CEA-EDF-INRIA Numerical Analysis Summer School 2025
Solving PDE in fields physics faster with physics-based machine learning

Machine Learning for Physical Dynamics, an Introduction

Patrick Gallinari, https://pages.isir.upmc.fr/gallinari/
patrick.gallinari@sorbonne-universite.fr, p.gallinari@criteo.com

Outline

- Context:
  - AI4science
- Background
  - Neural networks and ordinary differential equations
- NNs as surrogate models for solving PDEs and modeling Spatio-temporal dynamics
  - Focus: data-driven approaches
    - Discrete space models – 3 examples
      - ResNets, Graph Neural Networks, Transformers
    - Neural operators & Continuous space models– 3 examples
      - Frequential representations
      - Implicit Neural Representation
      - Attention mechanisms + Transformers

Context: AI4Science

AI4Science as a new scientific paradigm

- Paradigm shift: from explicit formulation to implicit knowledge discovery
  - Emerged in 2018 – rapidly growing field
  - Involves many scientific communities

AI4Science paradigm changes

How research is done: From hypothesis generation to data analysis, experimentation, and discovery.

The questions we can ask: Enabling exploration of complexity and scale previously impossible.

The pace of discovery: Accelerating insights in fields like drug discovery, material science, climate modeling, and fundamental physics.

Context - AI for Science - AI as digital twins
Weather forecasting

2022-2024 – Foundation Models for weather prediction (ERA5 dataset 40 years hourly reanalysis data)

GraphCast – Google & DeepMind 2022
https://arxiv.org/abs/2212.12794

ClimaX – Msoft & UCLA 2023
https://arxiv.org/abs/2301.10343

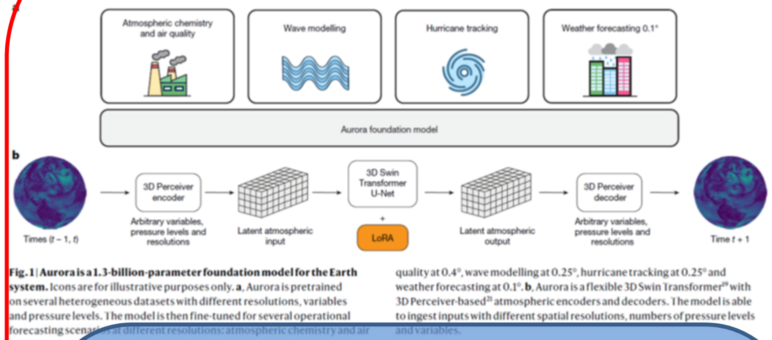
Pangu-Weather – Huawei 2023
http://arxiv.org/abs/2211.02556

FourCastNet – NVIDIA&Lawrence Berkeley lab.&al. 2022
http://arxiv.org/abs/2202.11214

Neural General Circulation Model – Google 2023
https://arxiv.org/abs/2311.07222

Aurora – Microsoft 2024
https://arxiv.org/abs/2405.13063

Aurora (Bodnar et al. 2024)



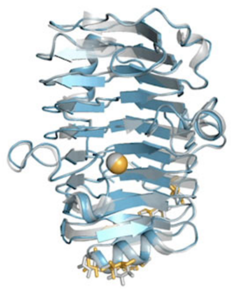
- Integrated Forcasting System Numerical simulation (ECMWF): 65 minutes on 352 high-end CPU for a 10-day forecast
- Aurora:
  - Inference: less than 1 mn on one A100 GPU roughly a ×5,000 speedup over IFS
  - Pre-training, 2.5 weeks on 32 A100

Context - AI for Science – Biology & Drug design

- **Alphafold: Tertiary protein structure prediction**
  - (2018) - Several modules trained separately
  - (2020) - Evoformer, end-to-end training
  - (2024) - Pairformer Structure of protein with DNA, RNA, ligands
  - Alphafold server

Fig - Google DeepMind Predicted enzyme structure (blue) and experimental structure (gray)

6



- **Drug design:** Speed up the drug discovery process
  - Designing molecules/ compounds with high binding affinity to given pathogenic protein targets
  - Inhibitor compounds against tubercolosis

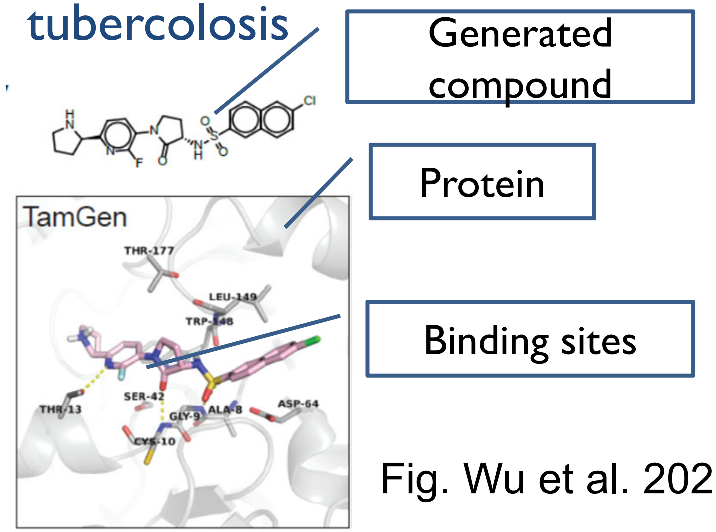


Fig. Wu et al. 2025

Context - AI for Science AI as a reasoning engine - AI4Math - Example MATH Benchmark

- Dataset
  - 12.5K problems from high school competitions + large pretraining dataset

Problem: The equation $x^2 + 2x = \overline{i}$ has two complex solutions. Determine the product of their real parts.
Solution: Complete the square by adding 1 to each side. Then $(x+1)^2 = 1 + i = e^{\frac{i\pi}{4}}\sqrt{2}$, so $x + 1 = \pm e^{\frac{i\pi}{8}}\sqrt[4]{2}$. The desired product is then $\left(-1+\cos\left(\frac{\pi}{8}\right)\sqrt[4]{2}\right)\left(-1-\cos\left(\frac{\pi}{8}\right)\sqrt[4]{2}\right) = 1 - \cos^2\left(\frac{\pi}{8}\right)\sqrt{2} = 1 - \frac{\left(1+\cos\left(\frac{\pi}{4}\right)\right)}{2}\sqrt{2} = \boxed{\frac{1-\sqrt{2}}{2}}$.

- Models: LLMs (Yang et al. 2024)

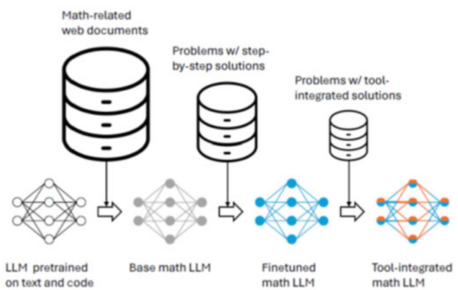


Figure 1: State-of-the-art math LLMs such as NuminaMath [49] typically undergo three stages: math pretraining, finetuning on step-by-step solutions, and further finetuning on tool-integrated solutions that interleave natural language reasoning with Python tool invocation.

Performances

- 2021 (Hendriycks et al. 2021)

Model	Prealgebra	Algebra	Number Theory	Counting & Probability	Geometry	Intermediate Algebra	Precalculus	Average
GPT-2 0.1B	5.2	5.1	5.0	2.8	5.7	6.5	7.3	5.4 +0%
GPT-2 0.3B	6.7	6.6	5.5	3.8	6.9	6.0	7.1	6.2 +15%
GPT-2 0.7B	6.9	6.1	5.5	5.1	8.2	5.8	7.7	6.4 +19%
GPT-2 1.5B	8.3	6.2	4.8	5.4	8.7	6.1	8.8	6.9 +28%
GPT-3 13B*	4.1	2.4	3.3	4.5	1.0	3.2	2.0	3.0 −44%
GPT-3 13B	6.8	5.3	5.5	4.1	7.1	4.7	5.8	5.6 +4%
GPT-3 175B*	7.7	6.0	4.4	4.7	3.1	4.4	4.0	5.2 −4%

Table 2: MATH accuracies across subjects. '*' indicates that the model is a few-shot model. The character 'B' denotes the number of parameters in billions. The gray text indicates the *relative* improvement over the 0.1B baseline. All GPT-2 models pretrain on AMPS, and all values are percentages. GPT-3 models do not pretrain on AMPS due to API limits. Model accuracy is increasing very slowly, so much future research is needed.

- 2025 vals.ai/benchmarks

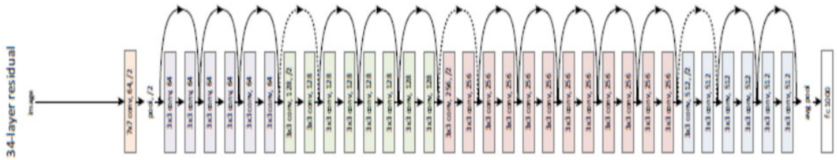
	Model (44)	Accuracy	Cost In / Out	Latency (s)
1	Gemini 2.5 Pro Exp	95.2%	$1.25 / $10.00	25.83 s
2	o3	94.6%	$10.00 / $40.00	16.59 s
3	Qwen 3 (235B)	94.6%	$1.20 / $1.20	142.75 s
4	Grok 3 Mini Fast High Reasoning	94.2%	$0.60 / $4.00	22.77 s
5	o4 Mini	94.2%	$1.10 / $4.40	12.54 s

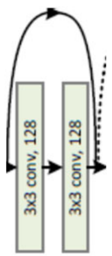
Neural networks and ordinary differential equations

NNs as numerical schemes for solving ODEs

NNs as numerical schemes for solving ODEs

- Several NNs use skip connections, e.g. ResNet





Resnet Module

x_t $\quad x_{t+1} = x_t + f(x_t, \theta_t)$

Input x is progressively modified by a residual $f(x, \theta)$

- ODE for initial value problem
  - $\frac{dx}{dt} = f(x(t); \theta(t))$ for $t \in [0, T]$, $x(0) = x_0$
    - What is the value of $x(T)$?
- Equivalent integral formulation
  - $x(T) = x(0) + \int_O^T f\left(t, x(t)\right) dt$
  - $\int_O^T f\left(t, x(t)\right) dt$ is approximated via numerical integration
  - Exemple: Euler numerical scheme
    - $x_{t+1} = x_t + hf(x_t, \theta_t), x(0) = x_0$

Forward pass of ResNet is similar to Euler scheme for solving IVP (E 2017, Haber 2017, Chang 2018, Lu 2018, …)

NNs as numerical schemes for solving ODEs – Learning problem

- Learning problem with ResNets
  - $Min_\theta \quad L(F(x,\theta),y)$
    $s.t. \quad x_l = x_{l-1} + f_l(x_{l-1}) \,, l = 1 \dots T, x_0 = x$

The constraint describes the Forward computation graph of the Resnet

  - x input, y target, θ parameters, x_l layer l activation, T layers
  - Solving this problem requires alternating
    - Forward pass – Euler numerical scheme for solving
      - $\frac{dx}{dt} = f(x(t),\theta(t))$ for $t \in [0,T], x(0) = x_0$
    - Backward pass – differentiation through Euler scheme for solving
      - $\frac{d\theta}{dt} = -\epsilon \frac{\partial L(\theta(t))}{\partial \theta}, \theta(0) = \theta_0$ &(gradient flow)
- Could this idea be generalized?
  - Replace Euler with any numerical integration scheme, explicit or implicit

