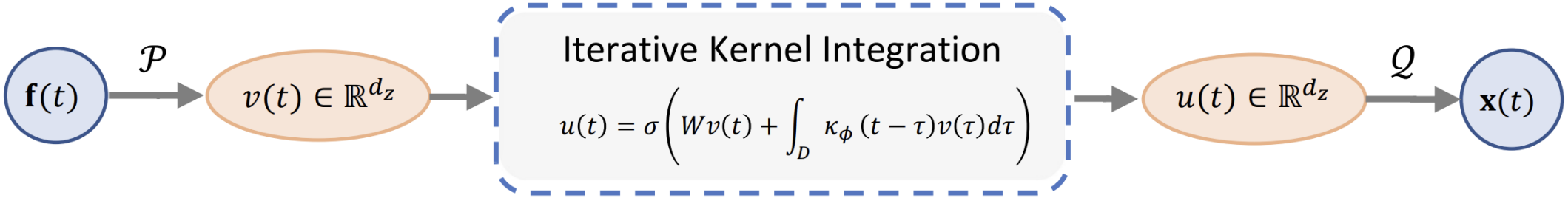
Basis-specific Neural Operators



- Graph Neural Operator [1]
- Low-rank Neural Operator
- Multipole Graph Neural Operator
- Fourier Neural Operator [2]
- Wavelet Neural Operator [3]

[1] Kovachki, N. et al. (2021). Neural operator: Learning maps between function spaces.

[2] Li, Z. et al. (2020). Fourier neural operator for parametric partial differential equations.

[3] Tripura, T. and Chakraborty, S. (2022). Wavelet neural operator: a neural operator for parametric pdes.

Integral Kernel – Green's Functions

Considering the generic family of PDEs of the following form:

$$(\mathcal{L}_a u)(x) = f(x), \qquad x \in D$$
$$u(x) = 0, \qquad x \in \partial D \tag{1}$$

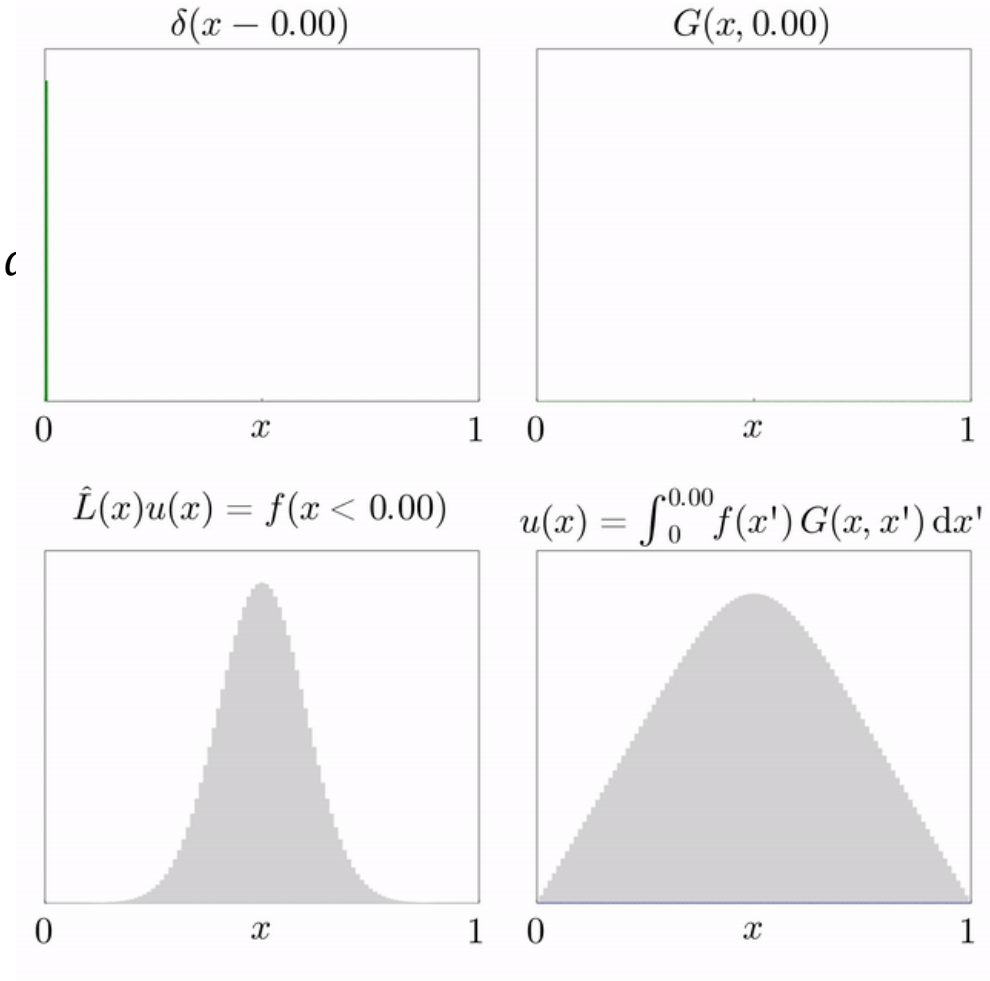
where $\mathcal{L}_a$ is a differential operator depending on a parameter a

