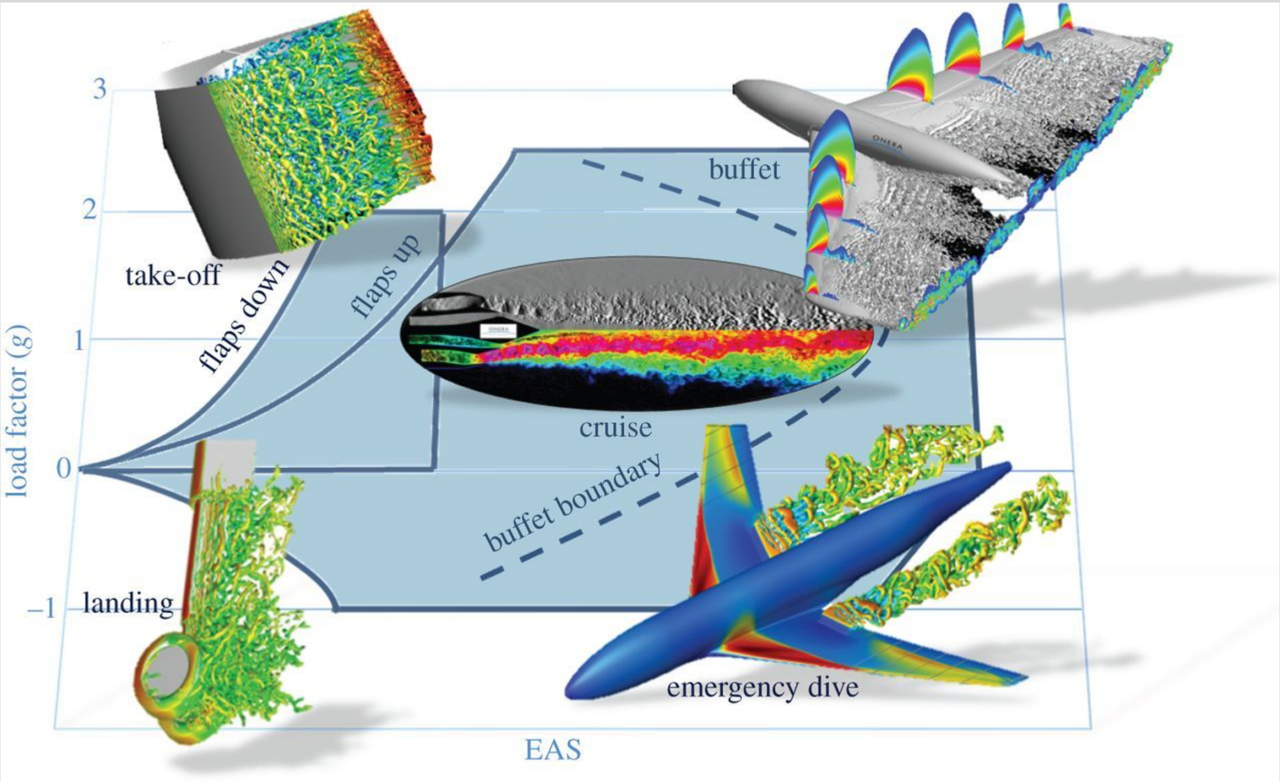
Deep Learning for Surrogate Models of Turbulence

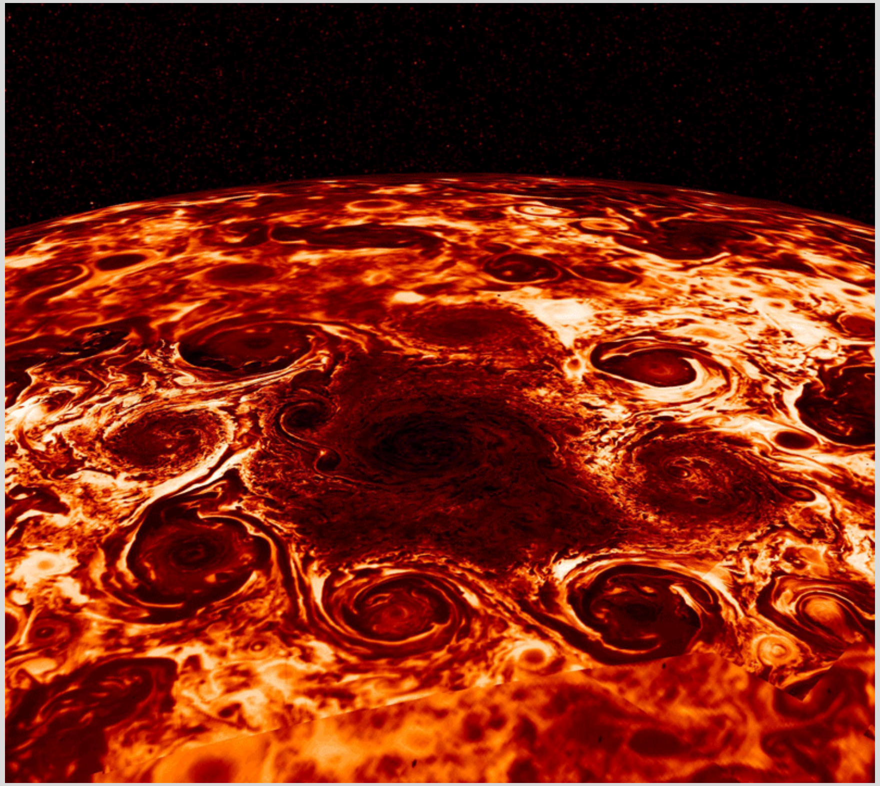
Fernando Gonzalez

fernando.gonzalez@coria.fr

Why the interest in predicting turbulence?

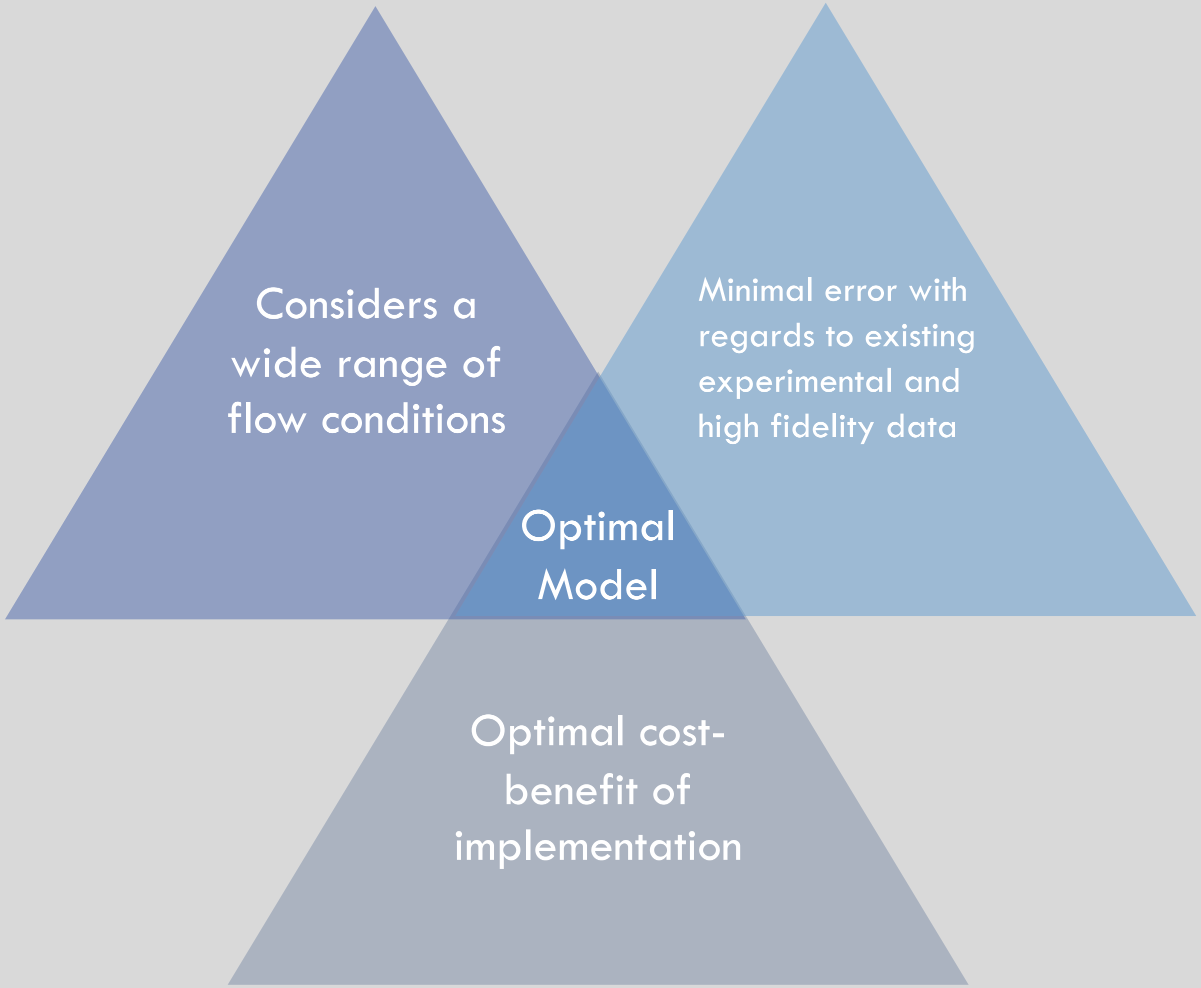


Deck, S., Gand, F., Brunet, V., & Ben Khelil, S. (2014). High-fidelity simulations of unsteady civil aircraft aerodynamics: stakes and perspectives. Application of zonal detached eddy simulation. *Philosophical Transactions Of The Royal Society A: Mathematical, Physical And Engineering Sciences*, *372*(2022), 20130325. doi: 10.1098/rsta.2013.0325



