.8
0.6
0.4
0.2
0.0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
−0.10
−0.05
0.00
0.05
0.10
m/s
−0.004
−0.002
0.000
0.002
0.004
N/m2
280
284
288
292
296
K

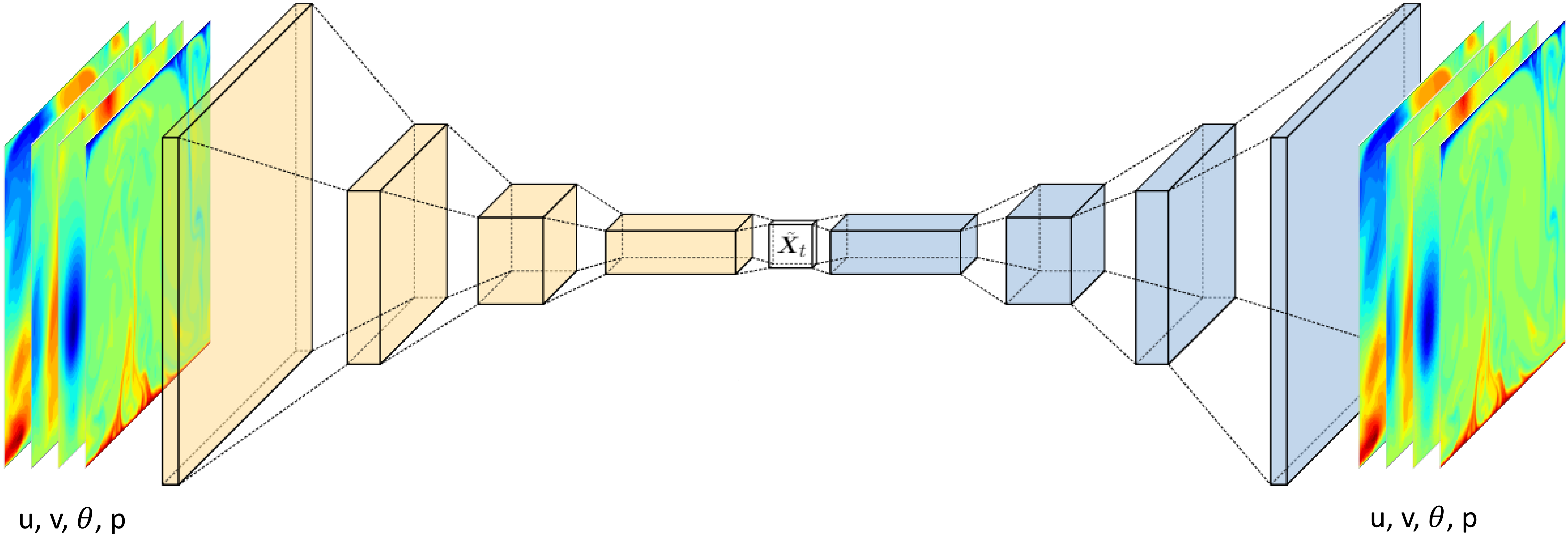
Convolutional Autoencoder

- We model the spatio-temporal process

$$\boldsymbol{Y}_t(\boldsymbol{s}) = \big(u_t(\boldsymbol{s}), w_t(\boldsymbol{s}), p_t(\boldsymbol{s}), \theta_t(\boldsymbol{s})\big),$$
$$\boldsymbol{Y}_t = \{\boldsymbol{Y}_t(\boldsymbol{s})\}$$