When the input is the delta measure centered at x, namely, $f(x) = \delta_x$, the unique solution of Eq. (1) is defined as the Green's function G_a, that is,

$$\mathcal{L}_a G(x, \cdot) = \delta_x \tag{2}$$

Since the source term $f(x)$ in Eq. (1) is a sum of delta functions, the solution of Eq. (1) can be represented as Eq. (3) by applying the superposition principle:

$$u(x) = \int_D G_a(x, y) f(y)\, dy \tag{3}$$



Building the solution for a general source based on Green's function [1]

[1] https://en.wikipedia.org/wiki/Green%27s_function

Neural Operators

$$u(x) = \int_D G_a(x, y) f(y) dy \tag{3}$$

Guided by Eq. (3), the following iterative architecture is proposed [2]:

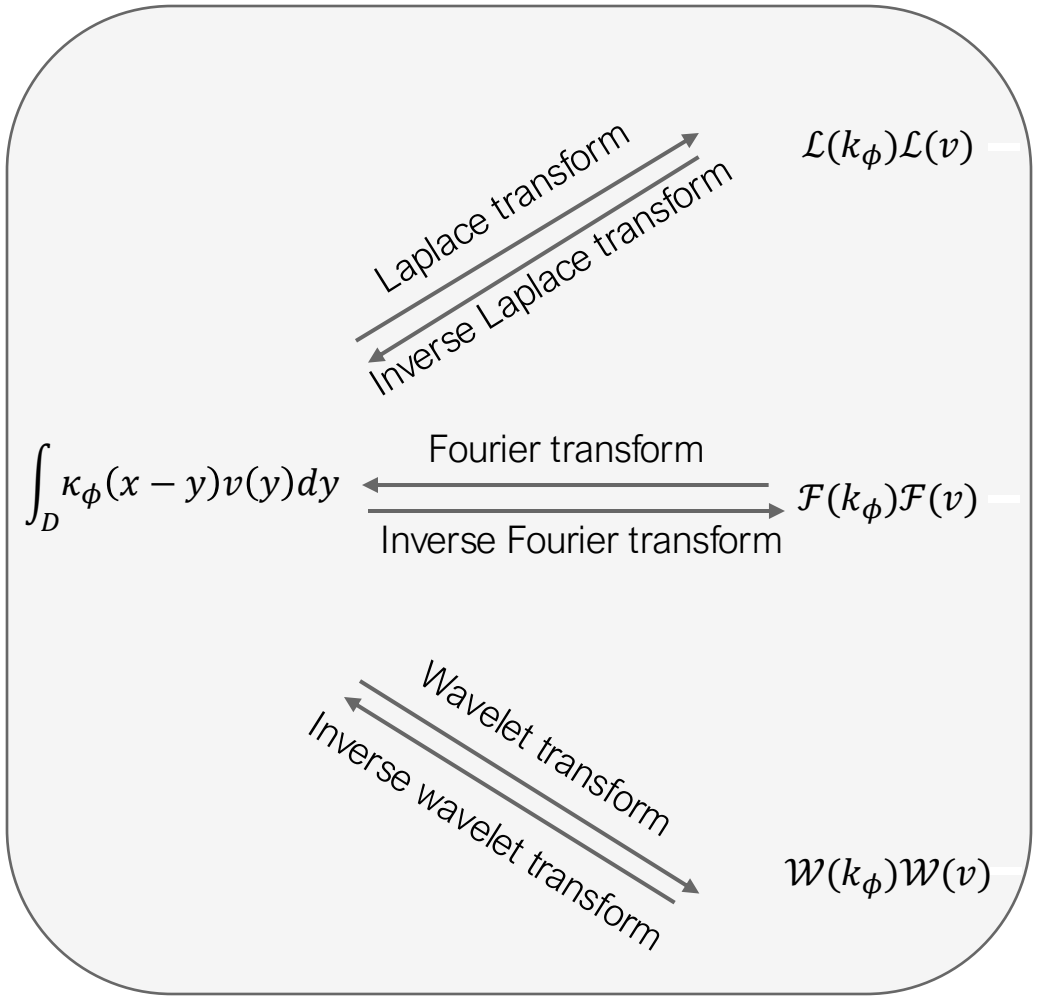
$$v_{t+1}(x) = \sigma\left(W v_t(x) + \int_D \kappa_\phi(x - y) v_t\,(y) dy\right) \tag{4}$$