Cyclones at Jupiter's north pole.
NASA, JPL-Caltech, SwRI, ASI, INAF, JIRAM

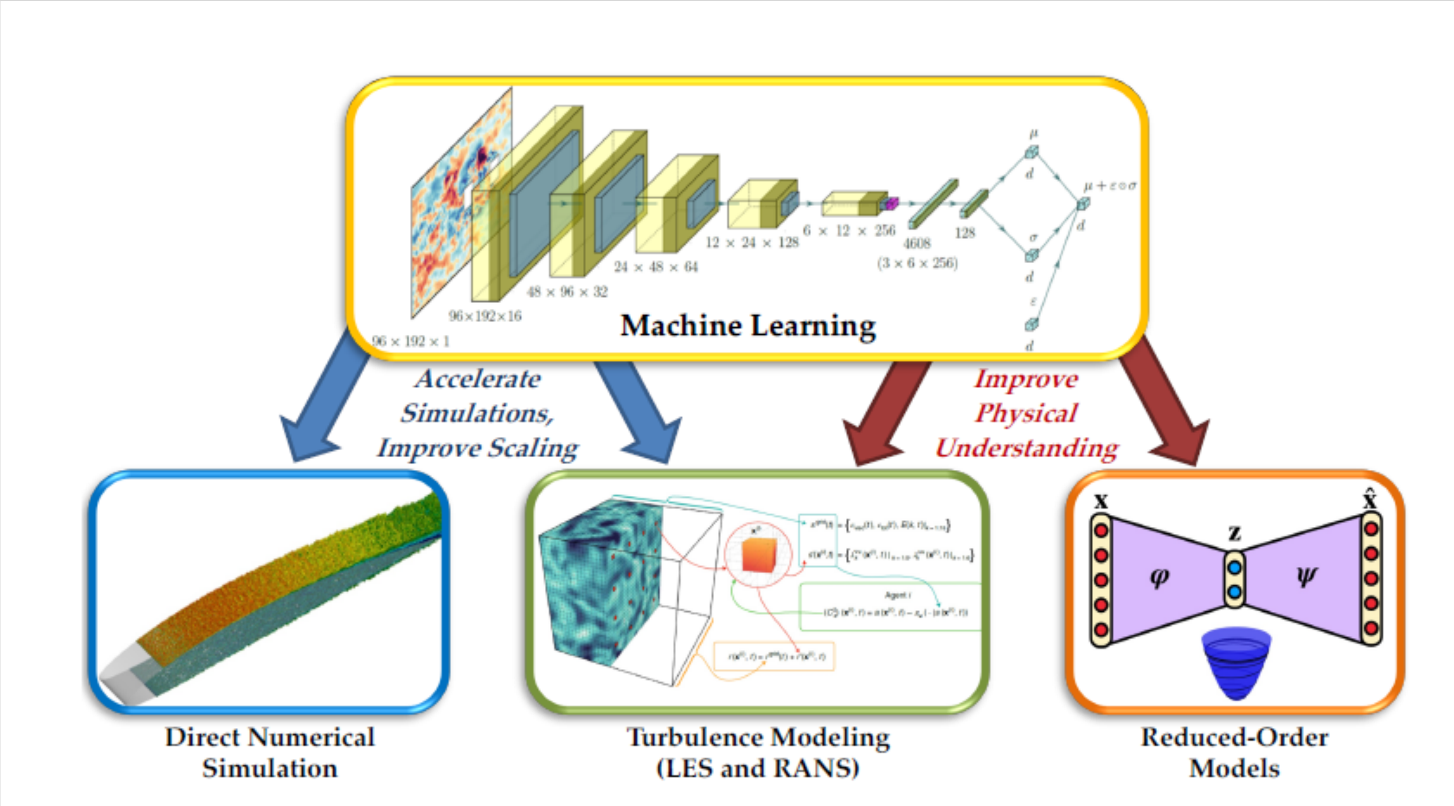
The opportunity for Machine Learning in Turbulence Modelling



"It would be extremely valuable to develop a general methodology to determine optimal models" –

Pope, Stephen (1999) . A Perspective On Turbulence Modelling

Uses of ML in Computational Fluid Dynamics



Vinuesa, R., & Brunton, S. L. (2021). The Potential of Machine Learning to Enhance Computational Fluid Dynamics. ArXiv:2110.02085 [Physics]. http://arxiv.org/abs/2110.02085

Reduced-order Modelling of Turbulent flows

Control Applications	Design Optimization
Accelerating Simulations	Risk Analysis

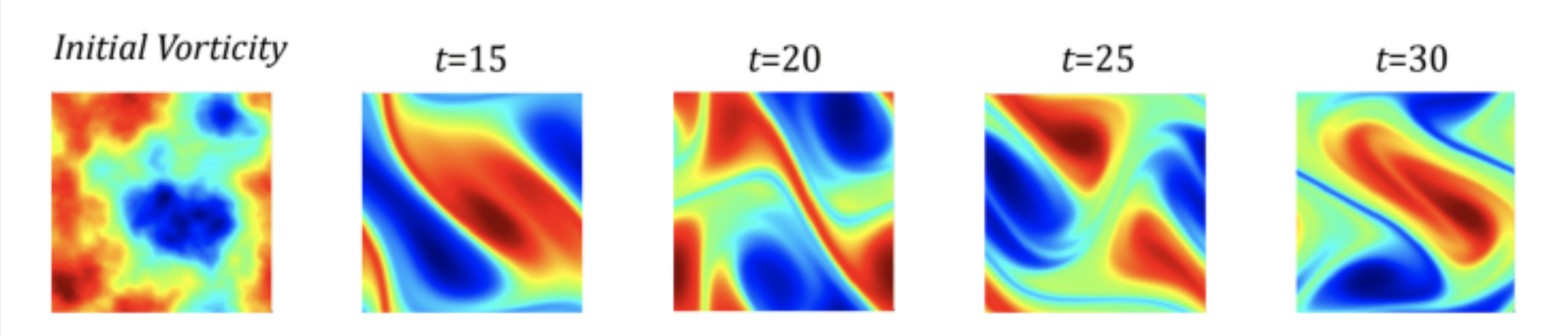
Methods

1. Fully Data-Driven approach: Convolutional Autoencoder + LSTM

2. Physics-Informed Machine Learning: DeepONet

Data: 2D Turbulence

$$
\begin{aligned}
\partial_t w(x,t) + u(x,t)\cdot\nabla w(x,t) &= \nu\Delta w(x,t) + f(x), && x\in(0,1)^2, t\in(0,T] \\
\nabla\cdot u(x,t) &= 0, && x\in(0,1)^2, t\in[0,T] \\
w(x,0) &= w_0(x), && x\in(0,1)^2
\end{aligned}
$$



Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier Neural Operator for Parametric Partial Differential Equations. *ArXiv:2010.08895 [Cs, Math]*. http://arxiv.org/abs/2010.08895

https://github.com/zongyi-li/fourier_neural_operator

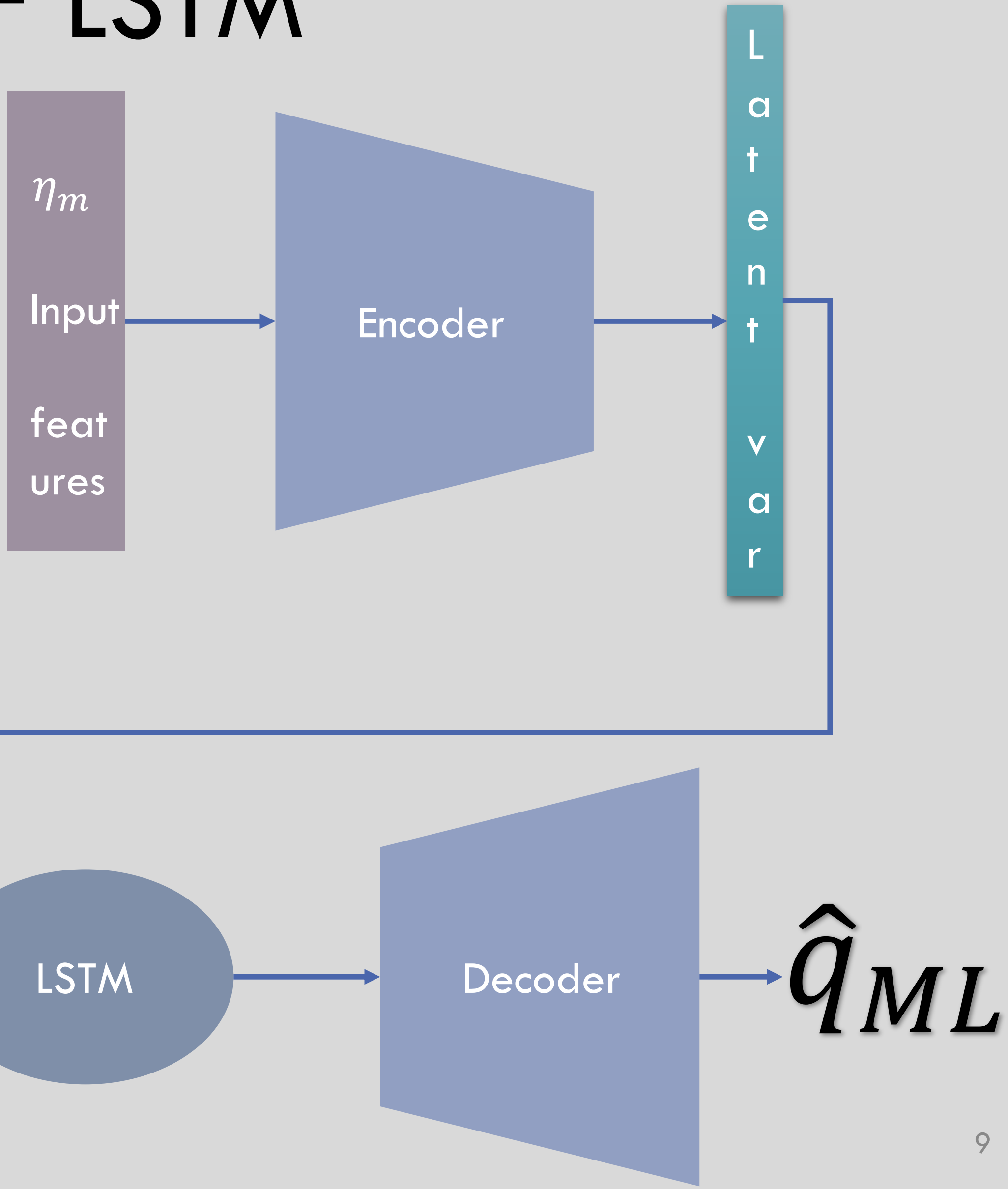
Convolutional Autoencoder + LSTM

Data-driven ROM: CAE + LSTM

A convolutional Autoencoder lowers the dimensionality of the problem.