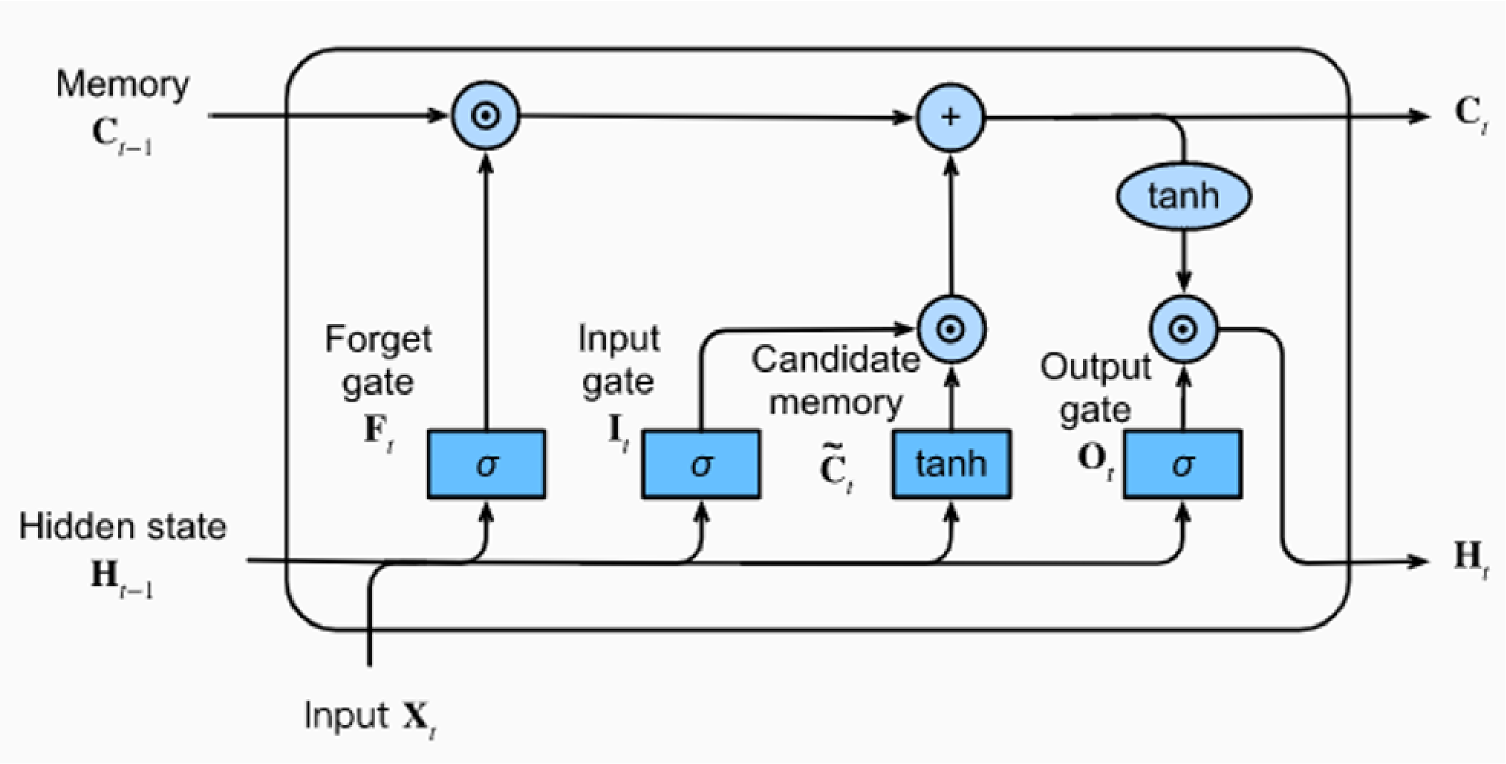
- We model the spatial structure of $\boldsymbol{Y}_t$ with a Convolutional Autoencoder (CAE)
- The input is projected onto a latent space: the **spatial encoder**
- Then projected back onto the original space: **spatial decoder**
- It’s a **dimension reduction technique** in space, like fixed rank Kriging, predictive processes, etc.