where σ is the activation function, the matrix W and the kernel κ_ϕ are modeled as neural networks, which are learned from data.

- Monte Carlo approximation ⟹ Expensive cost
- Convolution theorem ⟹ Multiplication in the other domain

[2] Li, Z. et al. (2020). Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*.

Convolution Theorem



- Laplace Neural Operator [3]

$$v_{t+1}(x) = \sigma\big(Wv_t(x) + \mathcal{L}^{-1}(\mathcal{L}(k_\phi)\mathcal{L}(v))\big)$$

- Fourier Neural Operator [4]

$$v_{t+1}(x) = \sigma\big(Wv_t(x) + \mathcal{F}^{-1}(\mathcal{F}(k_\phi)\mathcal{F}(v))\big)$$

- Wavelet Neural Operator [5]

$$v_{t+1}(x) = \sigma\big(Wv_t(x) + \mathcal{W}^{-1}(\mathcal{W}(k_\phi)\mathcal{W}(v))\big)$$

[3] Cao, Q., Goswami, S., & Karniadakis, G. E. (2023). LNO: Laplace neural operator for solving differential equations.

[4] Li, Z. et al. (2020). Fourier neural operator for parametric partial differential equations.

[5] Tripura, T. and Chakraborty, S. (2022). Wavelet neural operator: a neural operator for parametric partial differential equations.

Fourier Transform

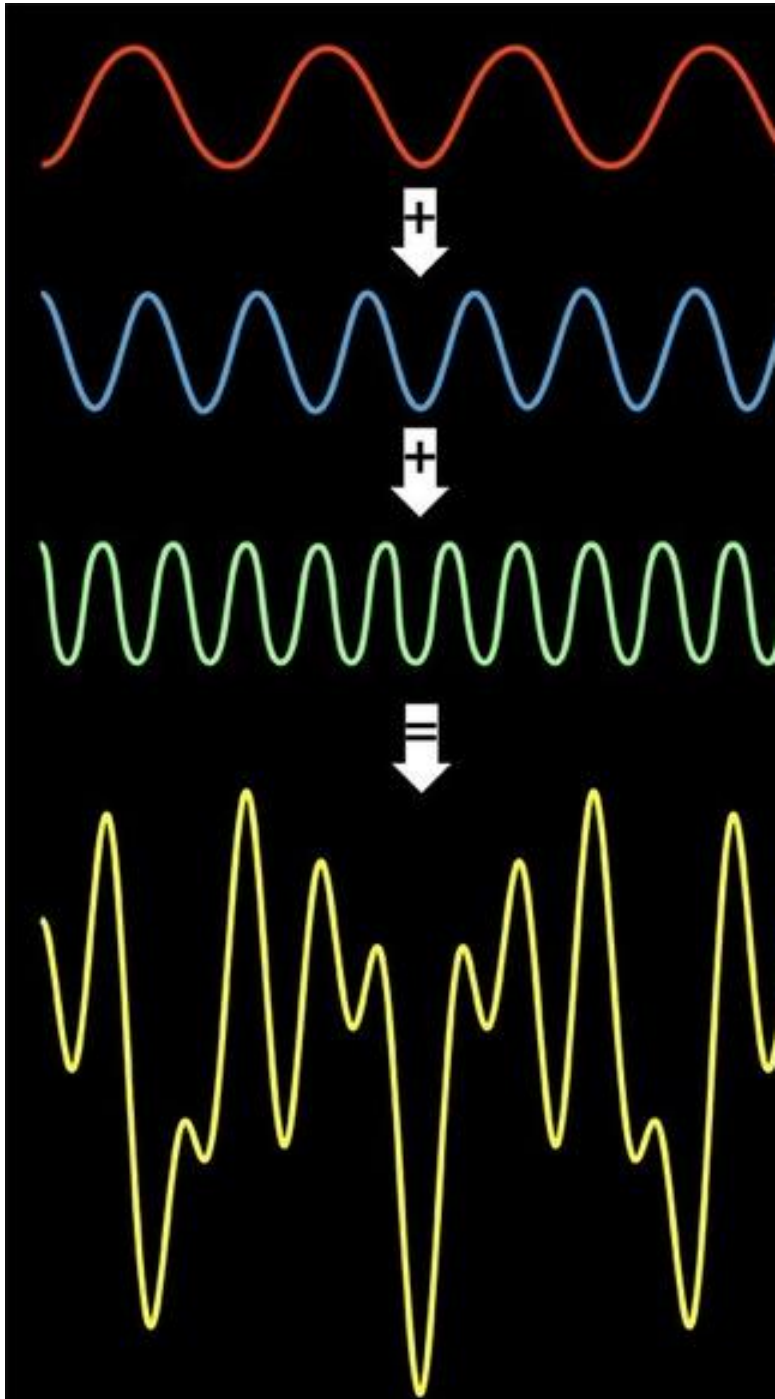
$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{\infty} A_k \cos(kx) + B_k \sin(kx),$$