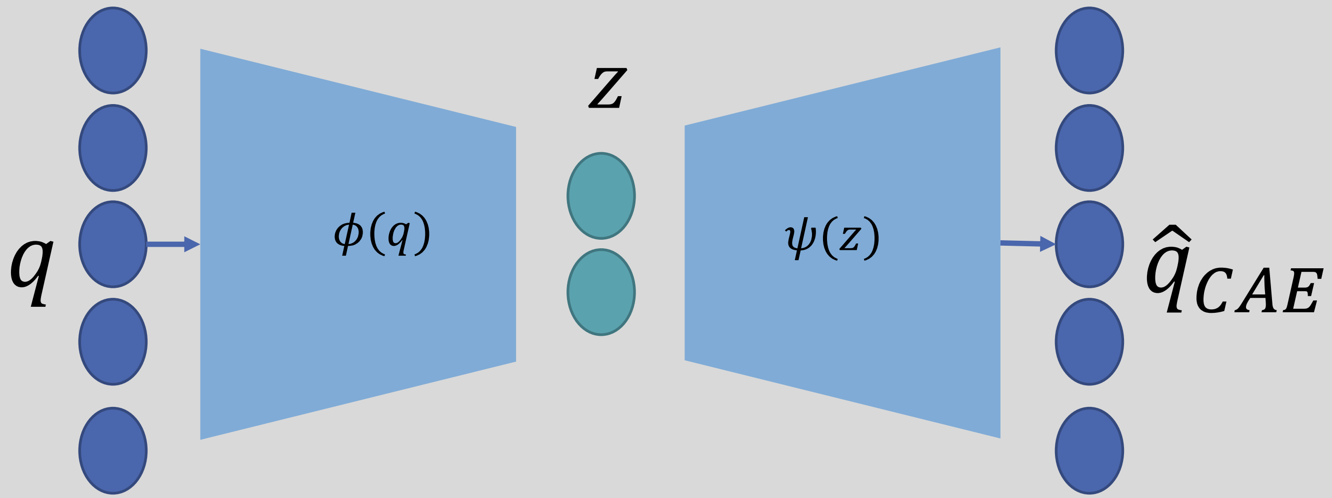
Learns unsteady behavior

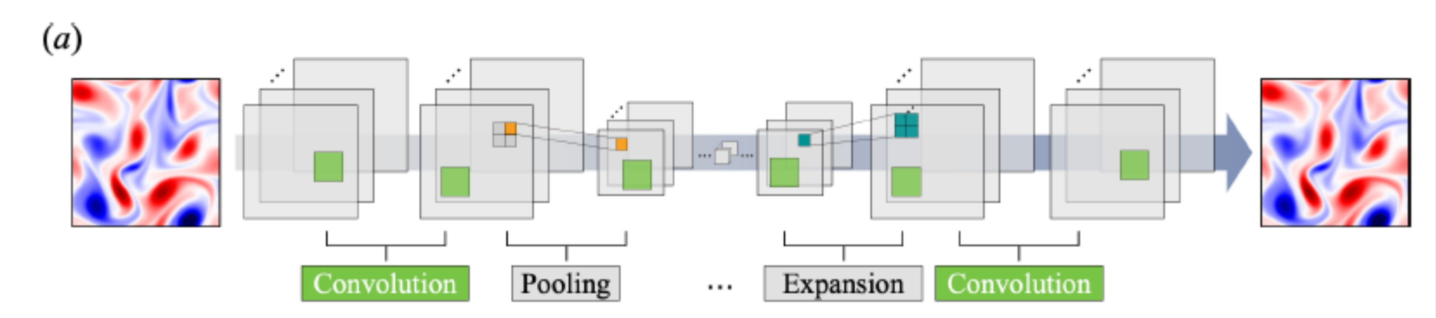
Tensor-Train decomposition for better long range predictions



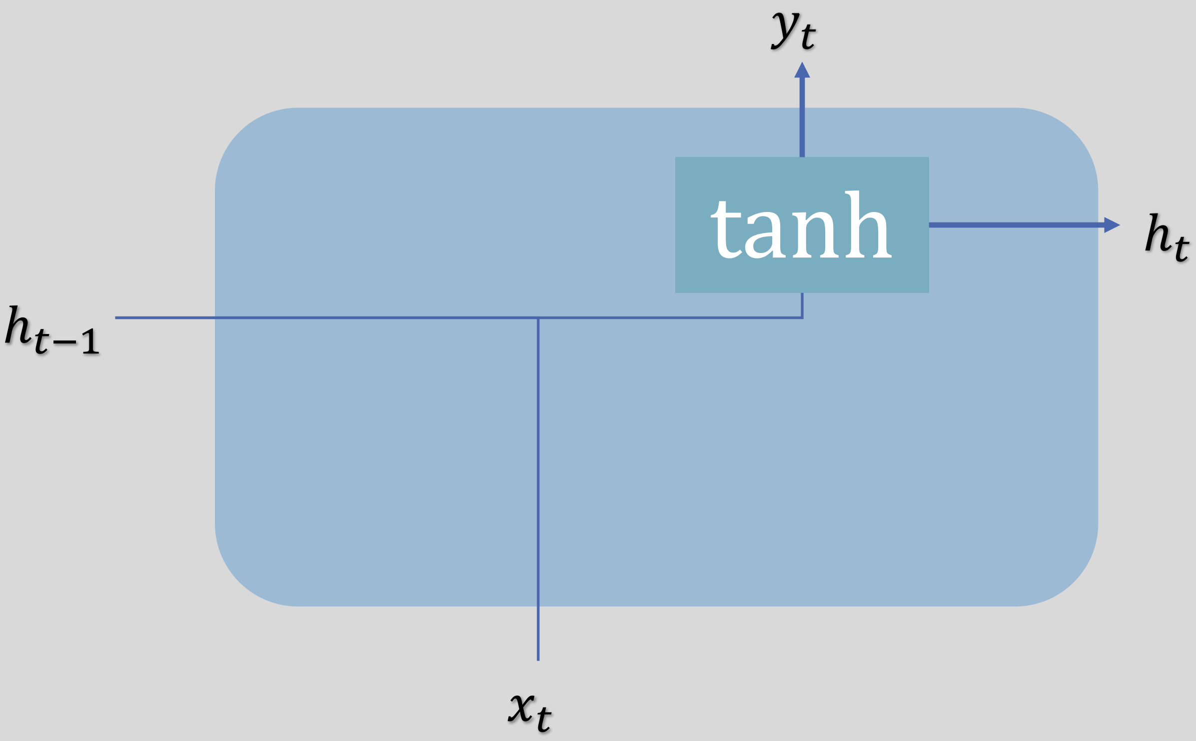
Mohan, A., Daniel, D., Chertkov, M., & Livescu, D. (2019). Compressed Convolutional LSTM: An Efficient Deep Learning framework to Model High Fidelity 3D Turbulence. *ArXiv:1903.00033 [Nlin, Physics:Physics].* http://arxiv.org/abs/1903.00033

Convolutional Autoencoder for dimensionality reduction



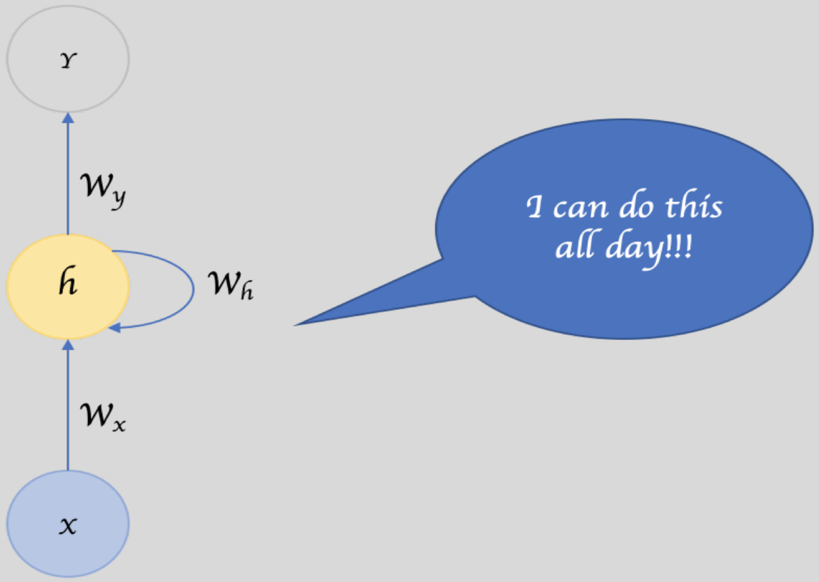


Recurrent Neural Network

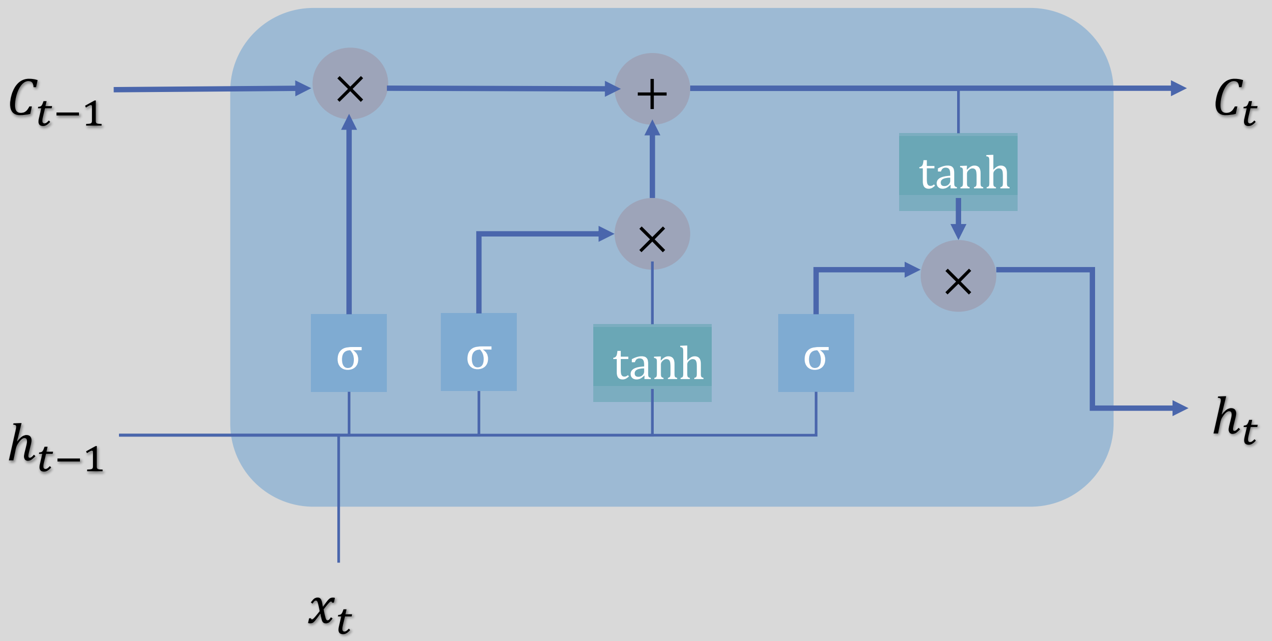


$$h_t = \tanh(W_x x_t + b_x + W_h h_{t-1} + b_h)$$

$$y_t = \varphi(W_y h_t + b_y)$$



Long Short Term Memory Network (LSTM)



$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

Training of CAE+LSTM

While $|L_i - L_{i+1}| < \epsilon_{treshold}$:

Sample Snapshots of solution of the system: q

Forward pass through autoencoder:

$$\hat{q} = h_{dec} \circ f_{lstm} \circ h_{enc}(q)$$

Compute Loss:

$$L = ||\hat{q} - q||_2^2$$

Update model parameters:

$$\theta_{i+1} = \theta_i - \alpha \nabla_\theta L$$

Online stage:

Project input features to latent space:

$$\xi_0 = h_{enc}(\tilde{q}_m)$$

Predictions on latent variable:

$$\xi_i = f(\xi_0; \omega),$$
$$i = \{1, .., n\}$$

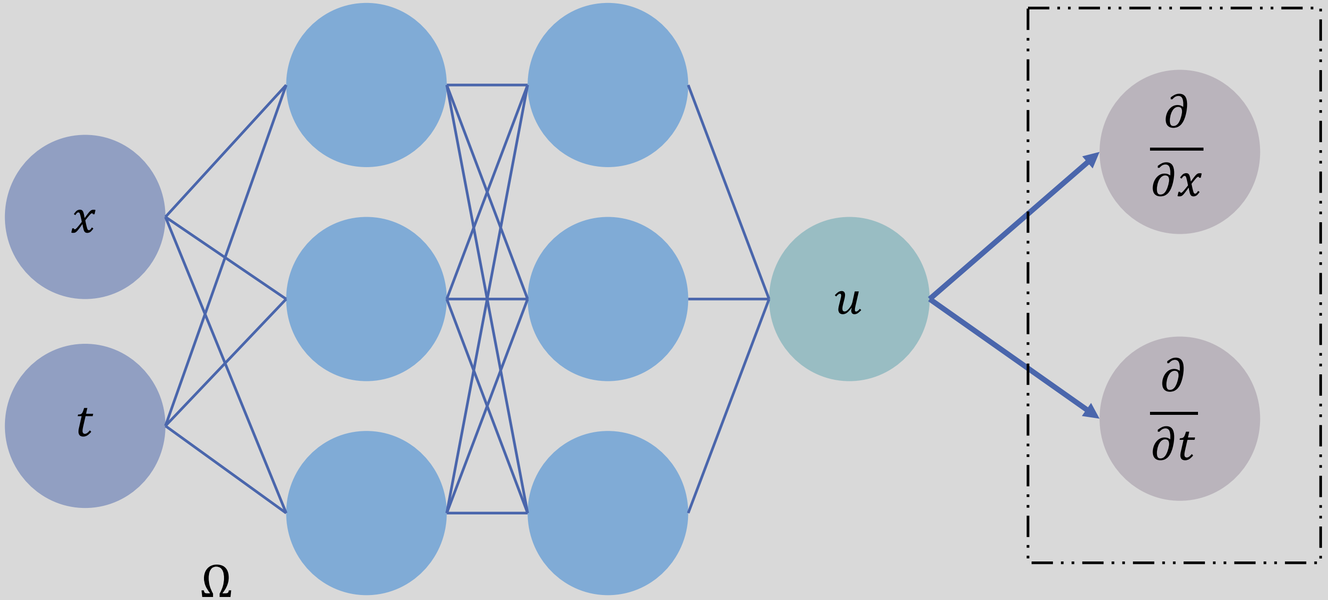
Reconstrcut to original variable space:

$$\hat{q}_{ML} = h_{dec}(\xi_i)$$

Physics-Informed DeepONet

A primer on Physics-Informed Neural Networks

Computed Using Backpropagation

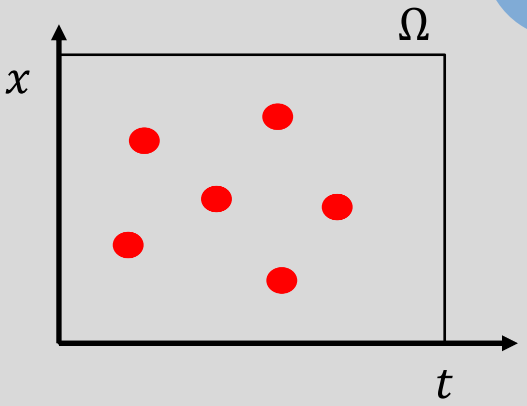


$$\frac{\partial u}{\partial t} + u.\frac{\partial u}{\partial x} = f(x)$$

$$u(x,0) = u_0$$

$$u(x^{\Omega}, t) = u_{\Omega}, \qquad x \in \Omega$$

$$L = L_{data} + L_{ics} + L_{BC} + L_{res}$$



$$L_{data} = \|u(x_i,t_i) - NN(x_i,t_i)\|_2^2$$
$$L_{ics} = \|u(x_i,0) - NN(x_i,0)\|_2^2$$
$$L_{bcs} = \left\|u\left(x^{\Omega},t_i\right) - NN\left(x^{\Omega},t_i\right)\right\|_2^2$$
$$L_{res} = \|NN_t(x_i,t_i) + NN(x_i,t_i)\ .\ NN_x(x_i,t_i) - f(x)\|_2^2$$

Finite Element Method vs. Physics-Informed Neural Networks

	PINN	FEM
Basis function	Neural Network	Piecewise polynomial
Parameters	Weights and Biases	Point Values
Discretization	Scattered Points (Mesh-free)	Mesh points
PDE Embedding	Loss Function	Algebraic System
Parameter Solver	Gradient Based Optimizer	Linear Solver
Errors	$\varepsilon_{app}, \varepsilon_{gen}, \varepsilon_{opt}$	Approximation/quadrature errors
Error bounds	Not available yet	Partially available

Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2021). DeepXDE:
A Deep Learning Library for Solving Differential Equations.
SIAM Review, *63*(1), 208–228.
https://doi.org/10.1137/19M1274067

Physics Informed Machine Learning: Learning Operators

Let A, U are Bancah Spaces of functions defined on bounded Domains $D \subset \mathbb{R}^{\mathrm{d}}, D' \subset \mathbb{R}^{\mathrm{d}'}$ respectively.

Suppose we have acces only to observations $\{a_j, u_j\}_{j=1}^{N}$

Where $a_j \sim \mu$ are samples drawn from some probability measure supported on A .

And $u_j = G(a_j)$ is possibly corrupted with noise.

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \mathbb{R}^p$$

Learning the solution operator of a PDE

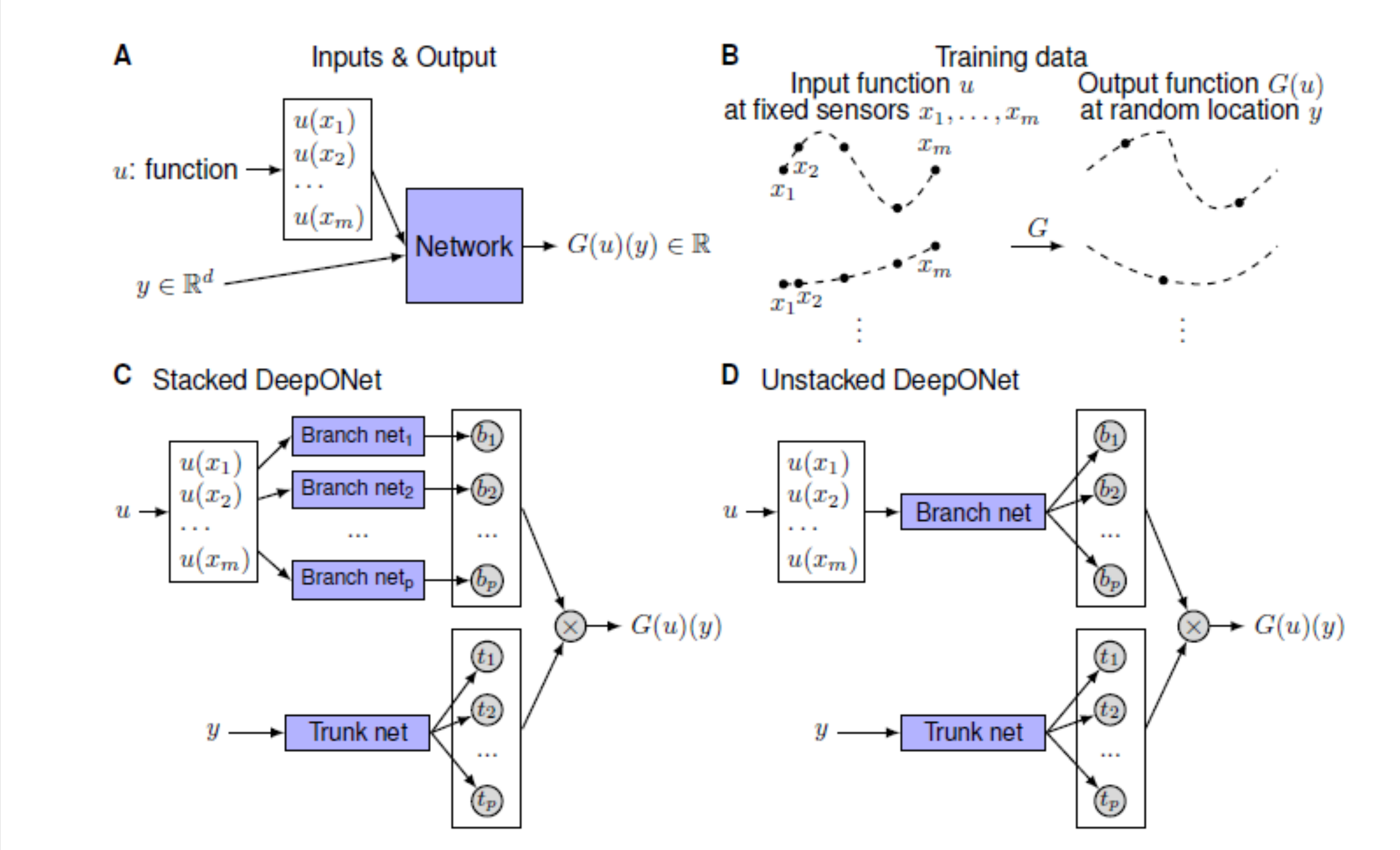
Given a non-linear PDE like:

$$\frac{du}{dt} + N(u) = f(x)$$

We can approximante the solution operator for the PDE that solves the initial value problem for a distribution of initial conditions:

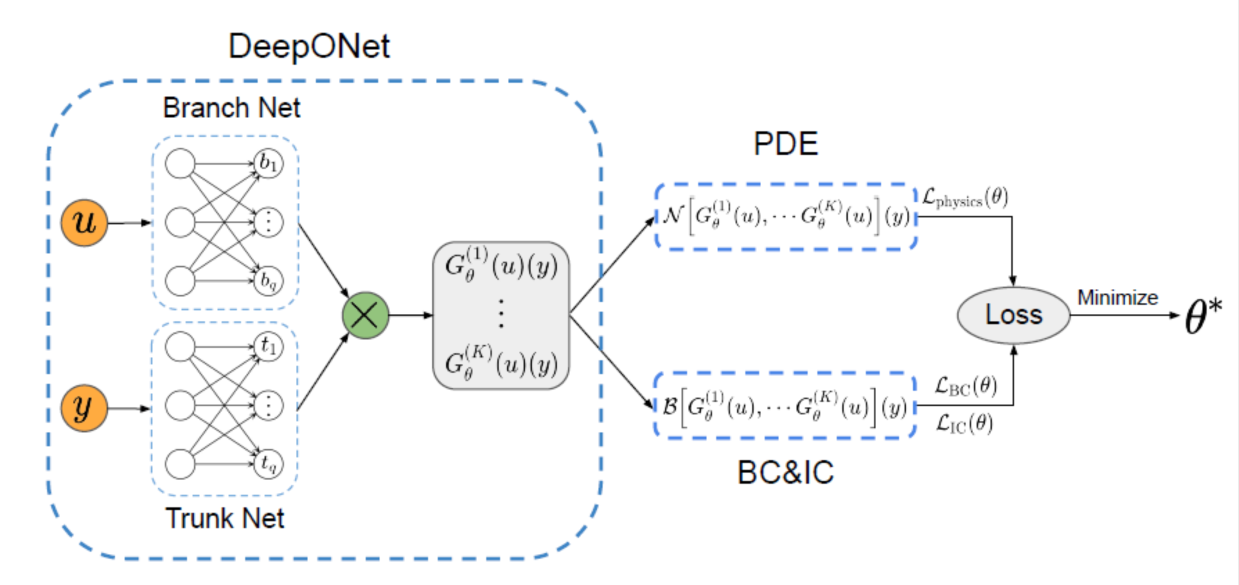
$$G_\theta : u_0 \to u(x, t + \Delta t), \qquad u_0 \sim p(\mu) \in A$$

Method: DeepONet



Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3), 218–229. https://doi.org/10.1038/s42256-021-00302-5

Physics Informed DeepONet

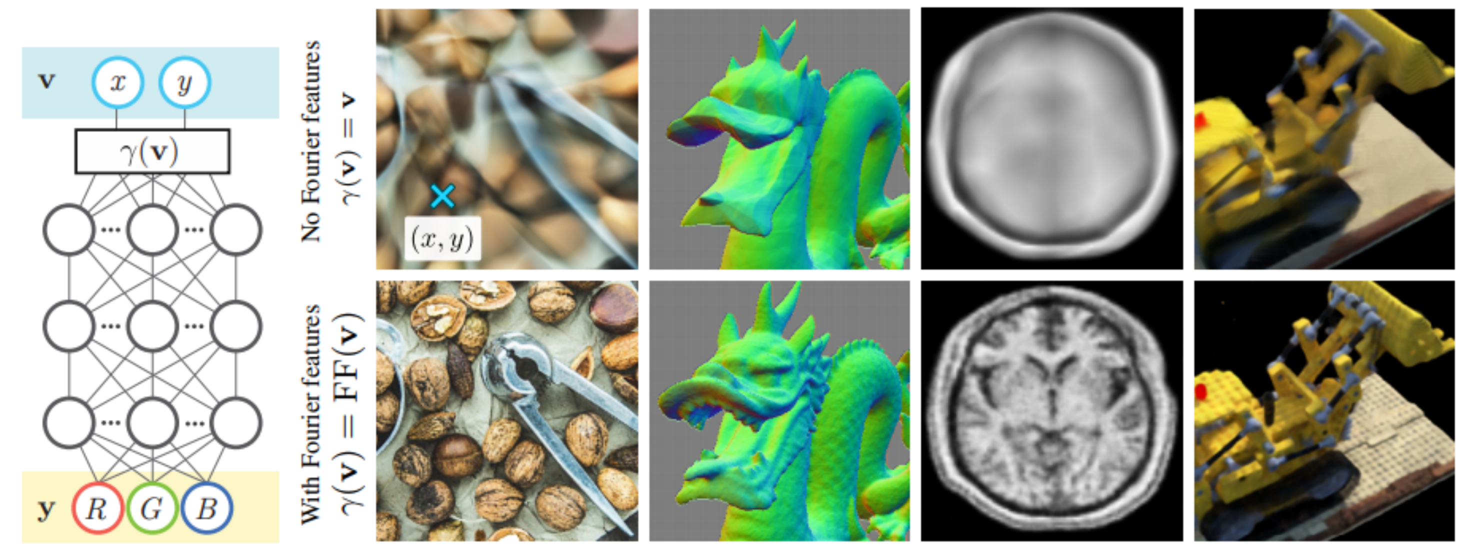


$$G_{\boldsymbol{\theta}}(\boldsymbol{u})(\boldsymbol{x},t) = \sum_{k=1}^{q} \underbrace{b_k(\boldsymbol{u}(\boldsymbol{x}_1),\boldsymbol{u}(\boldsymbol{x}_2),\ldots,\boldsymbol{u}(\boldsymbol{x}_m))}_{\text{branch}} \underbrace{t_k(\boldsymbol{x},t)}_{\text{trunk}},$$

$$\mathcal{R}_{\boldsymbol{\theta}}[\boldsymbol{u}](\boldsymbol{x},t) = \frac{\partial G_{\boldsymbol{\theta}}(\boldsymbol{u})(\boldsymbol{x},t)}{\partial t} + \mathcal{N}_{\boldsymbol{x}}[G_{\boldsymbol{\theta}}(\boldsymbol{u})](\boldsymbol{x},t)$$

Wang, S., Wang, H., & Perdikaris, P. (2021). Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. ArXiv:2103.10974 [Cs, Math, Stat]. http://arxiv.org/abs/2103.10974

Neural Network Architecture: Fourier Feature Network

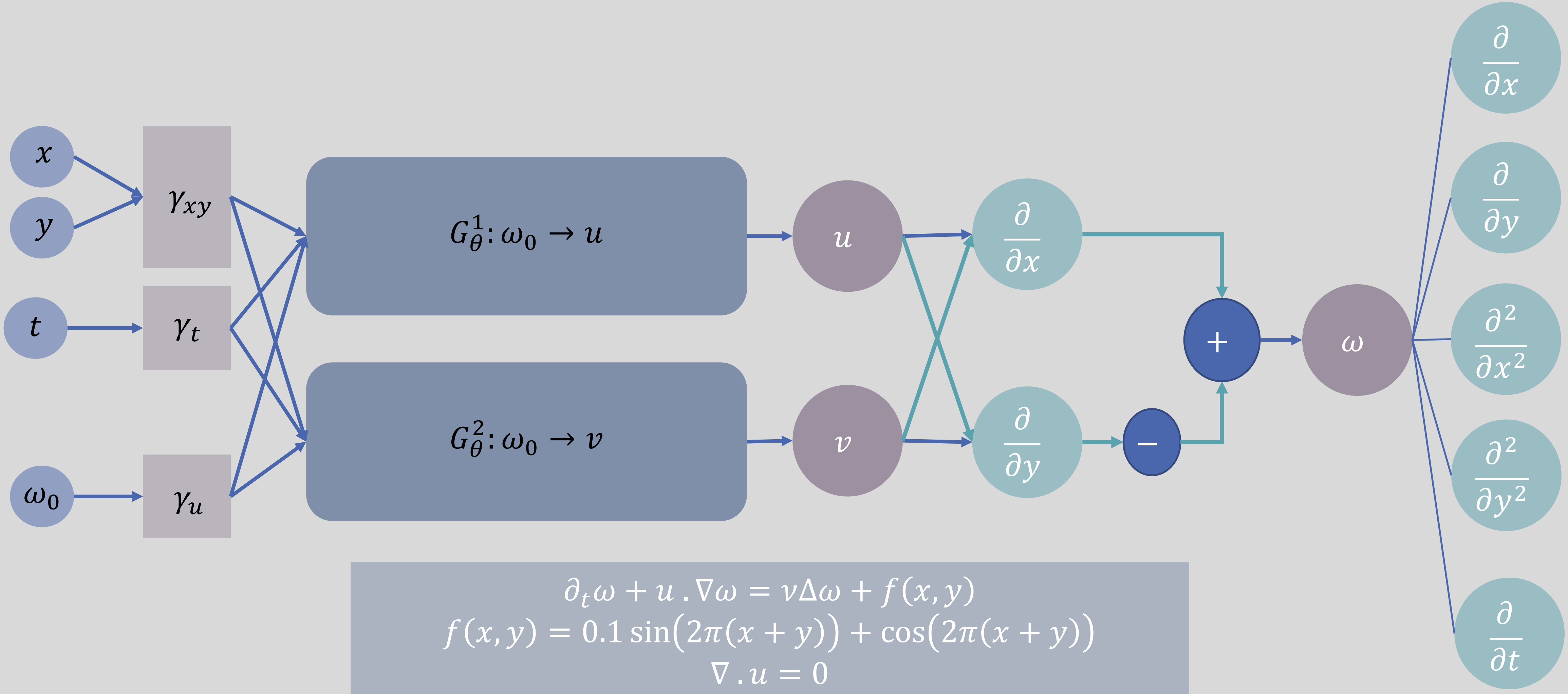


(a) Coordinate-based MLP (b) Image regression $(x,y) \to$ RGB (c) 3D shape regression $(x,y,z) \to$ occupancy (d) MRI reconstruction $(x,y,z) \to$ density (e) Inverse rendering $(x,y,z) \to$ RGB, density

$$\gamma(\mathbf{v}) = \left[a_1 \cos\left(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}\right), a_1 \sin\left(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}\right), \ldots, a_m \cos\left(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v}\right), a_m \sin\left(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v}\right)\right]^{\mathrm{T}}$$

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., & Ng, R. (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. ArXiv:2006.10739 [Cs]. http://arxiv.org/abs/2006.10739