NNs as numerical schemes for solving ODEs - Continuous limit

- Continuous limit
  - If we let $h \to 0$ in Euler, the ResNet learning problem becomes
  - $Min_\theta L(F(x,\theta),y)$
    - $s.t. \quad \frac{\partial x}{\partial t} = F\left(x(t),\theta(t)\right), t \in [0,T]\ , x_0 = x$
  - Two different families of methods for solving the learning problem:

Discretize then Optimize (DTO)

- Discretize in time and then solve
- Framework used in this presentation

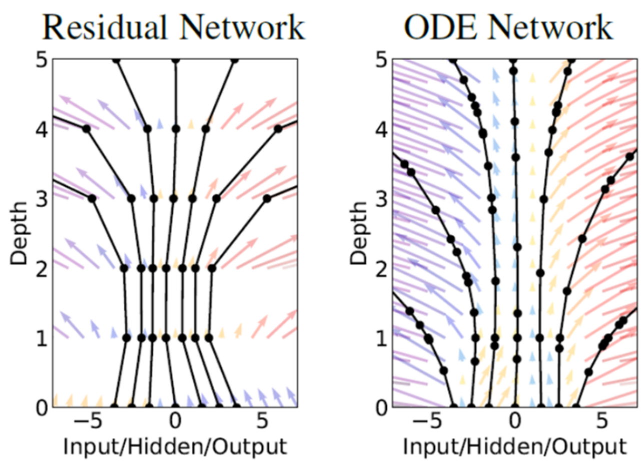


Fig. NeuralODE, Chen et al. 2018

Optimize then Discretize (OTD)

- Solves the continuous optimization problem via adjoint method
- Popularized by NeuralODE (Chen et al. 2018)

NNs as numerical schemes for solving ODEs - summary

- The dynamics of Neural Networks – explained by ODEs
  - NNs with an infinite number of layers can be modeled as ordinary differential equations (ODE)
  - Inference and training can be formulated as solving ODEs

  - In practice
    - This helped popularize the use of differentiable numerical solvers in the ML community
    - They are now implemented in deep learning libraries, e.g. PyTorch
    - Makes possible the integration of numerical solvers and deep learning components in hybrid systems
    - Key for modeling neural solvers

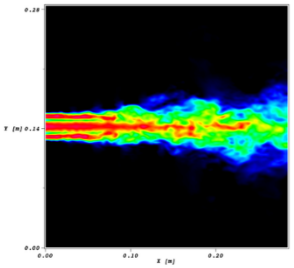
Modeling Spatio-temporal dynamics with Neural Networks

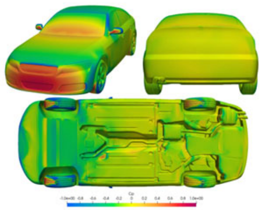
NNs as surrogate models for solving PDEs – Discrete space models
NNs as surrogate models for solving PDEs – Continuous space models & Neural operators

Modeling Spatio-temporal dynamics with Neural Networks Motivations

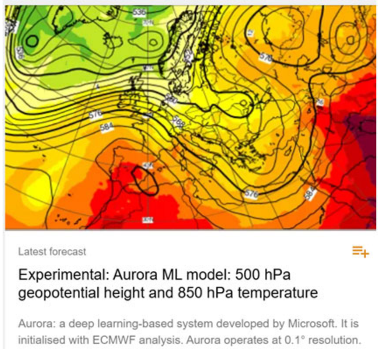
- Applications domains - examples

Computational Fluid Dynamics	Earth System Science – Weather prediction/ Climate	Graphical design

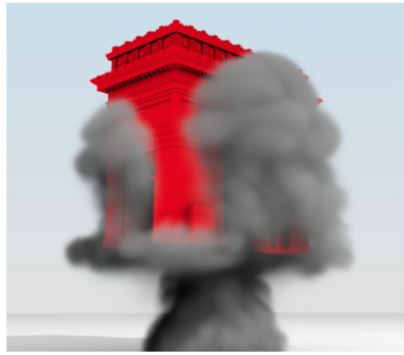




DrivaerNet 2025



ECMWF 2025



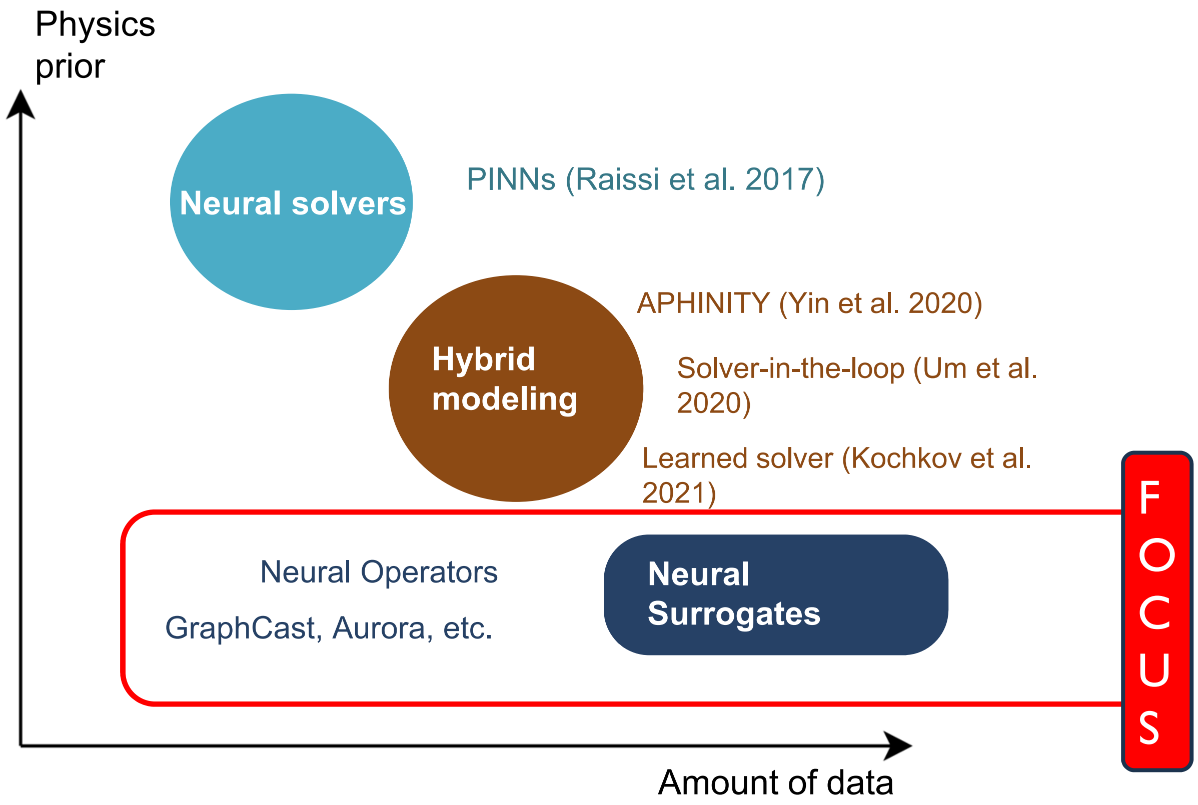
Tompson et al. 2017

- Objectives
  - Alternative to numerical solvers
    - Reduce computational cost – e.g. CFD Surrogate Models/ Reduced Order Models
    - Complement physical models: Hybrid Systems
    - Replace solvers

Modeling Spatio-temporal dynamics with Neural Networks Frameworks

Deep learning landscape for modeling dynamics and solving PDEs

Focus of the presentation: data-driven approaches



Modeling Spatio-temporal dynamics with Neural Networks

- ✓ NNs as surrogate models for solving PDEs – Discrete space models
- NNs as surrogate models for solving PDEs – Neural operators & Continuous space models

NNs as surrogate models for solving PDEs
Discrete space models

- Space discretization is performed a priori under the form of:
  - a regular grid
  - irregular meshes, e.g. in fluid mechanics
- Classical NN models s.a. CNNs, UNets, Graph NNs, Transformers can then be used as time steppers on these discretized representations
- This principle is similar to the method of lines for solving PDEs
  - Perform spatial discretization + algebraic approximation of spatial derivatives
  - Solving the PDE amounts at solving a system of ODEs that can be solved with a numerical ODE solver

NNs as surrogate models for solving PDEs
Discrete space models

- ✓ Learning from partial observations – ResNets –> regular grids
- Message passing PDE solvers – Graph NNs –> irregular meshes
- Transformers – > regular or irregular meshes

NN surrogates – discrete space models – regular grids
Learning from partial observations (Ayed et al. 2019- 2022)

- Forecasting non linear dynamical systems from observations only
  - Assumption: partial observations
    - The state of the system is only partially observed
- Objective
  - Learn the evolution of the system (observations and state) from scratch with a NN

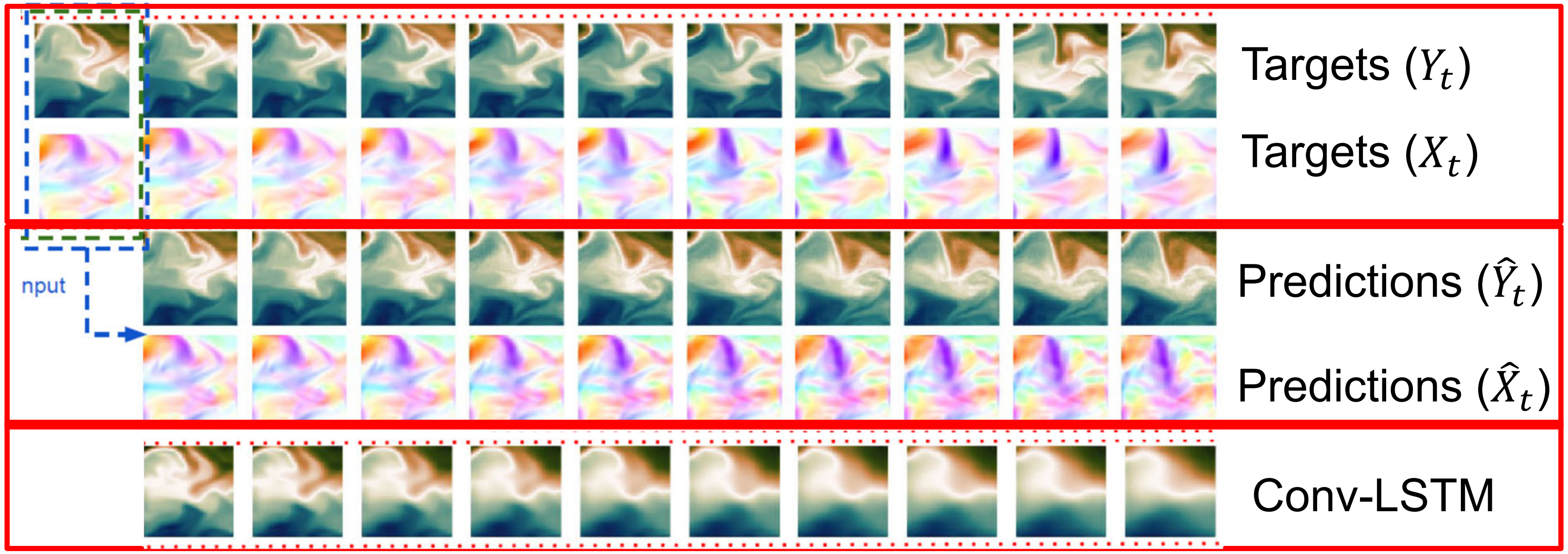
NN surrogates – discrete space models – regular grids
Learning from partial observations (Ayed et al. 2019- 2022)