The Fourier transform pair for a function $f(x)$ is defined as

$$\widehat{f}(k) = \mathcal{F}[f(x)] = \int_{-\infty}^{\infty} f(x) e^{-ikx} dx,$$

$$f(x) = \mathcal{F}^{-1}[\widehat{f}(k)] = \frac{1}{\pi}, \int_{-\infty}^{\infty} \widehat{f}(k) e^{ikx} dk$$

where k could be frequency or wavenumber in time or space domain.



Fourier Transform

Fourier transform of a partial derivative is defined as

$$\mathcal{F}\left(\frac{\partial f}{\partial x}\right) = \int_{-\infty}^{\infty} \frac{\partial f}{\partial x} e^{-ikx} dx$$

Performing integration by parts, yields

$$\mathcal{F}\left(\frac{\partial f}{\partial x}\right) = \left[f(x)e^{-ikx}\right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(x)(-ike^{-ikx})dx$$

Fourier transforms are applicable to periodic functions, and thus by cosntruction it requires $f(-\infty) = f(\infty) = 0$.

Therefore,

$$\mathcal{F}\left(\frac{\partial f}{\partial x}\right) = ik \int_{-\infty}^{\infty} f(x)(e^{-ikx})dx = ik\widehat{f}(k)$$

Now, taking the inverse Fourier transform on both sides:

$$\frac{\partial f}{\partial x} = ik\mathcal{F}^{-1}\left(\mathcal{F}(f(x))\right)$$