PI-DeepONet for the N-S equation

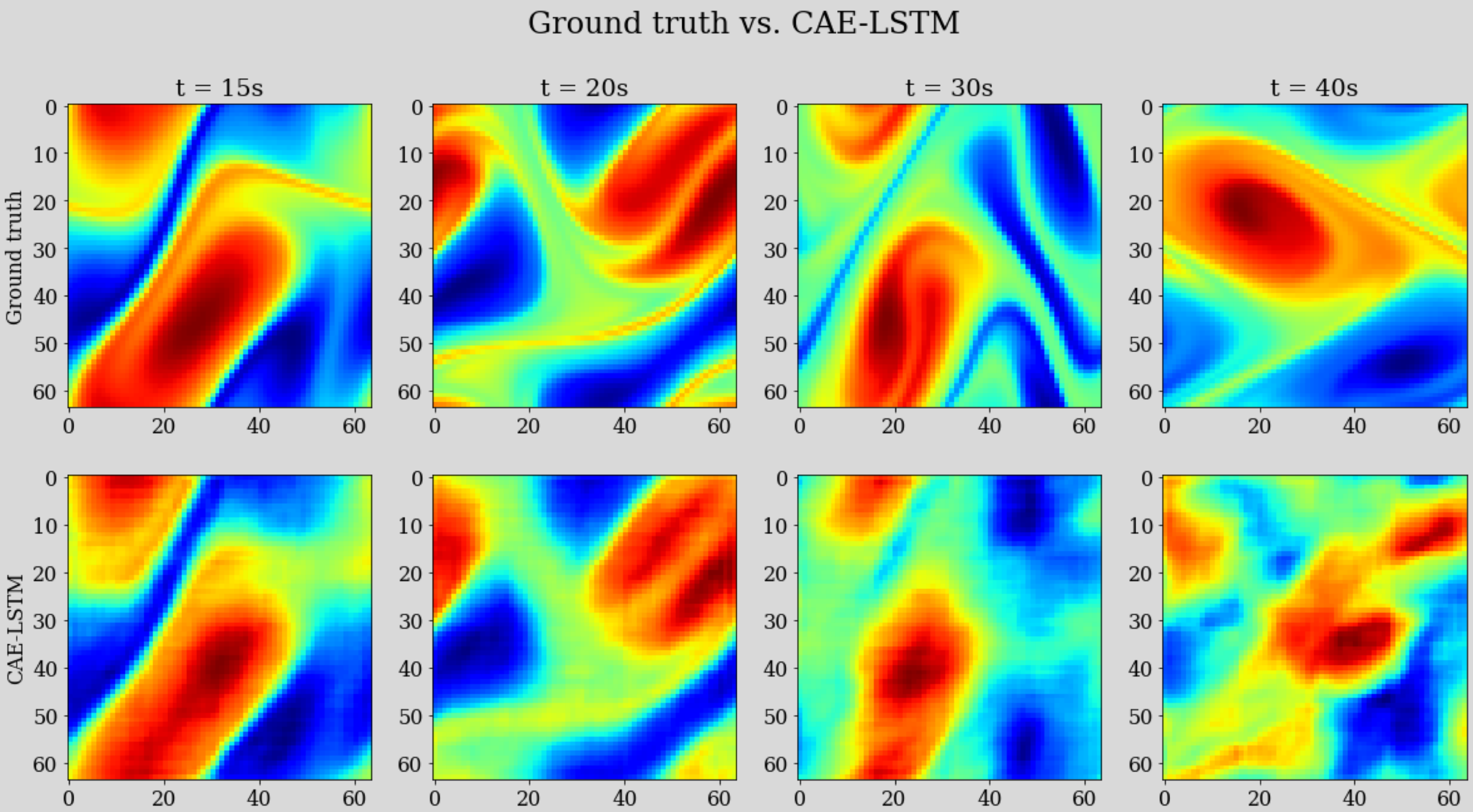


$$\partial_t \omega + u\,.\nabla\omega = \nu\Delta\omega + f(x,y)$$
$$f(x,y) = 0.1\sin\big(2\pi(x+y)\big) + \cos\big(2\pi(x+y)\big)$$
$$\nabla\,.\,u = 0$$
$$\omega(x,y,0) = \omega_0$$