- Assume an (unknown) underlying dynamical system with initial conditions
  - $$\begin{cases} X_0 & \text{Initial state of the system} \\ \frac{dX_t}{dt} = F^*(X_t) & \text{State dynamics} \\ Y_t = H(X_t) & \text{Observations} \end{cases}$$
- Variables
  - $X_t \in R^d$: state of the system at time t
    - function of time and space, partially observed
    - e.g. 3 D dynamics of the Ocean: velocity, pressure on the ocean surface
  - Y_t : observation, i.e. only available data for training $\{Y_t, 0 \leq t \leq T\}$
    - e.g. satellite observations: temperature, salinity, ocean color, waves height, …
  - H: measurement process linking state to observation is known
  - F^* describes the evolution of the state and is unknown

NN surrogates – discrete space models – regular grids
Learning from partial observations (Ayed et al. 2019- 2022)

- Objective
  - Learn the evolution of the system (observations and state) from scratch with a NN
- Learning problem
  - $\underset{F_\theta, g_\theta}{minimize}\ E_Y[\sum_{t=0}^{T}\|Y_t - H(\hat{X}_t)\|_2^2$ learn **trajectories** from **observations**
  - Subject to $\forall t, \frac{d\hat{X}_t}{dt} = F_\theta(\hat{X}_t),$ learn the **state** dynamics
  - $\hat{X}_0 = g_\theta(Y_{-k}, 0 < k \leq K)$ learn **initial state** from previous observations
- Implementation
  - Evolution function F_θ is implemented as a convolutional ResNet, similar to forward Euler solver for ODEs
    - Solve $\frac{d\hat{X}_t}{dt} = F_\theta(\hat{X}_t)$
      - $X_{t+\delta t}^{\theta} = X_t^{\theta} + \delta t F_\theta(X_t^{\theta})$
  - g_θ is a Unet or a ResNet

- NEMO – Nucleus for European Modelling of the Ocean Engine
  - State: 7 variables, we make use only of 2 variables corresponding to the velocity field
  - Observations: Sea Surface Temperature
  - Initial state: interpolated from previous observations



NN surrogates – discrete space models – regular grids
Learning from partial observations (Ayed et al. 2019- 2022)

- Summary
  - Modeling a state space system with NNs
  - The evolution function is learned in the unobserved state space
  - Better than working directly in the observation space

NNs as surrogate models for solving PDEs
Discrete space models

Learning from partial observations – ResNets – Unets –> regular grids

- ✓ Message passing PDE solvers – Graph NNs –> irregular meshes

Transformers – > regular or irregular meshes

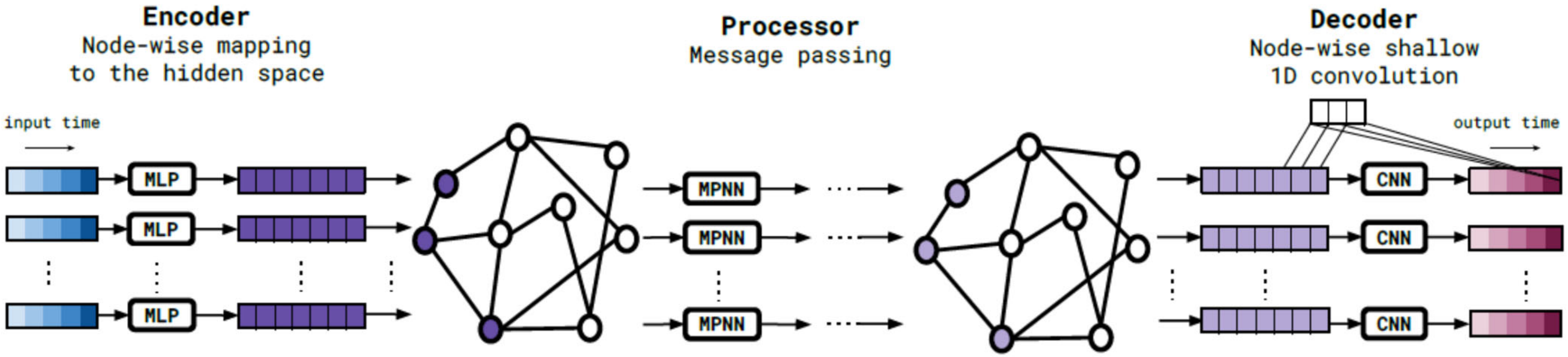
NN surrogates – discrete space models – irregular meshes
Graph Neural Networks

- GNNs are well adapted to handle irregular meshes
  - Mesh nodes are mapped to a graph which is processed with a GNN
    - The GNN acts as a time stepper Neural ODE solver
  - Several efforts for developing neural PDE solvers based on graphs
    - Sanchez-Gonzales et al. 2020, Belbute-Peres et al. 2020, Pfaff et al. 2021, …
    - Large scale implementation: Graphcast (Google 2022)
- Example: Brandstetter et al. 2022 - Message Passing Neural PDE Solvers
  - Objective: forecast spatio-temporal dynamics
    - Auto-regressive model $u(x,t) \to u(x,t+\Delta t) \to u(x,t+2\Delta t)$ ...
    - Representative GNN solver
      - Handle multiple situations
        - Multiple resolutions, boundary problems, parametric PDEs, etc

NN surrogates – discrete space models – irregular meshes
Message Passing Neural PDE Solvers (Brandstetter et al. 2022)

- Framework: Encode-Process-Decode (Sanchez-Gonzales 2020)
  - Process: message passing on the graph node embeddings

Fig. Bransdtetter et al. 2022



Node i, step k		
Encoding Input: last K values at each node i f_i^0 =Encode($u_i^{k-K}, \dots, u_i^k$)	**Process in latent space** M message passing steps $f_i^m, m = 1 \dots M$	**Decode** Output: next K values $u_i^{k+1}, \dots, u_i^{k+K}$

NN surrogates – discrete space models – irregular meshes
Message Passing Neural PDE Solvers (Brandstetter et al. 2022)

- Example
  - Burgers equation $1D$: $\frac{\partial u(t,x)}{\partial t} = -\mathrm{u}\frac{\partial u(t,x)}{\partial x} + v\frac{\partial^2 u(t,x)}{\partial x^2} + f(t,x)$
    - simplified equation for fluid flows, u velocity field, v viscosity coef.

Experiments performed at different resolutions
Color = time

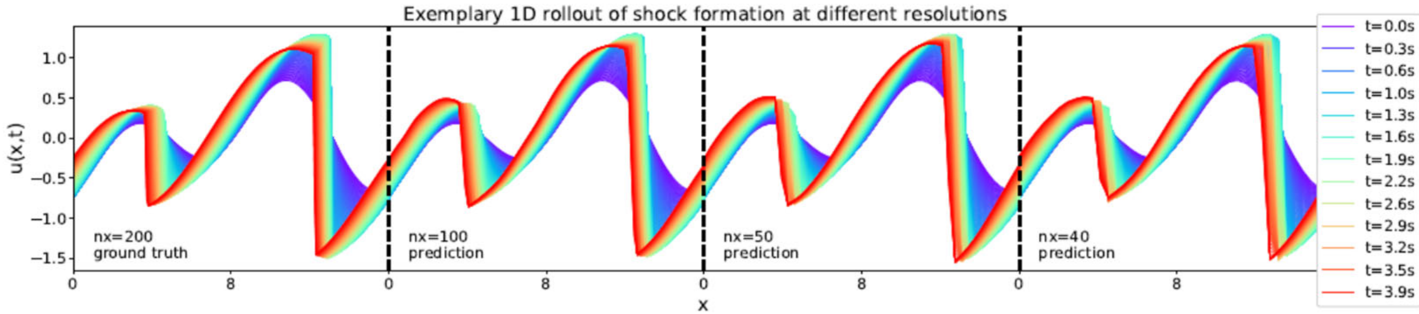


Figure 4: TOP: Exemplary 1D rollout of shock formation at different resolutions. The different colors represent PDE solutions at different timepoints. Both the small and the large shock are neatly captured and preserved even for low resolutions; boundary conditions are perfectly modeled. BOT-

Fig. Branstetter 2022

NNs as surrogate models for solving PDEs
Discrete space models

Learning from partial observations – ResNets – Unets —> regular grids

Message passing PDE solvers – Graph NNs —> irregular meshes

- ✓ Transformers — > regular or irregular meshes

NN surrogates – discrete space models -Transformers

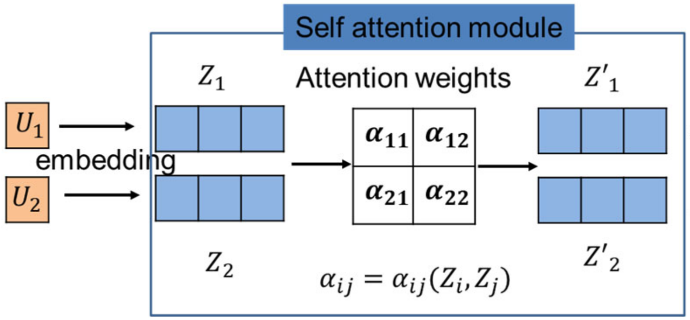
- Why transformers for modeling dynamical systems?
  - They allow us to leverage the recent developments in vision/ NLP for modeling complex dynamics
  - They are the core components of recent neural solvers

- They operate on tokenized representations (vectors) of the input/ output data
  - Either directly in the physical space
  - Most often on a latent representation, leveraging an Encode-Process-Decode framework
  - Architectures are often inspired from Vision Transformers

NN surrogates – discrete space models –Transformers
Attention mechanisms

2 core attention mechanisms used in transformers

Self Attention



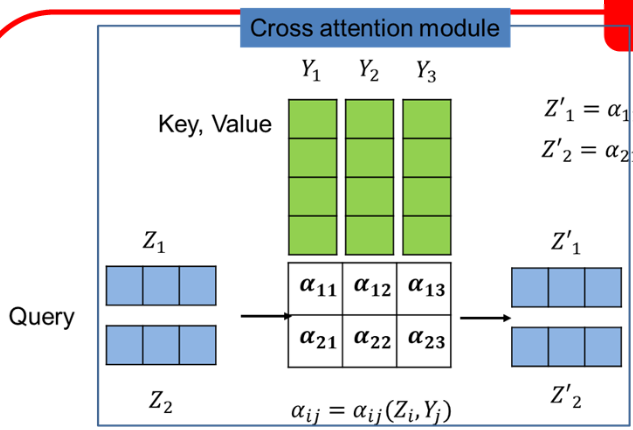
$Z'_1 = \alpha_{11} Z_1 + \alpha_{12} Z_2$

$Z'_2 = \alpha_{21} Z_1 + \alpha_{22} Z_2$

S.A. captures contextual representation of the inputs

Complexity $O(N^2)$ with N the size of the input sequence

Cross Attention



$Z'_1 = \alpha_{11} W Y_1 + \alpha_{12} W Y_2 + \alpha_{13} W Y_3$

$Z'_2 = \alpha_{21} W Y_1 + \alpha_{22} W Y_2 + \alpha_{13} W Y_3$

W projection matrix

Complexity: $O(N(Y)N(Z))$

C.A. maps a sequence of vector (Y) of variable size N into a sequence of vector (Z') of fixed size M

NN surrogates – discrete space models –Transformers
Vision transformers (Dosovitskiy et al. 2021)

ViT
- Splits the image in patches
- Embeds them linearly and feed them to a transformer encoder
- Stacked encoders could be used for time stepping