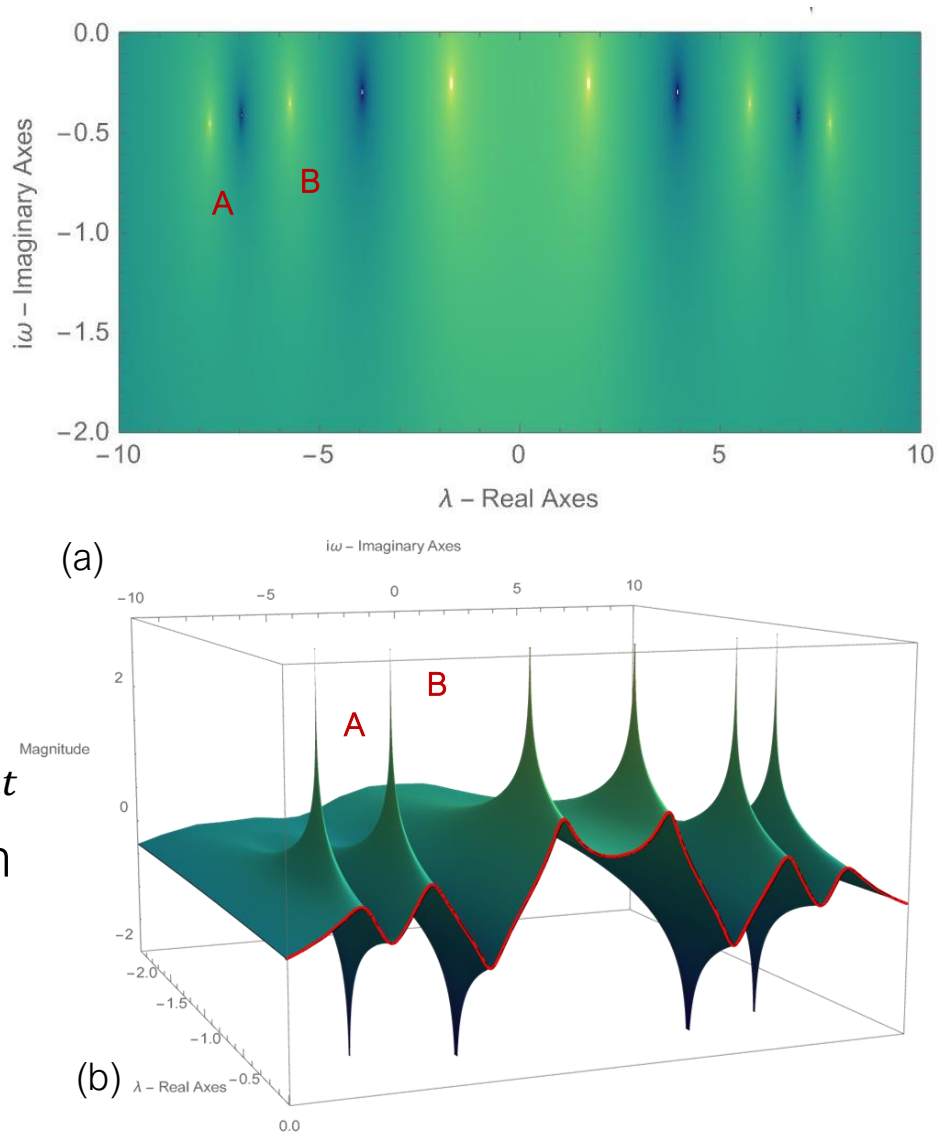
Relation between Laplace transform and Fourier transform

$$\mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-st}dt \qquad (5)$$

$$s = \lambda + i\omega$$

$$\mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-\lambda t}e^{-i\omega t}dt \qquad (6)$$

- Laplace transform of $f(t)$ is the Fourier transform of $f(t)e^{-\lambda t}$
- Letting $\lambda = 0$, Laplace transform becomes Fourier transform



Visualization of transform functions: (a) 2D plot; (b) 3D plot

Relation between LNO and FNO

LNO

$$u_1(x) = \int_0^\infty \kappa_\phi(x-y)v(y)dy \tag{7}$$

↓ Laplace transform

$$U_1(s) = K_\phi(s)V(s) \tag{8}$$

$$U_1(s) = \left(\sum_{n=1}^{N} \frac{\beta_n}{s-\mu_n}\right)\left(\sum_{\ell=-\infty}^{\infty} \frac{\alpha_\ell}{s-i\omega_\ell}\right) \tag{9}$$

↓ Residue theorem

$$U_1(s) = \sum_{n=1}^{N} \frac{\beta_n V(\mu_n)}{s-\mu_n} + \sum_{\ell=-\infty}^{\infty} \frac{\alpha_\ell K_\phi(i\omega_\ell)}{s-i\omega_\ell} \tag{10}$$

↓ Inverse Laplace transform

$$u_1(x) = \sum_{n=1}^{N} \beta_n V(\mu_n) e^{\mu_n x} + \sum_{\ell=-\infty}^{\infty} \alpha_\ell K_\phi(i\omega_\ell) e^{i\omega_\ell x} \tag{11}$$