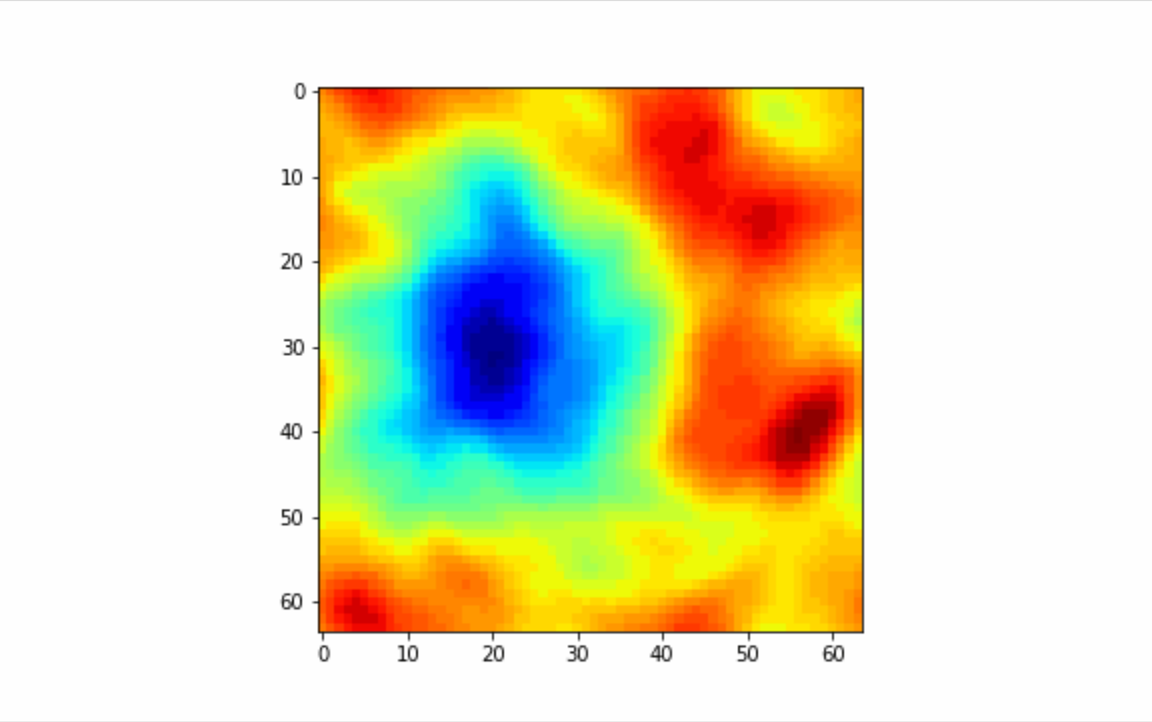
Results

Results for CAE-LSTM

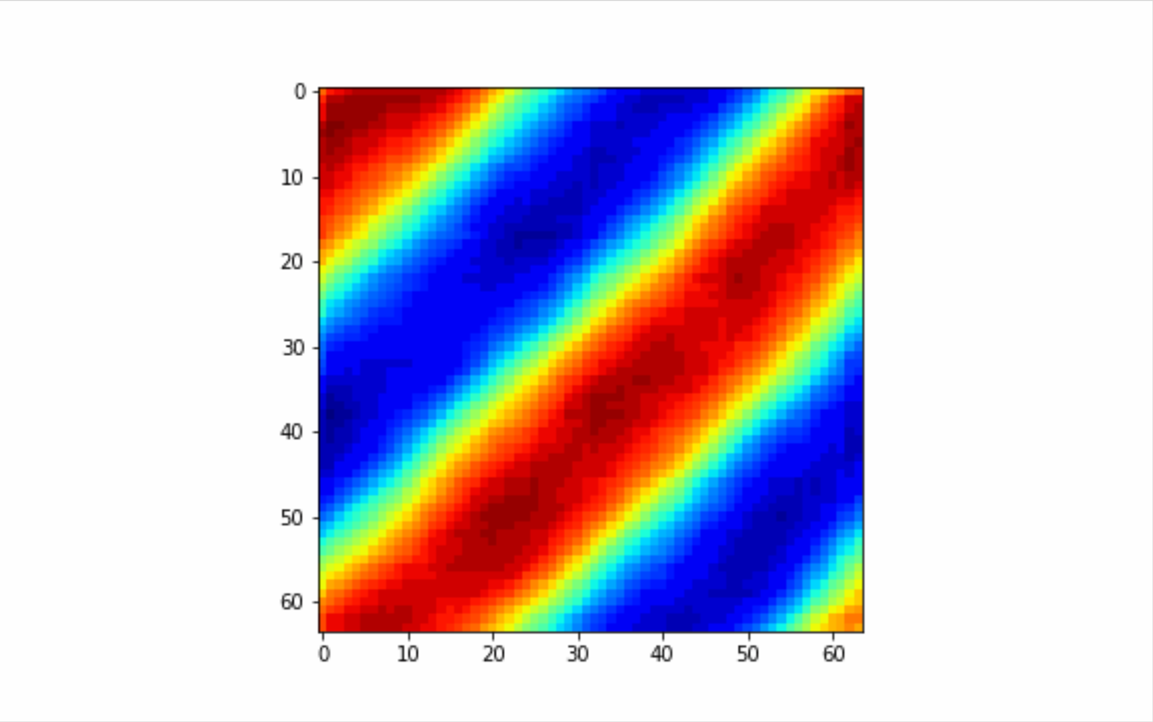
$$\nu = 1e-4$$



Results for CAE-LSTM



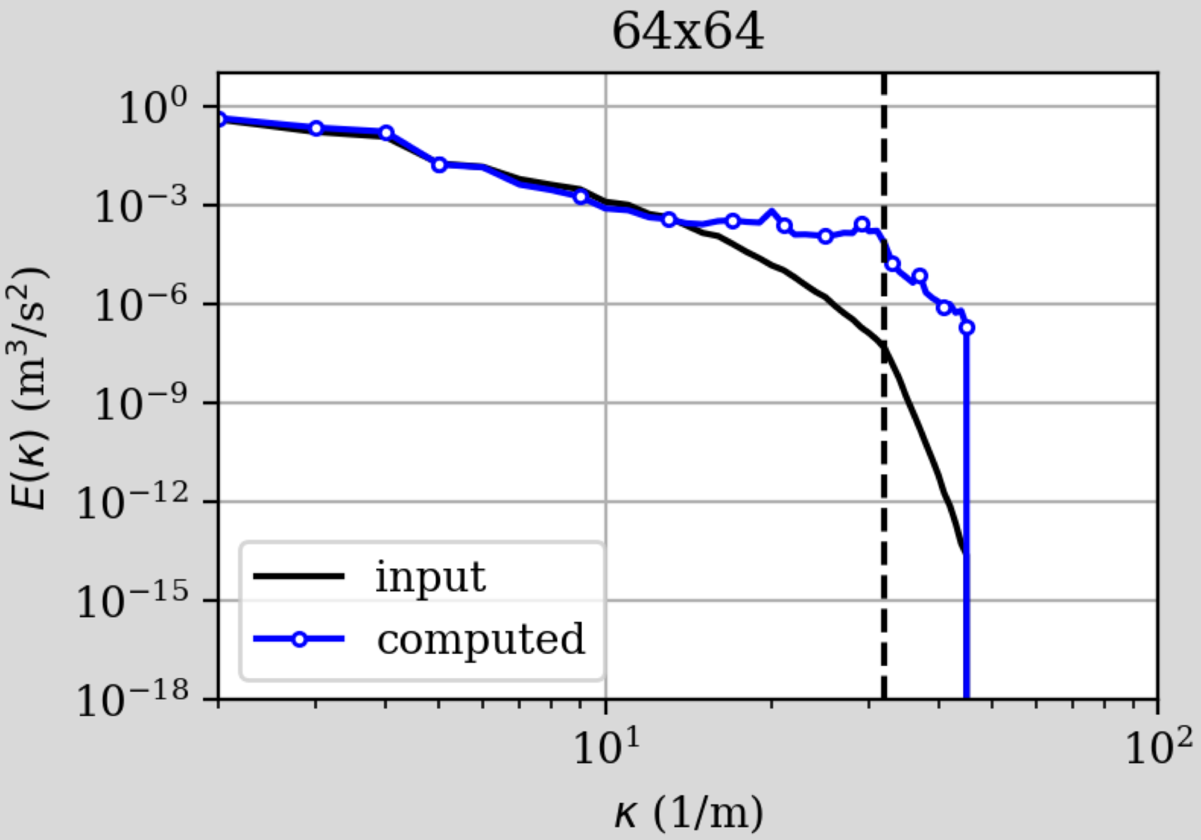
Ground truth



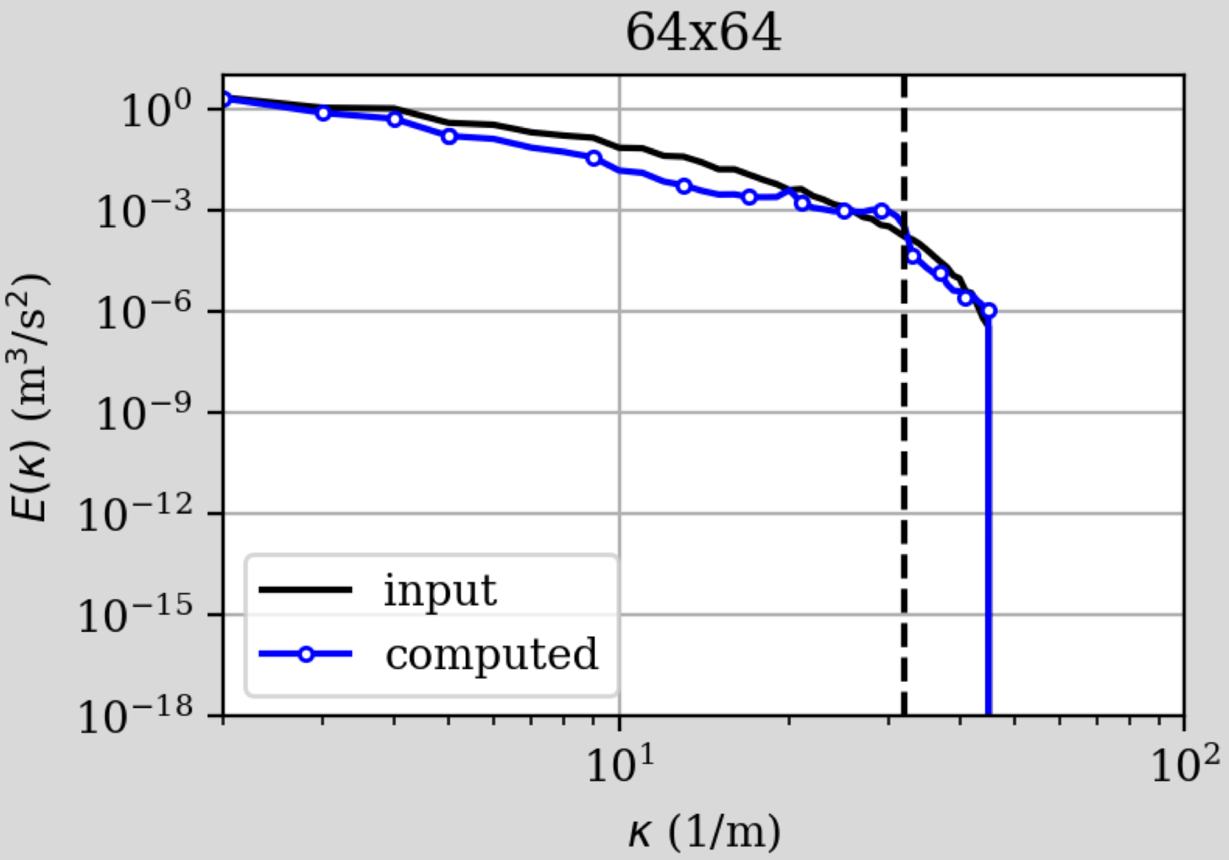
Prediction

Results for CAE-LSTM

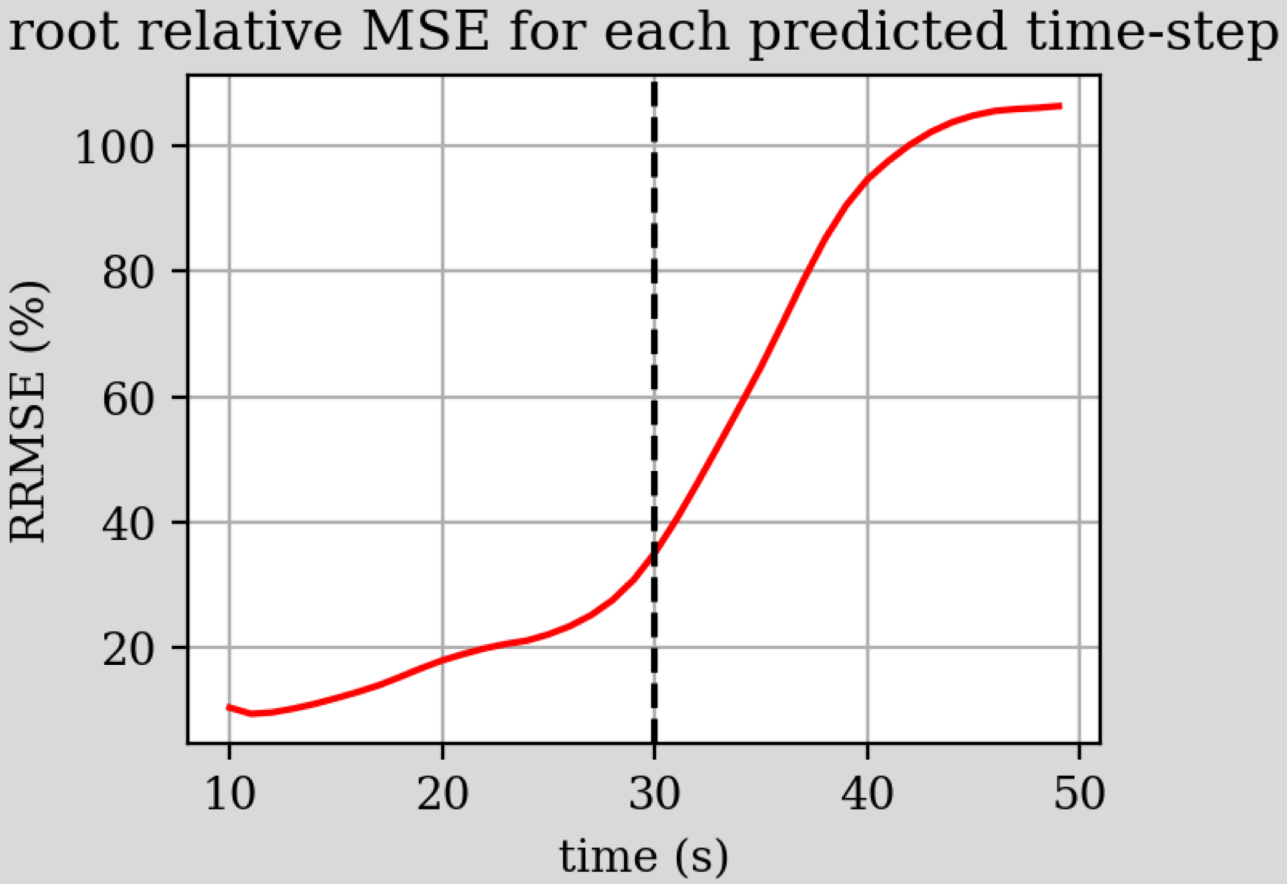
$t = 11s$



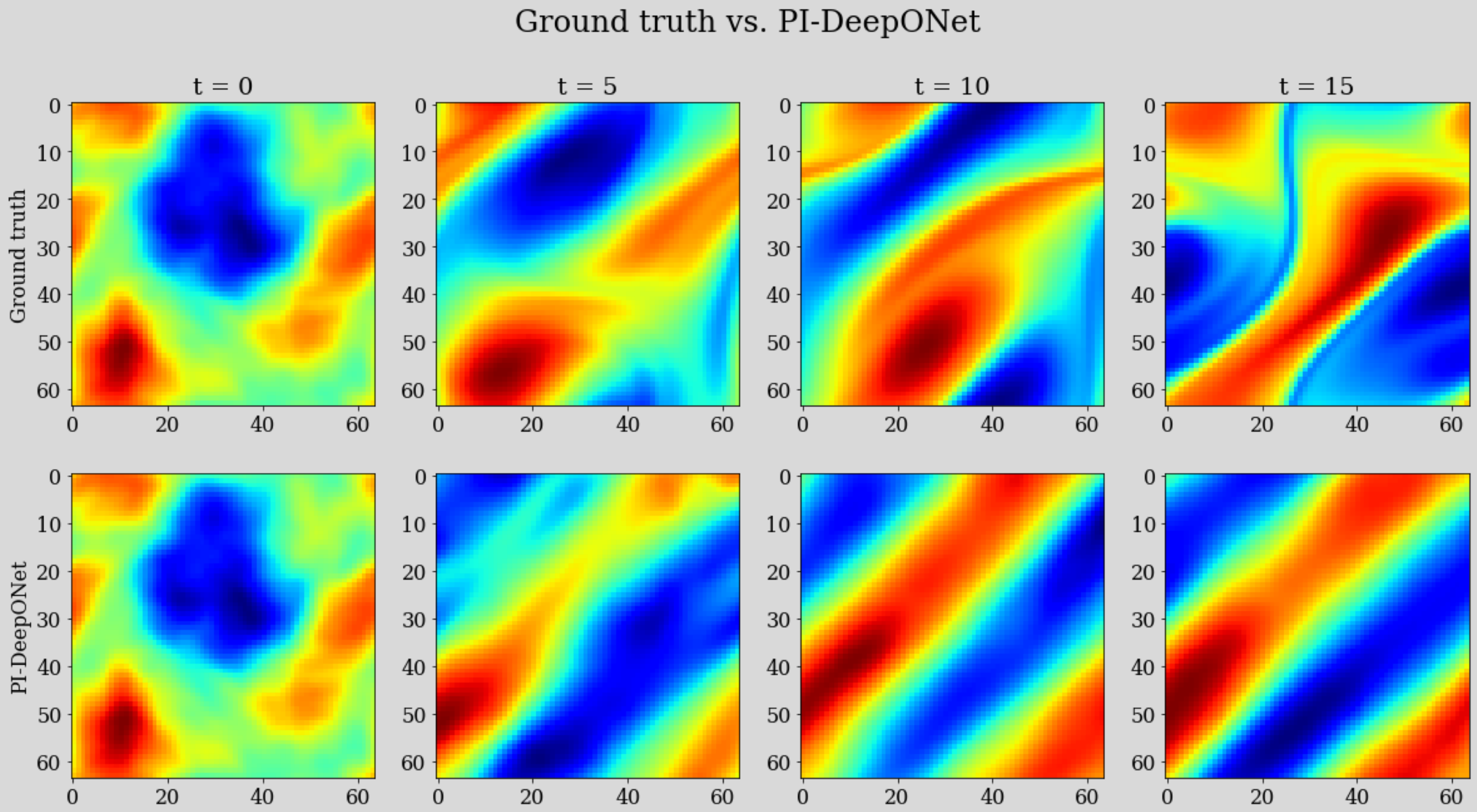
$t = 30s$



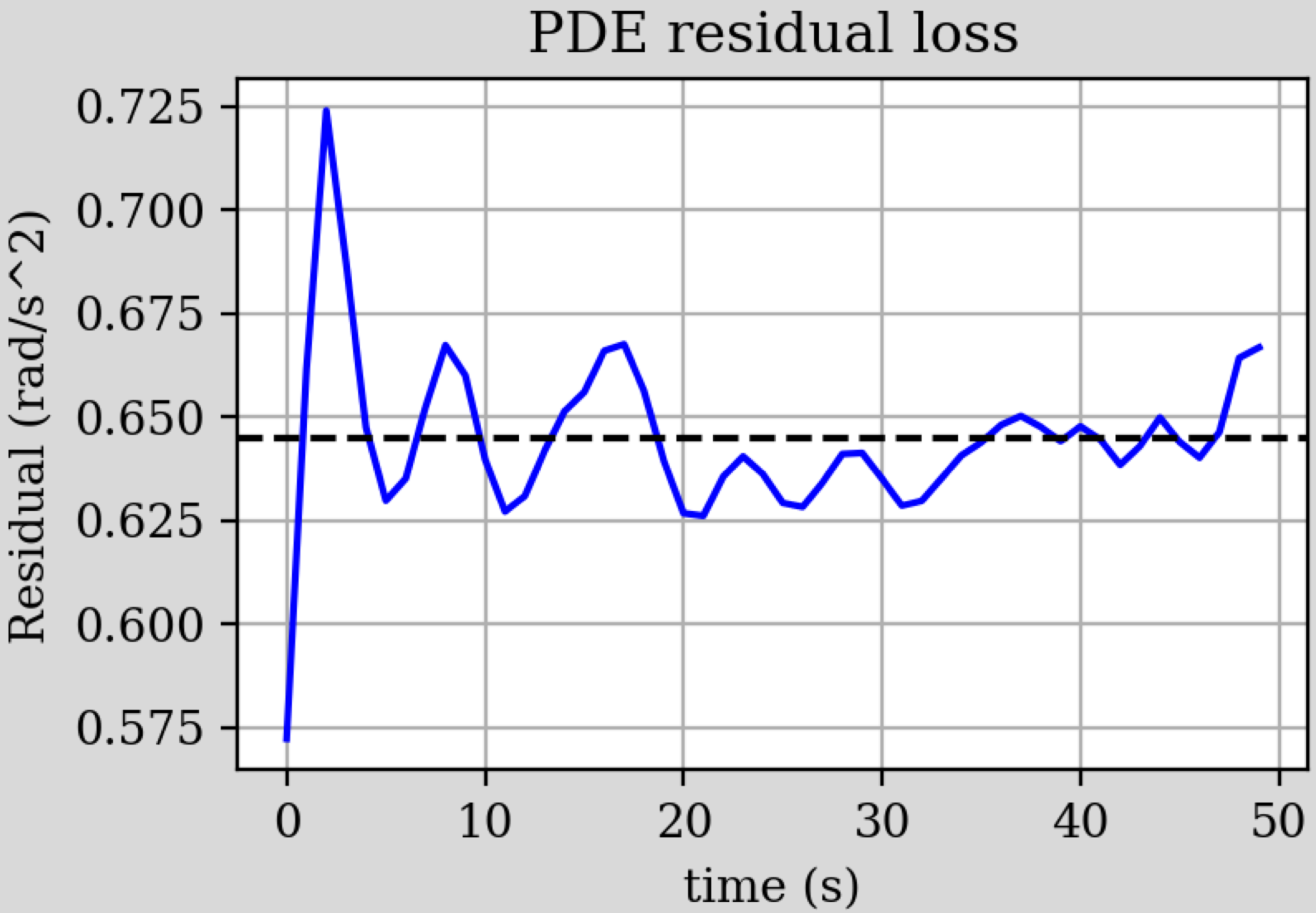
Results for CAE-LSTM



Results for PI-DeepONet

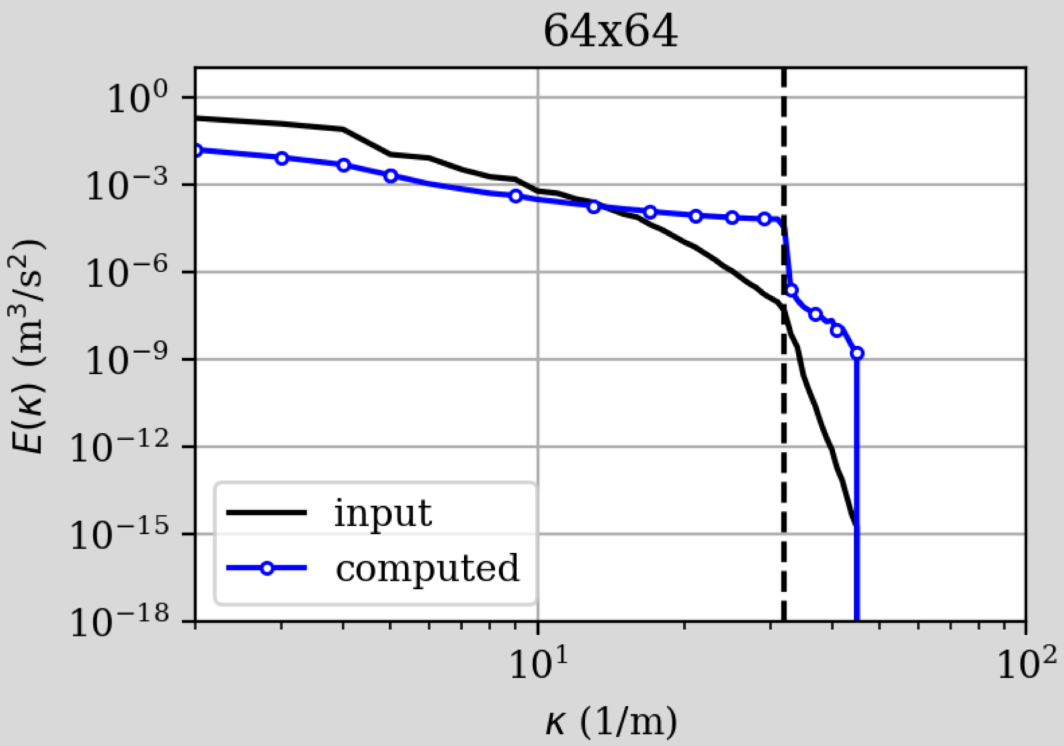


Results for PI-DeepONet

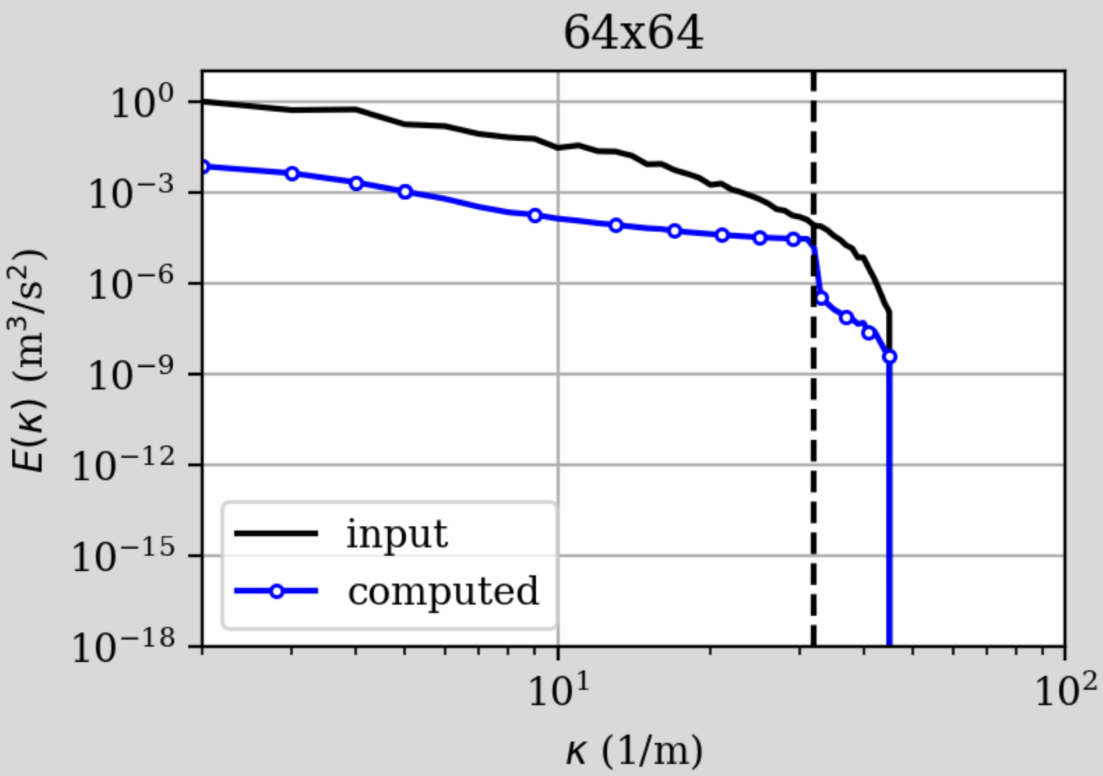


Results for PI-DeepONet

$t = 11s$



$t = 30s$



Limitations

CAE-LSTM	PI-DeepONet
• Dependent on discretization • Difficulty for long-term predictions. • No physical knowledge imposed to the model. • Needs lots of data	• Tricky to train • Not accurate enough (yet!) • Many parameters to tune, including the type of architecture. • Long training times

Perspectives

1. Try other architectures for Neural Operators (Fourier Neural Operator, Graph Neural Operator, etc) and use them with PINNs.

2. Different training algorithms could improve results, for example recurrent training or transfer learning.

3. Use physics Informed Neural Operators for a 3D turbulent case.

4. Apply these methods in a CFD application.

Thank you for listening