Fig. Dosovitskiy et al. 2021

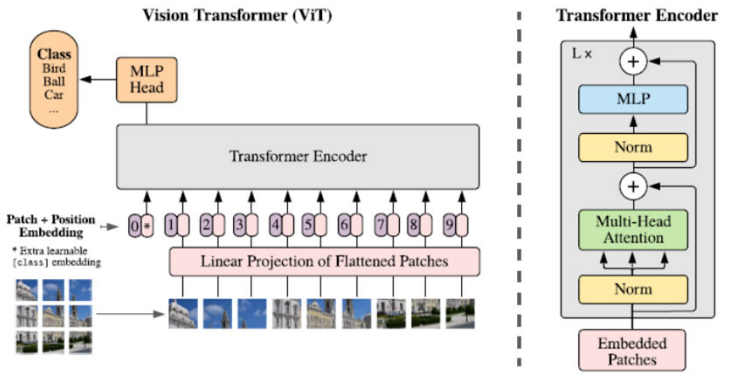
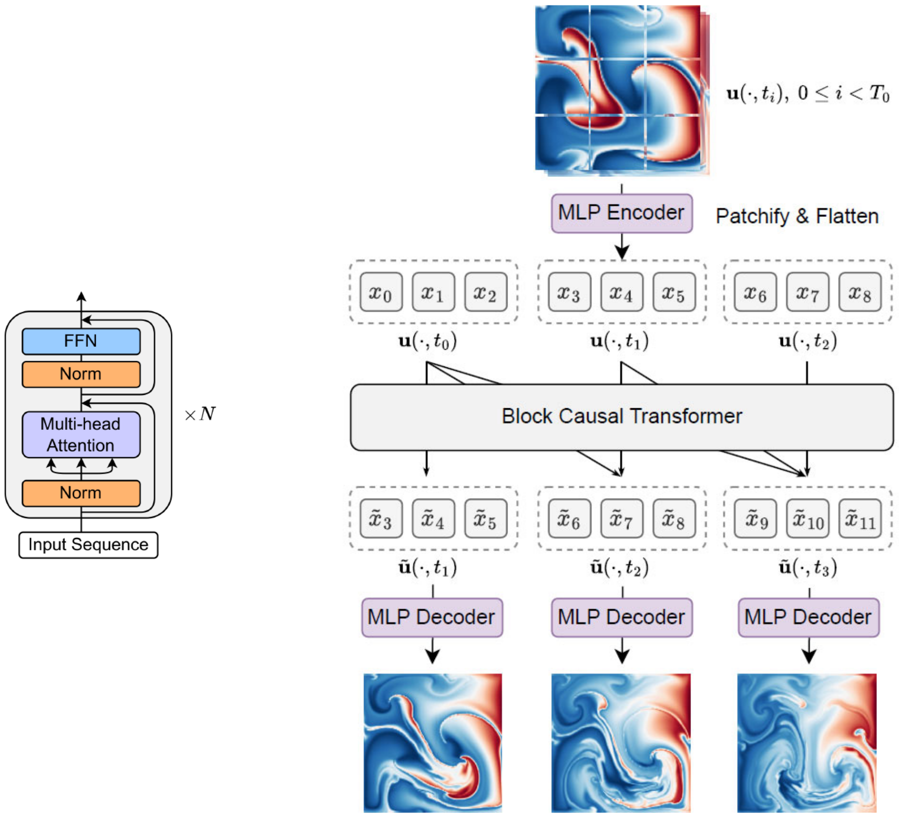


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Lots of transformer/ attention variants in order to
- Adapt to the physics/ Limit the complexity
- Implement the encode/process/decode process
- Examples follow

NN surrogates – discrete space models –Transformers BCAT model (Liu et al. 2025)

- Simple autoregressive transformer (decoder as in GPT/ Llama) for spatio-temporal prediction (sequential)



Patchify & flatten frames

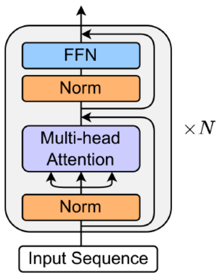
MLP encoder: Sequence of tokens/ frames

Process: Causal GPT like decoder: predicts next frame token sequence

MLP decoder: back to physical space

NN surrogates – discrete space models –Transformers
Transformers - BCAT model (Liu et al. 2025)

- Simple autoregressive transformer for spatio-temporal prediction



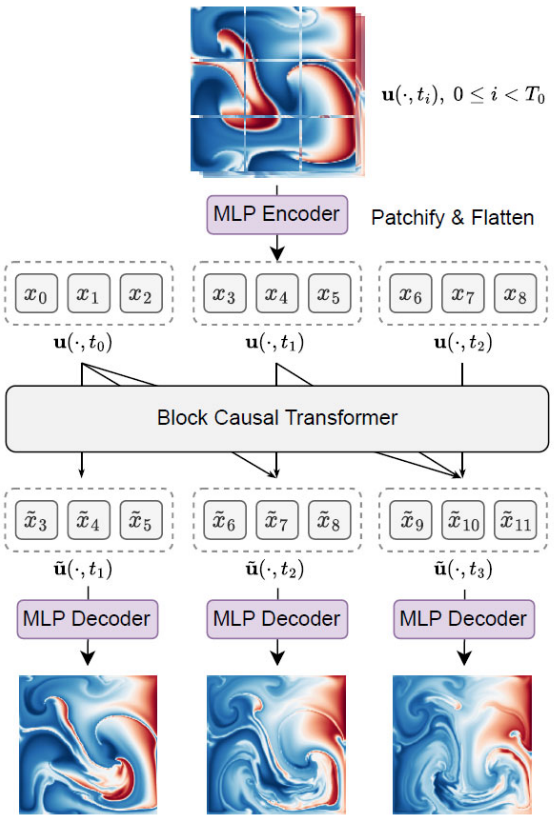


Table 1: **Main Results and Comparisons with Baselines**. The numbers reported are relative L^2 errors (%). The averages are taken with respect to the 6 distinct families listed in the columns of the table. We **bold** the best result in each column.

Model	Param	PDEBench			PDEArena		CFDBench	Average
		SWE	CNS*	INS	NS	NS-cond	-	
DeepONet	3.5M	3.55	7.41	64.61	35.33	51.85	12.50	29.21
FNO	0.6M	3.71	6.31	36.84	38.67	55.63	8.52	24.95
UNet	5.6M	0.33	3.19	3.43	12.56	16.82	0.76	6.18
MPP-B	116M	1.02	1.90	7.52	5.71	12.57	1.23	4.99
ViT	162M	0.25	1.49	2.82	7.05	12.41	0.55	4.10
MPP-L	407M	0.47	1.53	6.42	4.64	9.64	0.73	3.91
DPOT-M	122M	0.54	1.01	5.20	4.92	8.55	0.64	3.47
PROSE-FD	165M	0.28	1.41	2.75	5.27	9.61	0.61	3.32
DPOT-L	523M	0.15	0.89	4.08	2.21	5.29	**0.34**	2.16
BCAT	156M	**0.10**	**0.39**	**1.34**	**1.59**	**3.13**	0.52	**1.18**

NN surrogates – discrete space models –Transformers Transolver (Wu et al. 2024)

- Regular patches do not capture the underlying physics
- Objective: learn « physical » tokens and decrease attention complexity
  - Physical tokens
    - Decompose automatically the mesh into domains where points share similar physical states
    - Encode each slice (domain) into a « physical » token

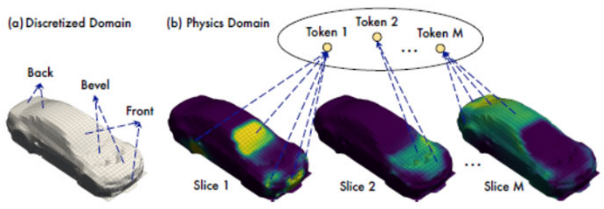


Figure 2. Learning physics-aware tokens from Transolver slices.

Slices for car surface pressure, 3D mesh

  - Decrease attention complexity
    - Apply attention on these physical tokens instead of regular patches or mesh points
- Operates on point clouds, meshes, regular grids

NN surrogates – discrete space models –Transformers
Transolver (Wu et al. 2024)

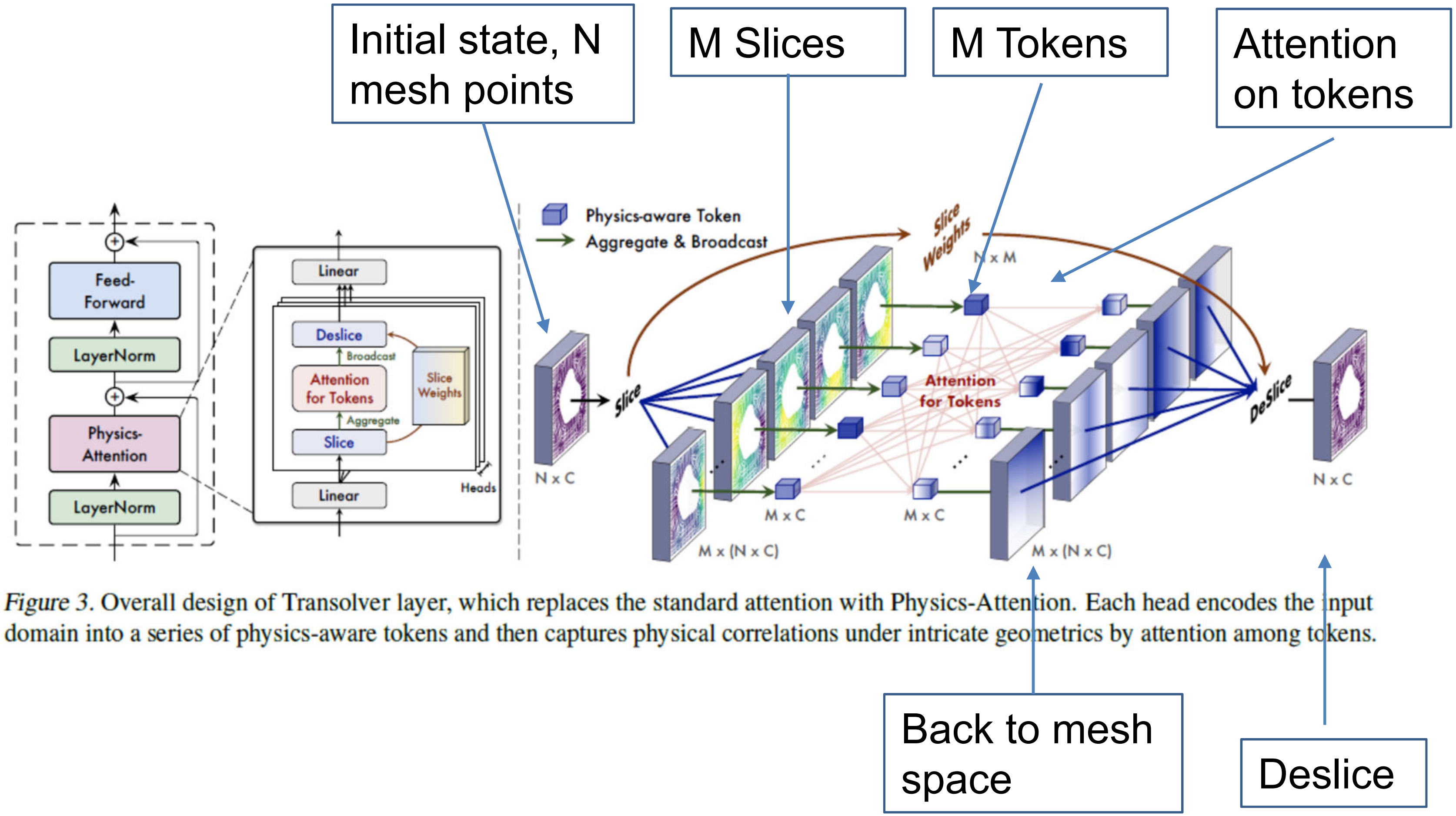


Figure 3. Overall design of Transolver layer, which replaces the standard attention with Physics-Attention. Each head encodes the input domain into a series of physics-aware tokens and then captures physical correlations under intricate geometrics by attention among tokens.