A CAE in a picture

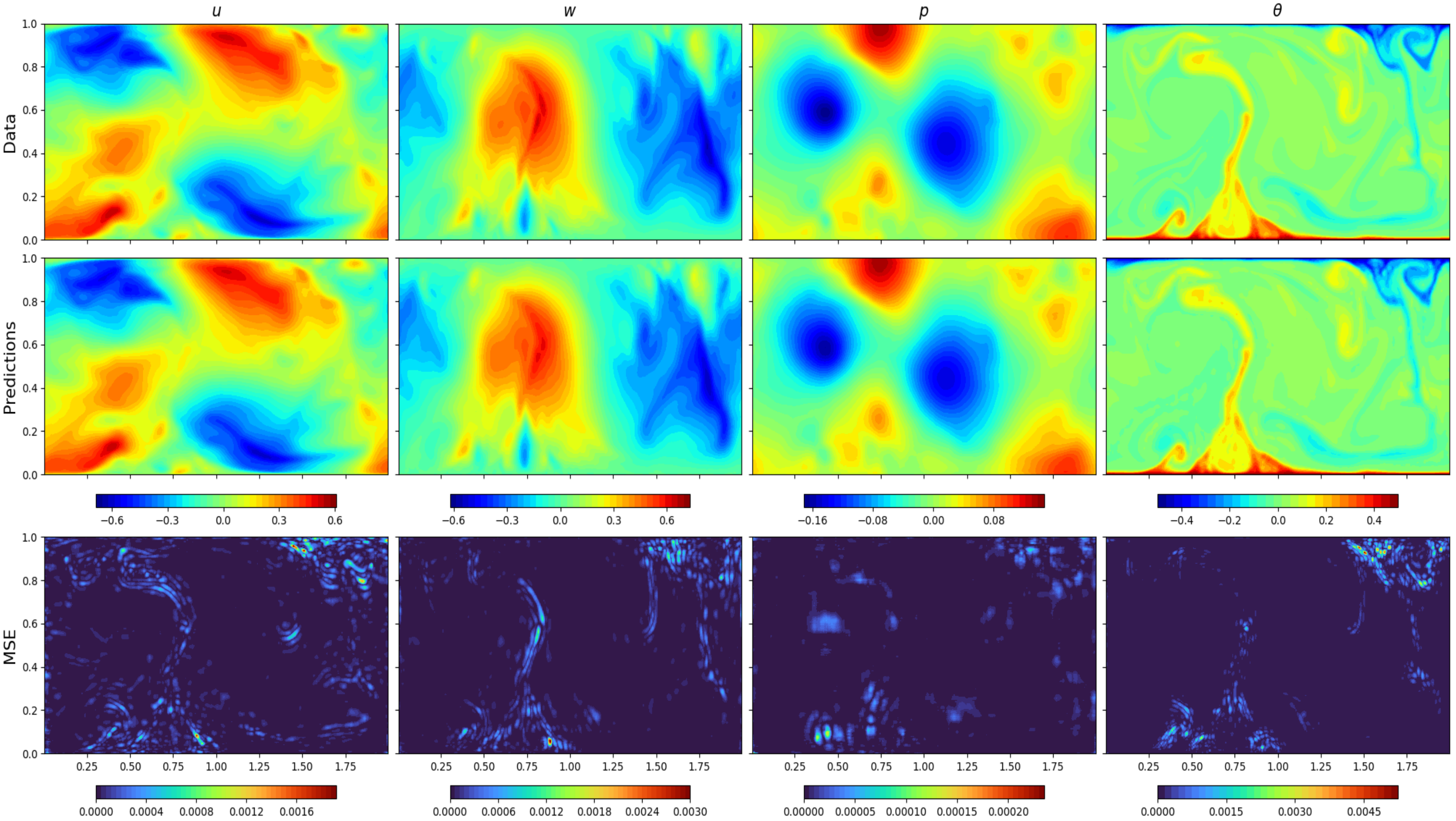


ConvLSTM

- We assume an LSTM model for the temporal structure
- This is called a Convolutional Long-Short Term Memory (ConvLSTM) model