Modeling Spatio-temporal dynamics with Neural Networks

NNs as surrogate models for solving PDEs – Discrete space models

✓ NNs as surrogate models for solving PDEs – Continuous space models

NN surrogates – Operators

- Instead of learning maps between vector spaces (functions), learn maps between function spaces (operators)
  - Images for example are considered as continuous functions
  - The objective is then to learn the operator mapping an input image to an output one
- Objectives
  - Handle irregular and diverse geometries (inputs): meshes, point sets, grids
  - Query at any space-time coordinate in the output space
- Examples: 3 families of methods
  - Frequential representations - Neural Fourier operators (2020)
  - Implicit Neural Representation - CORAL (2023)
  - Attention mechanisms + Transformers - AROMA (2024)

NNs as surrogate models for solving PDEs – Continuous space models

- ✓ Fourier Neural Operators
- CORAL: COordinate-based model for opeRAtor Learning
- AROMA: Attentive Reduced Order Model with Attention

NNs as surrogate models for solving PDEs – Operators
Fourier Neural Operator (Li et al. 2021)

- We consider
  - $\mathcal{V} = \mathcal{V}(\Omega \subset R^d; R^n), \mathcal{U} = U(\Omega' \subset R^{d'}; \mathrm{R}^{\mathrm{m}})$ two function spaces
  - $\mathcal{G}: \mathcal{V} \to \mathcal{U}$ a non linear unknown mapping between the two function spaces
    - FNO considers mappings $\mathcal{G}$ that correspond to the solution operator of a parametric PDE
- Objective
  - Learn $\mathcal{G}_\theta$ an approximation of $\mathcal{G}$ from a finite set of samples
  - Samples are provided as p-points discretization of functions $v \in \mathcal{V}$ and $\mathrm{u} \in \mathcal{U}$
    - i.e. in practice we learn from discrete spaces, the representation of the continuous functions $v \in \mathcal{V}$ and $u \in \mathcal{U}$

NNs as surrogate models for solving PDEs – Operators
Fourier Neural Operator (Li et al. 2021)

- Classical neural network

$$\boxed{u = (K_T \circ \sigma_T \circ \cdots \circ \sigma_t \circ K_t \circ \cdots \circ \sigma_1 \circ K_0) v}$$

  - With K_t a linear operator, σ_t a non linearity, u, v vectors
- Neural operators (simplified)
  - Follow a similar framework but u and v are no more vectors but functions

$$\boxed{v_{t+1}(x) = \sigma_{t+1}\big(K_t(v_t)(x)\big)}$$

  - With $K_t(v_t)$ an integral operator

$$\boxed{K_t(v_t)(x) = \int_\Omega \varkappa_t(x, y) v_t(y) dy}$$

  - $\varkappa_t(x, y)$is a kernel function
  - $v_t : \Omega \to R^n, v_{t+1} : \Omega \to R^m, \Omega \subset R^d$ a bounded space

NNs as surrogate models for solving PDEs – Operators
Fourier Neural Operator (Li et al. 2021)

- How to learn the kernel function $\varkappa_t$?
- Let us consider the simplified update rule

$$\boxed{u(x) = K(v)(x) = \int_{\Omega} \varkappa(x,y)v(y)dy}$$

  - with $v, u: \Omega \to R^n$
- FNO works in Fourier space
  - $\varkappa(x,y) = \varkappa(x-y)$ is a convolution operator

$$\boxed{u(x) = (\varkappa * v')(x)}$$

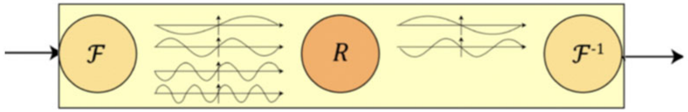
  - Convolution theorem:

$$\boxed{u(x) = \mathcal{F}^{-1}(\mathcal{F}(\varkappa).\mathcal{F}(v'))(x)}$$

  - Convolution in space is equivalent to pointwise multiplication in Fourier domain
  - $\mathcal{F}(\varkappa)$is a linear transformation

NNs as surrogate models for solving PDEs – Operators
Fourier Neural Operator (Li et al. 2021)

- Fourier transform – Linear Transform – Inverse Fourier



$$u(x) = \mathcal{F}^{-1}(\mathcal{F}(\varkappa).\mathcal{F}(v'))(x)$$

R is a linear operator – implemented as a tensor
$\mathcal{F}$ is implemented via a Fast Fourier Transform (complexity $nlogn$, n nb of spatial points)

- Operates on regular grids
- FFT is independent of the grid size
  - Could be used on resolutions different from the training ones

FNO model

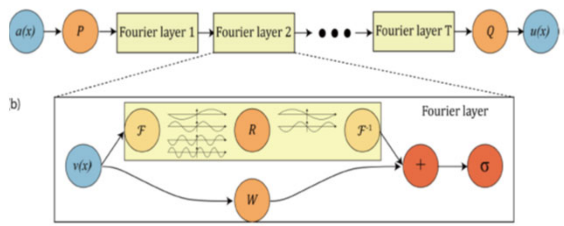


Fig Li et al. 2021

NNs as surrogate models for solving PDEs – Continuous space models Fourier Neural Operator (Li et al. 2021)

- Example: zero shot super-resolution
  - 2 D Navier Stokes, vorticity form, viscuous incompressible fluid
    - $\frac{\partial}{\partial t} w(x,t) + u(x,t).\nabla w(x,t) = \nu \Delta w(x,t) + f(x), x \in (0,1)^2, t \in (O,T]$
    - $\nabla . u(x,t) = 0, x \in (0,1)^2, t \in (0,T)$
    - $u(x,t)$ velocity field, $w(x,t)$ vorticity, characterizes local rotation of the fluid
  - Fig. Illustrates super-resolution: trained at 64x64, test on 256x256

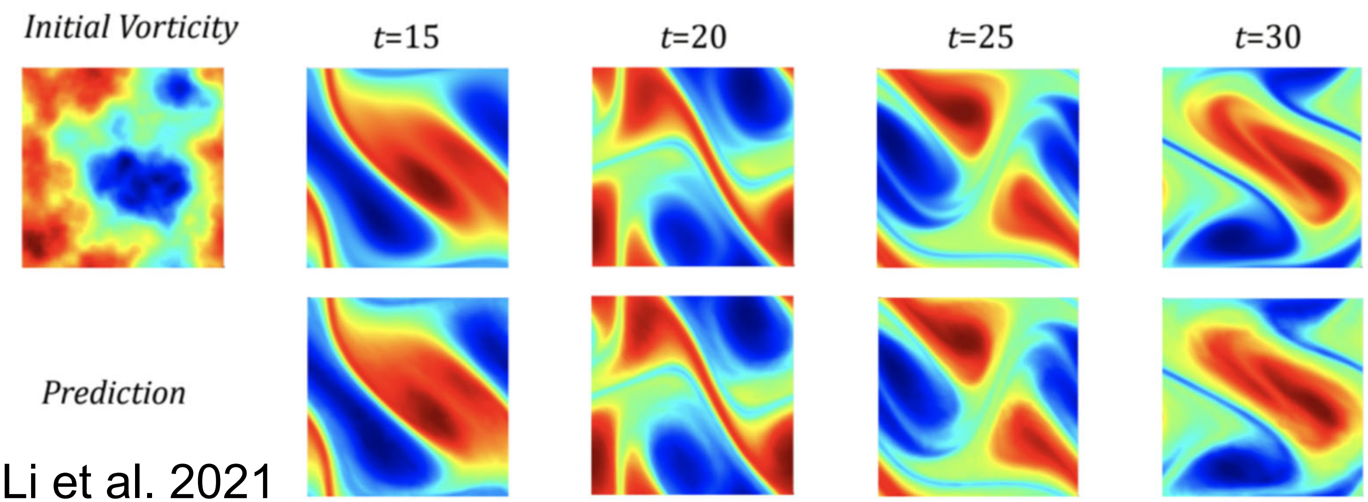


Fig Li et al. 2021

Zero-shot super-resolution: Navier-Stokes Equation with viscosity $\nu = 1e{-4}$; Ground truth on top and prediction on bottom; trained on $64 \times 64 \times 20$ dataset; evaluated on $256 \times 256 \times 80$ (see Section 5.4).

NNs as surrogate models for solving PDEs – Continuous space models

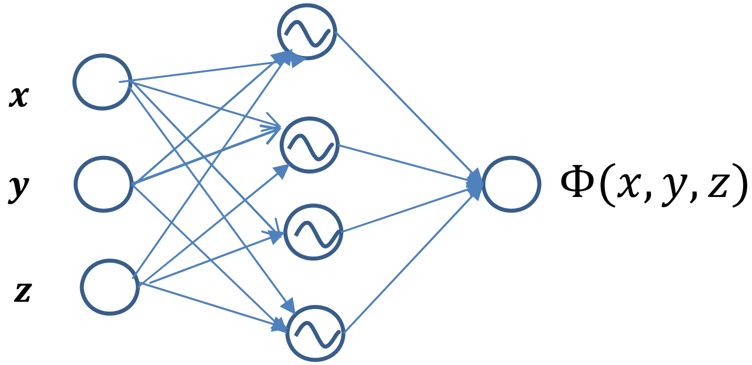
Fourier Neural Operators

✓ CORAL: COordinate-based model for opeRAtor Learning

AROMA: Attentive Reduced Order Model with Attention

NNs as surrogate models for solving PDEs – Operators
Neural Fields (Implicit Neural Representations)

- Coordinate-based approximation of functions
  - Continuous representations of objects as coordinate-dependent functions
  - Appeared initially as a novel way to represent 3D shapes in place of discrete representations
    - Example: signed distance



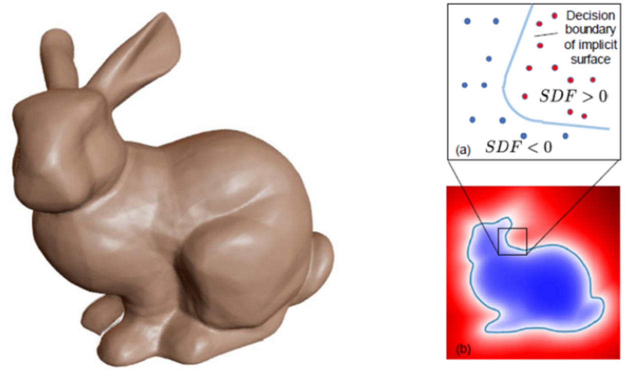


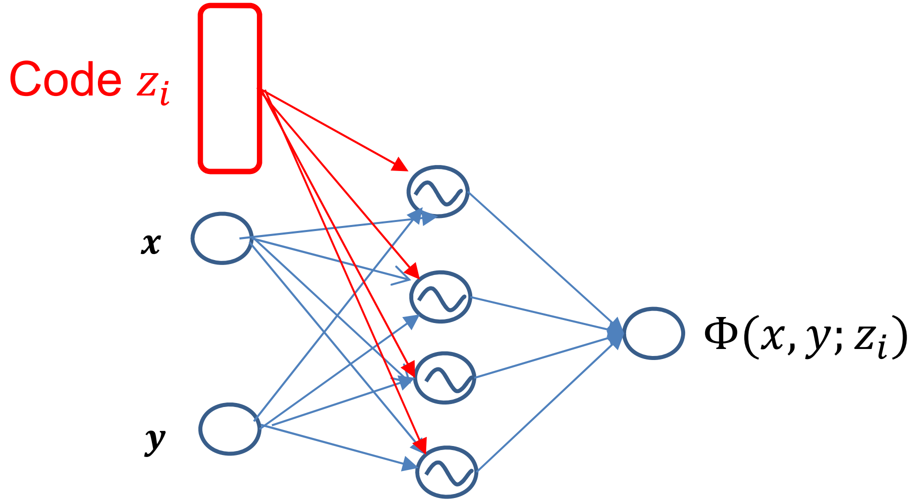
Fig. Park et al. 2019

    - The shape is fully described by the NN parameters
    - Mesh-free approach – independent of the resolution: learn from point sets

- References: Sitzmann et al. 2020, Fathony et al., 2021, Tancik et al. 2020, etc

NNs as surrogate models for solving PDEs – Operators
Neural Fields (Implicit Neural Representations)

- Learning several images
  - A neural field model represents one image
  - How to represent multiple images using a single model?
    - Condition the neural field on a compact code specific of an image



    - This code z_i could be learned e.g. through auto decoding by gradient descent and is specific to an image
    - Conditioning is performed through e.g. a hypernetwork that adapts some networks parameters to each image
    - Network weights (in blue) are shared across images

CORAL : Operator Learning with Neural Fields (Serrano et al. 2023)

- Tasks: learn mappings between input – output functions

Direct forecasting (IVP) $u_0 \to u_T$
Dynamics modeling $u_t \to u_{t+1}$
Geometry aware inference $a_{geo} \to u_{sol}$

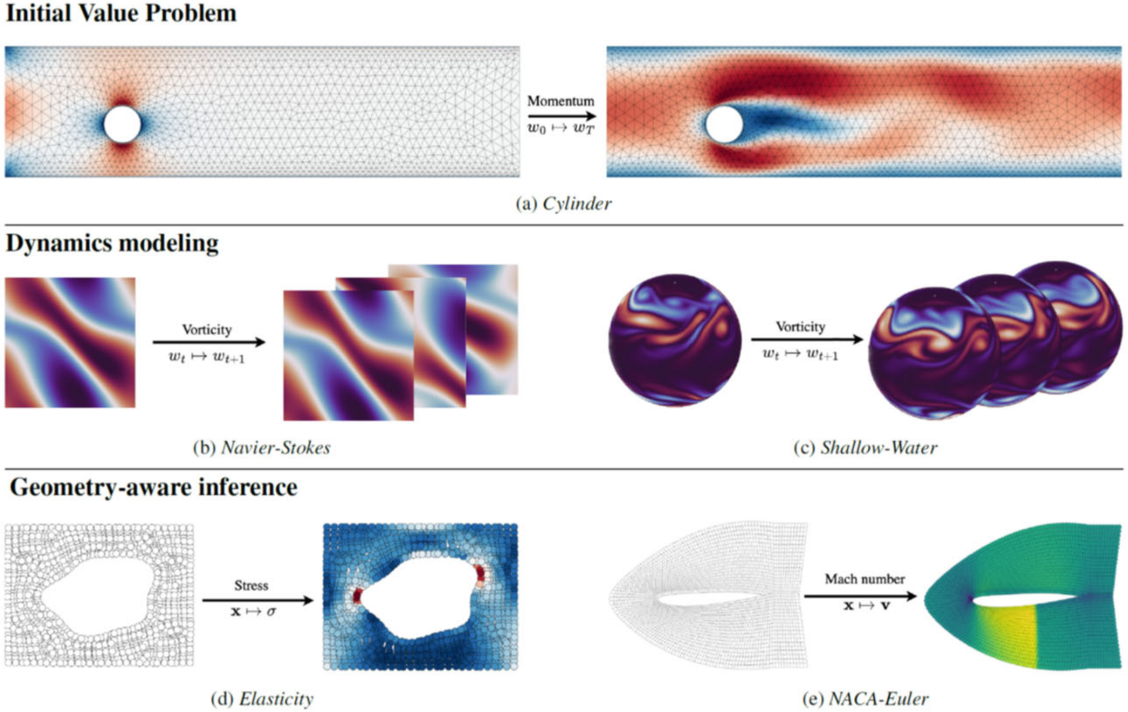
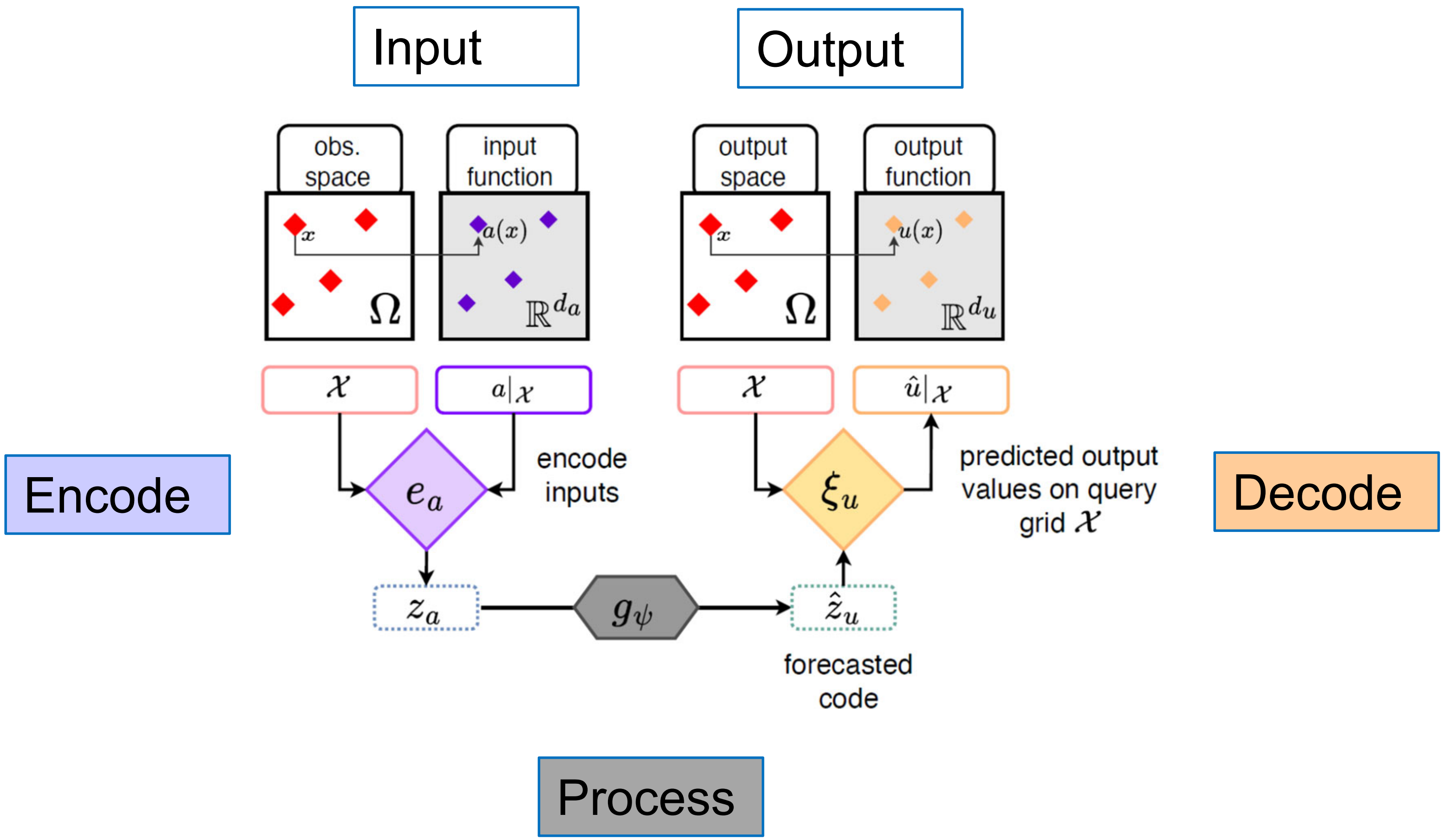


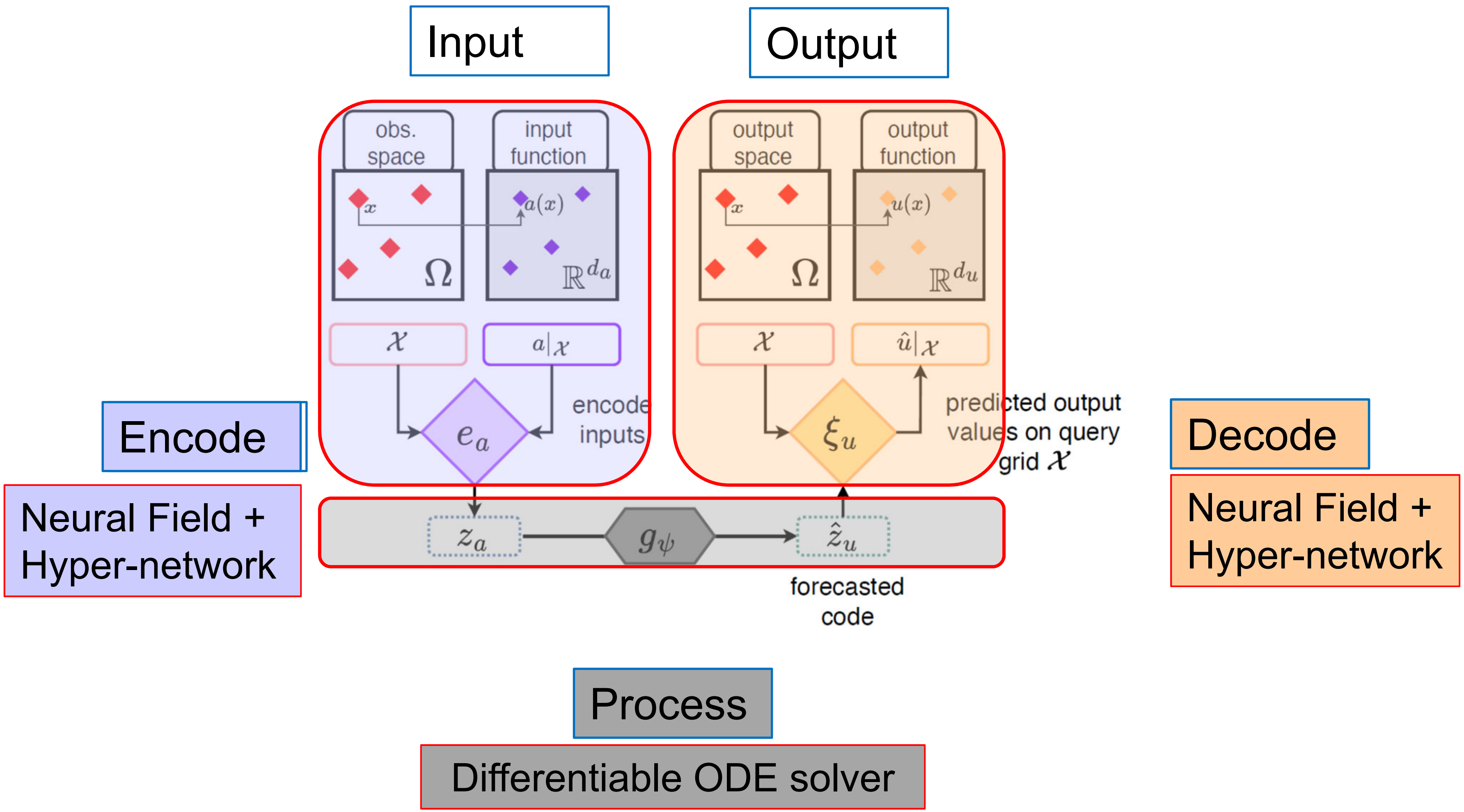
Figure 1: Illustration of the problem classes addressed in this work: Initial Value Problem (IVP) (a), dynamic forecasting (b and c) and geometry-aware inference (d and e).