Results: Spatial part



MSE

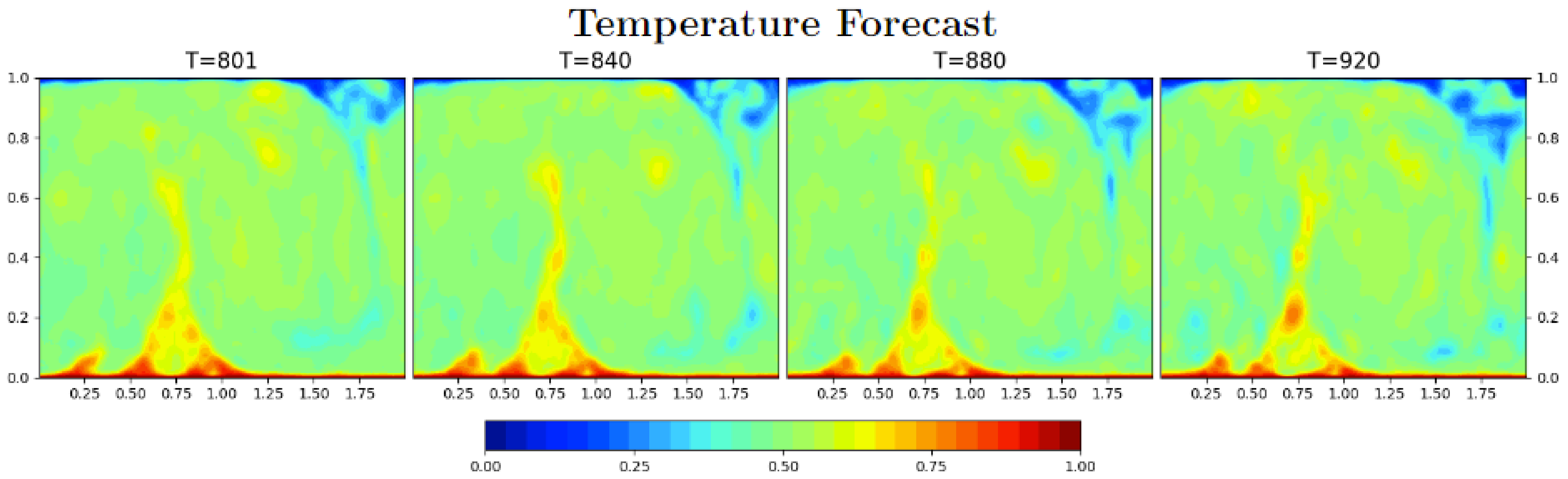
Fixed Kriging: 5.0×10^{5}

PCA: 1.5×10^{-3}

ICA: 3.6×10^{-3}

CAE: $\mathbf{3.1 \times 10^{-5}}$

Results: Forecast



MSE

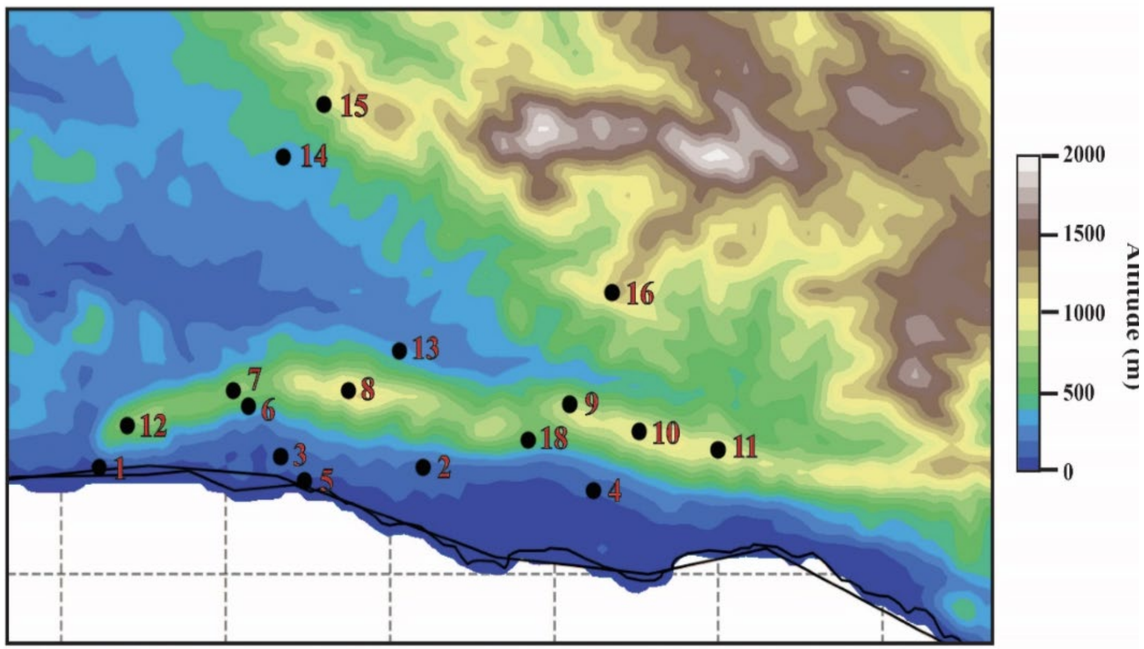
ARMA: 163.25

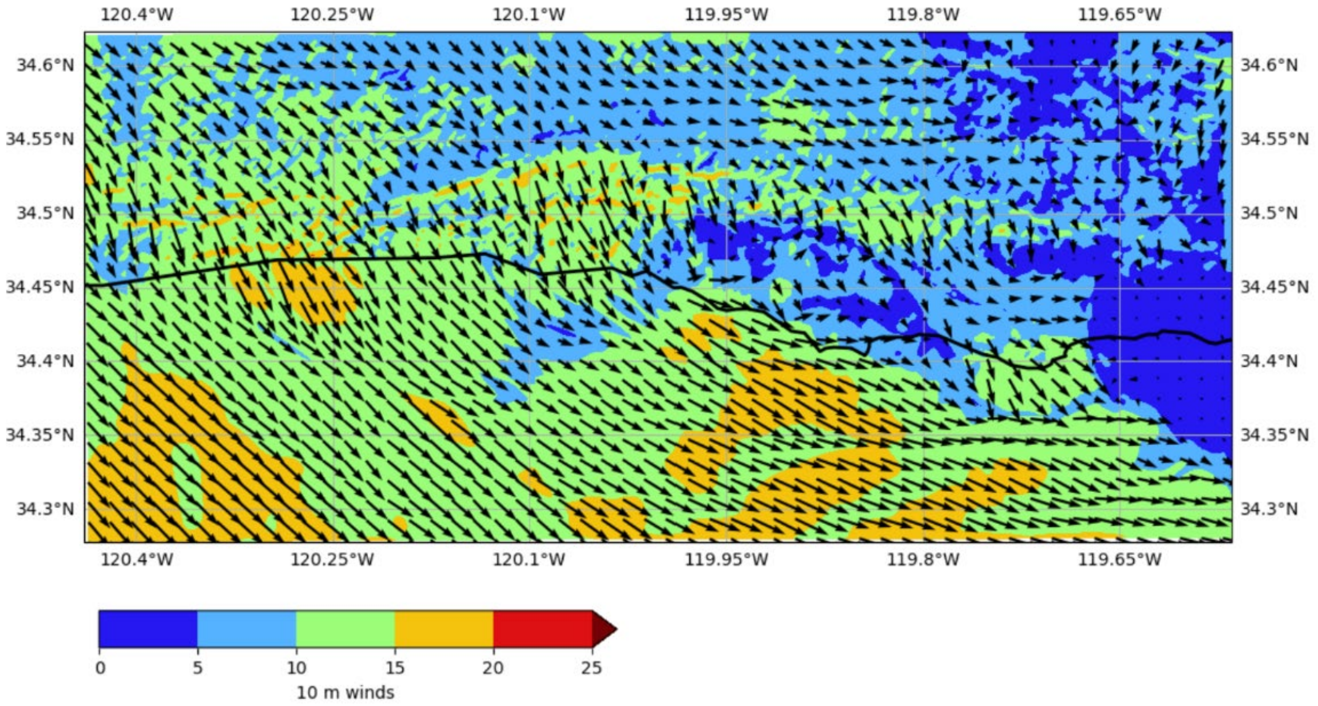
PI-ESN: 21.48

CRNN: 0.36

Our Model: 0.22

PINNs and WRF-LES: the SWEX experiment





- A PINN for WRF-LES (mesoscale-microscale): no more simplifying Navier Stokes
- Tasks: integrate a statistical model with the WRF Fortran (!) and build a PINN

Other interesting projects

- **PINNs and physics-informed emulators** for satellite retrievals
- **Physics-informed priors**
- **Land-vegetation dynamics** in the Midwest Holocene
- **Pushing emulations to exascale** for Petabytes of data (2024 Gordon Bell!)
- Statistical models for climate output **compression**
- Visualization in VR and other 3D environments
- High energy particle physics, optics, etc.

Thanks very much for your attention!