NNs as surrogate models for solving PDEs – Operators
CORAL : Operator Learning with Neural Fields - (Serrano et al. 2023)
Inference

Encode-Process-Decode framework



NNs as surrogate models for solving PDEs – Operators
CORAL : Operator Learning with Neural Fields - (Serrano et al. 2023)
Inference



NNs as surrogate models for solving PDEs – Operators
CORAL : Operator Learning with Neural Fields (Serrano et al. 2023) - Inference

Geometry aware inference: NACA-Euler (Mach number)

Forecasting on Shallow-Water (vorticity)
Robustness to changes of grid and time extrapolation

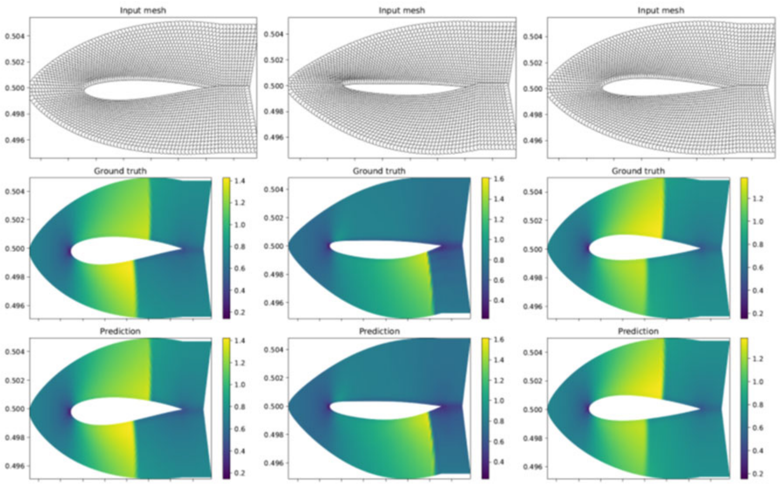


Figure 14: CORAL predictions on *NACA-Euler*

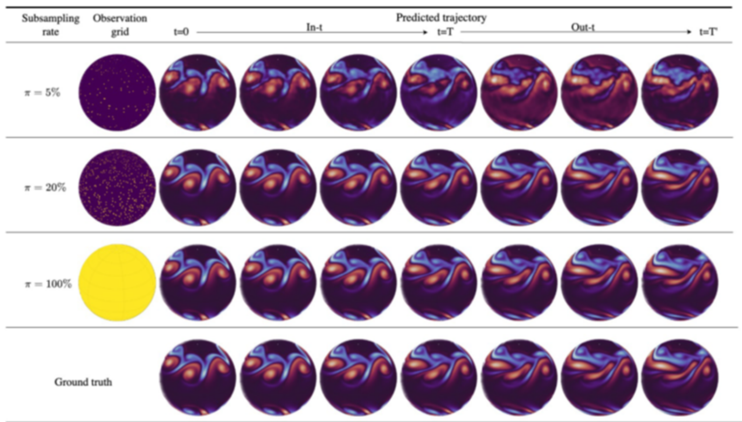


Figure 13: Prediction MSE per frame for CORAL on *Shallow-Water* with its corresponding training grid $\mathcal{X}$. Each row corresponds to a different sampling rate and the last row is the ground truth. The predicted trajectory is predicted from $t = 0$ to $t = T'$. In our setting, $T = 19$ and $T' = 39$.

NNs as surrogate models for solving PDEs – Continuous space models

Fourier Neural Operators

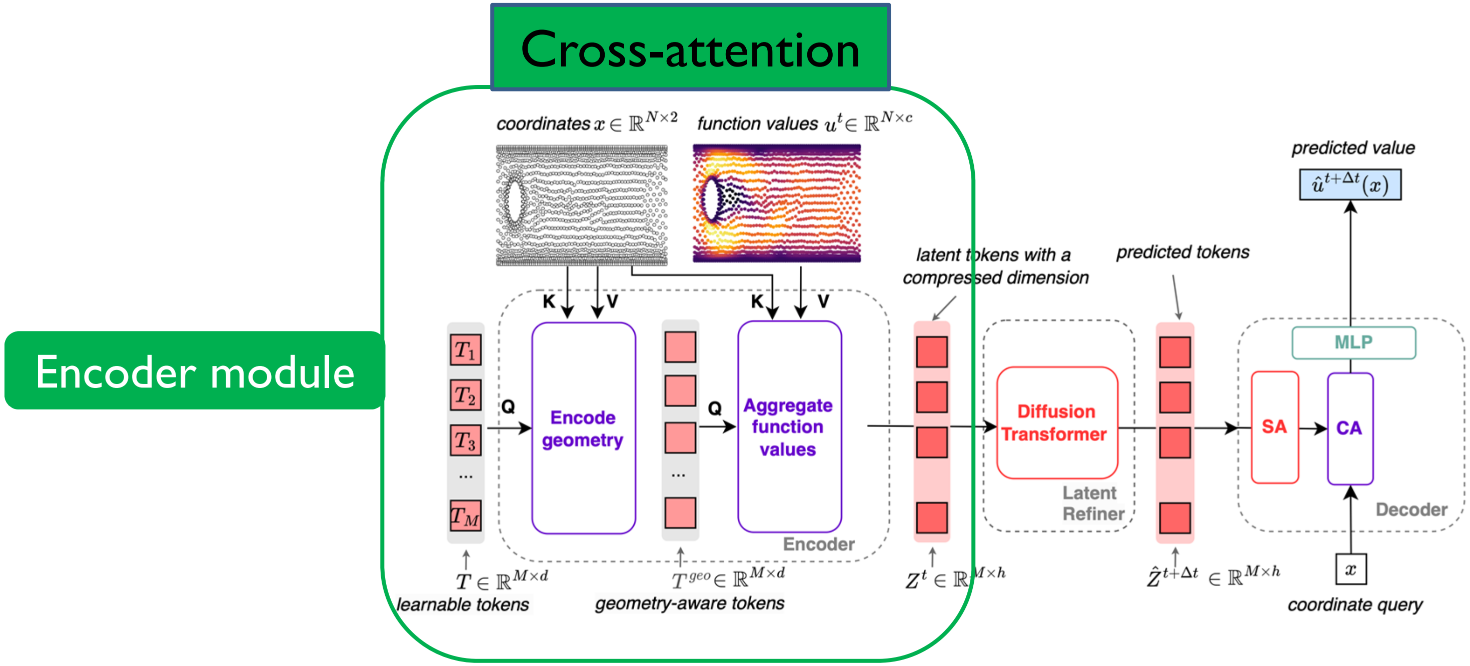
CORAL:COordinate-based model for opeRAtor Learning

✓ AROMA: Attentive Reduced Order Model with Attention

NNs as surrogate models for solving PDEs – Operators
AROMA: Attentive Reduced Order Model with Attention (Serrano et al. 2024)

- Principled Framework:
  - Properties
    - Handle diverse geometries: inputs and outputs may consist in point sets, grids, irregular meshes
    - Captures local spatial information
    - Can be queried at any spatial position
  - Demonstrates how modern NN components allow building versatile PDE solvers
    - Encode/ Process/ Decode framework
      - Encoding: cross-attention maps variable-size inputs to a fixed-size compact latent token space encoding local spatial information
      - Processing: a diffusion transformer architecture to model dynamics and exploit spatial relations locally and globally via self-attention + model uncertainty
      - Decoding: uses a conditional neural field + cross-attention to query forecast values at any spatial point within the equation's domain

NNs as surrogate models for solving PDEs – Operators
AROMA: Attentive Reduced Order Model with Attention (Serrano et al. 2024) - General framework



Cross-attention encoder: $u^t \to Z^t$

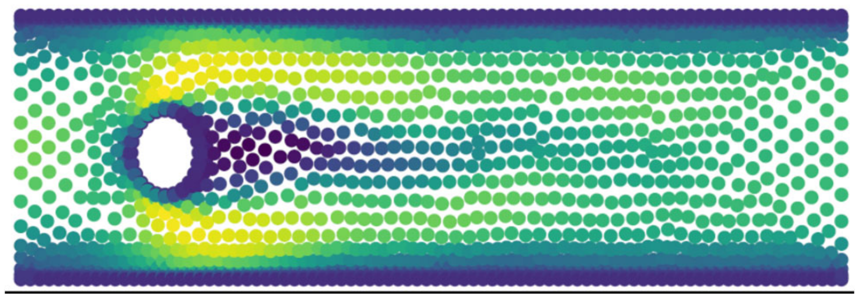
- Encodes variable size discretized input $u()$ into a fixed size & small dimensional sequence of latent embedding tokens Z
- Z encodes local spatial information on problem geometry + variable values

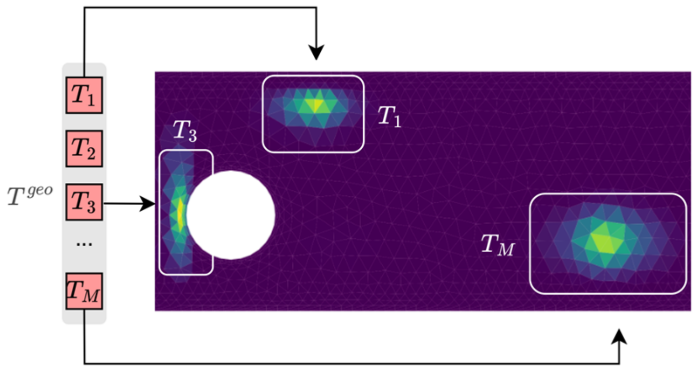
NNs as surrogate models for solving PDEs – Operators
AROMA: Attentive Reduced Order Model with Attention (Serrano et al. 2024) - Cross-attention encoder captures spatial attention

Example: Cross attention on cylinder flow

- Cylinder flow ground truth

- Tokens encode local spatial information – cross attention between T^{geo} tokens and "x"

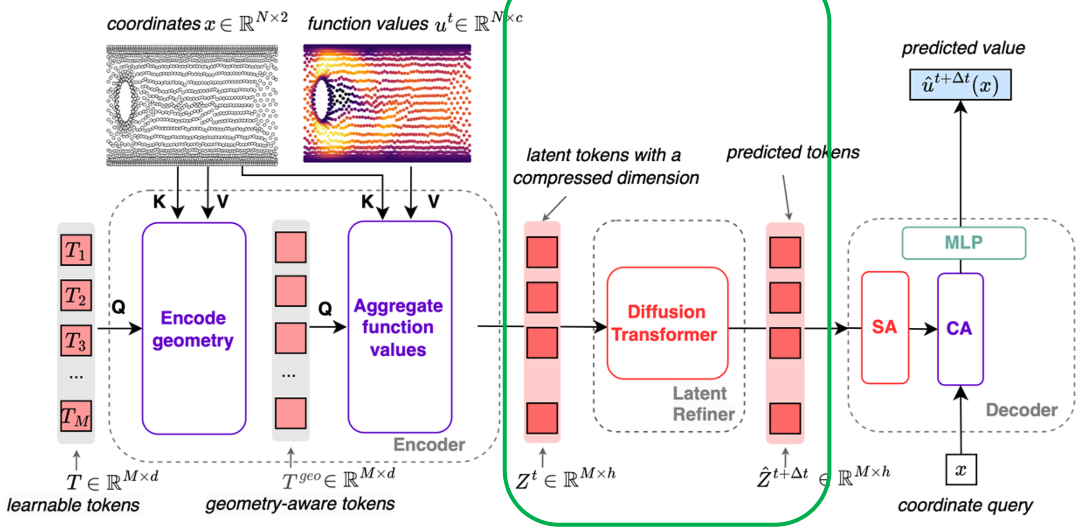




NNs as surrogate models for solving PDEs – Operators
AROMA: Attentive Reduced Order Model with Attention Attention (Serrano et al. 2024) -General framework

Self-attention

Processor



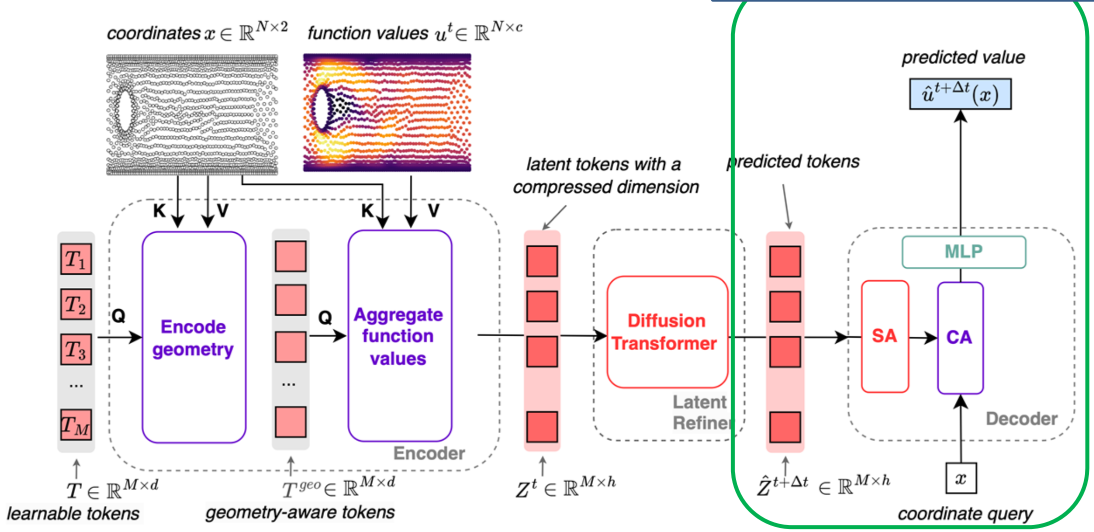
Time stepping transformer: $Z^t \rightarrow Z^{t+\Delta t}$

- Learns the dynamics in the small dimensional latent space
- Self attention models relations between spatial latent tokens
- Inference: dynamics is enrolled in the latent space starting from an initial condition– low complexity
- Diffusion: introduces a stochastic component

NNs as surrogate models for solving PDEs – Operators
AROMA: Attentive Reduced Order Model with Attention Attention (Serrano et al. 2024)-General framework

Cross-attention

Decoder module



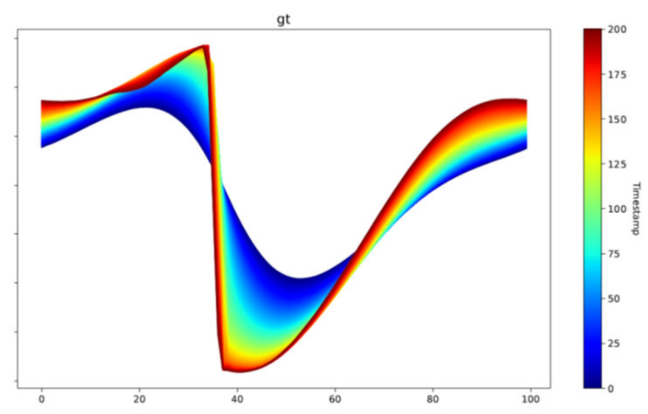
Cross-attention neural fields decoder: $Z^{t+\Delta t} \rightarrow u^{t+\Delta t}$

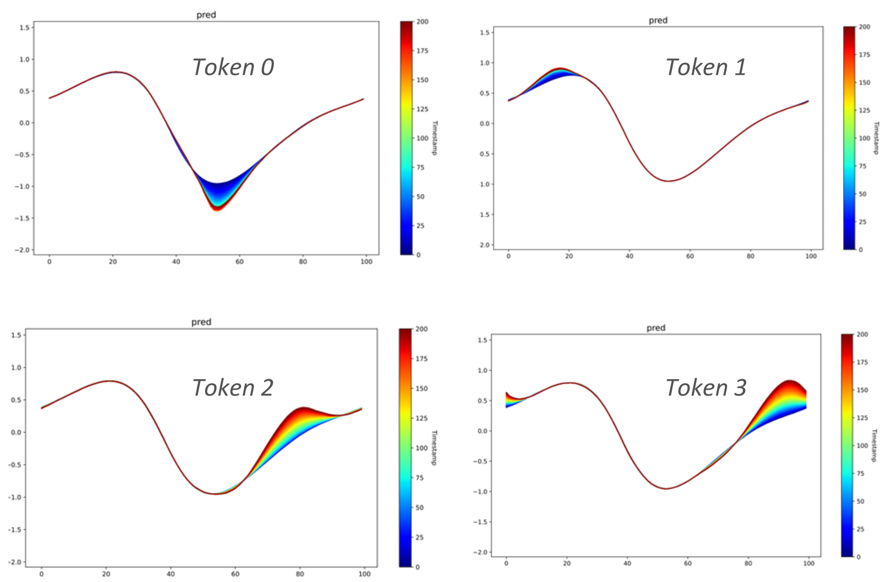
- Maps the latent representation $Z^{t+\Delta t}$ to the original physical space
- Can be queried at any position x of the physical space

NNs as surrogate models for solving PDEs – Operators AROMA: Attentive Reduced Order Model with Attention - (Serrano et al. 2024)Cross-attention encoder captures spatial attention

Example: Burgers equation – perturbation analysis on the tokens

- Burgers equation ground truth
- Tokens encode local spatial information





Conclusion

- Survey of main current data-driven methods for modeling physical dynamics
  - Benefits from advances in different ML fields (Vision, NLP, …)
  - Large size applications deployed in some domains e.g. weather forecast
  - Gap to real world applications in many domains e.g. CFD

- Takeaways
  - Scaling is a central problem
  - Latent models are probably the correct way to proceed
  - Importance of efficient and reliable « physical » encodings / decoding operating on multiple geometries
  - Design of scalable neural operators

- Thanks for your attention

References used in the presentation

- A blog on Neural ODE: https://www.depthfirstlearning.com/2019/NeuralODEs
- Ayed I., de Bezenac E., Pajot A., Brajard J. , Gallinari P. , (2019), Learning Dynamical Systems from Partial Observations, arXiv:1902.11136
- Ayed, I., de Bézenac, E., Pajot, A., & Gallinari, P. (2022). Modelling spatiotemporal dynamics from Earth observation data with neural differential equations. *Machine Learning*, *111*(6), 2349–2380.
- Belbute-Peres, F. de A., Economon, T. D., & Kolter, J. Z. (2020). Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. *ICML*.
- Brandstetter, J., Worrall, D. E., & Welling, M. (2022). Message Passing Neural PDE Solvers. *ICLR*.
- Chen, R.T.Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. 2018. Neural Ordinary Differential Equations. *NIPS*.
- de Bezenac, E., Pajot, A., & Gallinari, P. (2018). Deep Learning For Physical Processes: Incorporating Prior Scientific Knowledge. In *ICLR, also in Journal of Statistical Mechanics: Theory and Experiment, 2019*.
- Dona, J., Dechelle, M., Gallinari, P., & Levy, M. (2022). Constrained Physical-Statistical Models for Dynamical System Identification and Prediction. *ICLR*.
- E, W. (2017). A Proposal on Machine Learning via Dynamical Systems. *Communications in Mathematics and Statistics*, *5*(1). https://doi.org/10.1007/s40304-017-0103-z
- Fathony, R., Sahu, A. K., Willmott, D., & Kolter, J. Z. (2021). Multiplicative Filter Networks. *ICLR*, 1–10.
- Haber, E. and Ruthotto, L. 2018. Stable architectures for deep neural networks. *Inverse Problems*. 34, 1 (2018).

References used in the presentation + additional closely related references

- Kirchmeyer, M., Yin, Y., Dona, J., Baskiotis, N., Rakotomamonjy, A. and Gallinari, P. 2022. Generalizing to New Physical Systems via Context-Informed Dynamics Model. *arXiv:2202.01889v1* (2021).
- Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning accelerated computational fluid dynamics. Proceedings of the National Academy of Sciences, 118(21).
- Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2022). Neural Operator: Learning Maps Between Function Spaces. *JMLR*. http://arxiv.org/abs/2108.08481
- Liu, Y., Sun, J., & Schaeffer, H. (2025). BCAT: A Block Causal Transformer for PDE Foundation Models for Fluid Dynamics. http://arxiv.org/abs/2501.18972
- Park, J.J, Florence, P, , Straub, J, Newcombe, R, and Lovegrove,, S, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),
- Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., & Anandkumar, A. (2022). *FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators*. http://arxiv.org/abs/2202.11214

References used in the presentation + additional closely related references

- Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., & Prabhat. (2019). Deep learning and process understanding for data-driven Earth system science. *Nature*, *566*, 195–204. https://doi.org/10.1038/s41586-019-0912-1
- Serrano, L., Boudec, L. le, Koupaï, A. K., Wang, T. X., Yin, Y., Vittaut, J.-N., & Gallinari, P. (2023). Operator Learning with Neural Fields: Tackling PDEs on General Geometries. NeurIPS. http://arxiv.org/abs/2306.07266
- Serrano, L., Wang, T., le Naour, E., Vittaut, J.-N., & Gallinari, P. (2024). AROMA : Preserving Spatial Structure for Latent PDE Modeling with Local Neural Fields. NeurIPS.
- Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., Wetzstein, G., & University, S. (2020). Implicit Neural Representations with Periodic Activation Functions. *Neurips*.
- 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. *Neurips*.
- Wu, H., Luo, H., Wang, H., Wang, J., & Long, M. (2024). Transolver: A Fast Transformer Solver for PDEs on General Geometries. http://arxiv.org/abs/2402.02366
- Wu, J., Chan, C. K., Chen, S., Zhou, L., Yu, N., Chen, E., … Liu, T. Y. (2024). TamGen: drug design with target-aware molecule generation through a chemical language model. Nature Communications, 15(1), 9360. https://doi.org/10.1038/s41467-024-53632-4
- Yin, Y., Ayed, I., de Bézenac, E., Baskiotis, N., & Gallinari, P. (2021). LEADS: Learning Dynamical Systems that Generalize Across Environments. *Neurips*.
- Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., & Gallinari, P. (2023). Continuous PDE Dynamics Forecasting with Implicit Neural Representations. *ICLR*, 1–19. http://arxiv.org/abs/2209.14855
- Yin, Y., Le Guen, V., Dona, J., de Bezenac, E., Ayed, I., Thome, N., & Gallinari, P. (2021). Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting. *ICLR*.