Transient response

Steady-state response

FNO

$$u_1(x) = \int_0^\infty \kappa_\phi(x-y)v(y)dy \tag{12}$$

↓ Fourier transform

$$U_1(\omega) = K_\phi(\omega)\alpha(\omega) \tag{13}$$

↓ Inverse Discrete Fourier transform

$$u_1(x) = \sum_{\ell=-\infty}^{\infty} U(\omega_\ell)\, e^{i\omega_\ell x} \tag{14}$$

$$u_1(x) = \sum_{\ell=-\infty}^{\infty} \alpha_\ell K_\phi(\omega_\ell)\, e^{i\omega_\ell x} \tag{15}$$

Relation between Wavelet transform and Fourier transform (WNO and FNO)

Wavelet transform

$$\mathcal{W}(a,b) = \int_D v(x) \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) dx$$

when one considers the complex exponential $\psi(x) = e^{-i\omega_0 x}$, then :

$$\mathcal{W}(a,b) = \int_D v(x) \frac{1}{\sqrt{a}} e^{-i\omega_0 \frac{x-b}{a}} dx$$

Performing $\tau = \frac{x-b}{a}$, then:

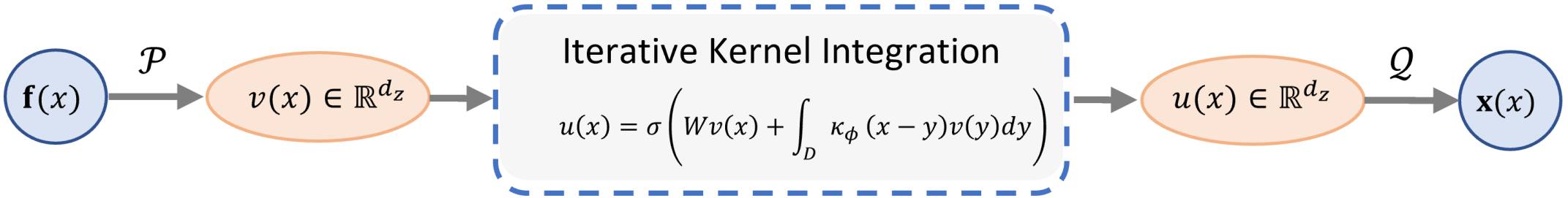
$$\mathcal{W}(a,b) = \sqrt{a} \int_D v(a\tau + b) e^{-i\omega_0 \tau} d\tau$$

when $a = 1$, and use window function $w(t)$, the above equation becomes short-time Fourier transform.

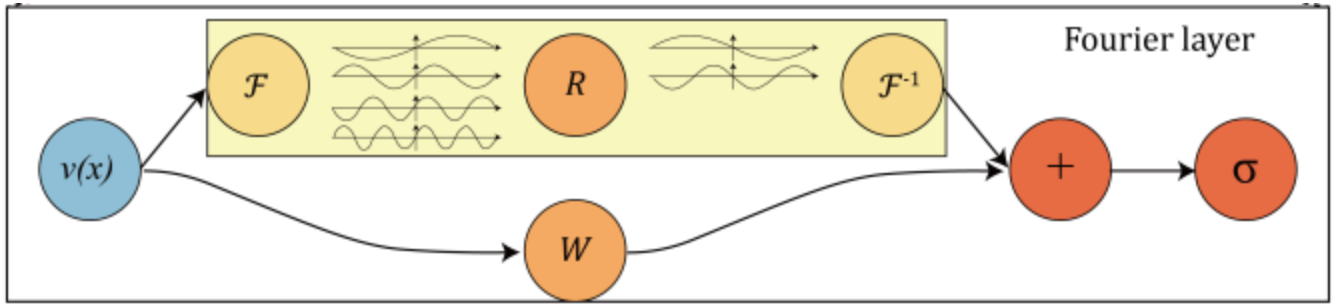
Since the short-time Fourier transform can be derived from wavelet transform, the FNO may be regarded as a special case of WNO [5].

[5] Tripura, T. and Chakraborty, S. (2022). Wavelet neural operator: a neural operator for parametric partial differential equations.

[6] Bruns, A. (2004). Fourier-, Hilbert-and wavelet-based signal analysis: are they really different approaches. Journal of Neuroscience Methods, 137(2), 321-332.



- Lifting input $f(x)$ to a higher dimensional representation $v(x)$ by a linear transformation $\mathcal{P}$
- Laplace layer/ Fourier layer/Wavelet layer
- Projecting $u(x)$ back to the original dimension $\mathcal{Q}$

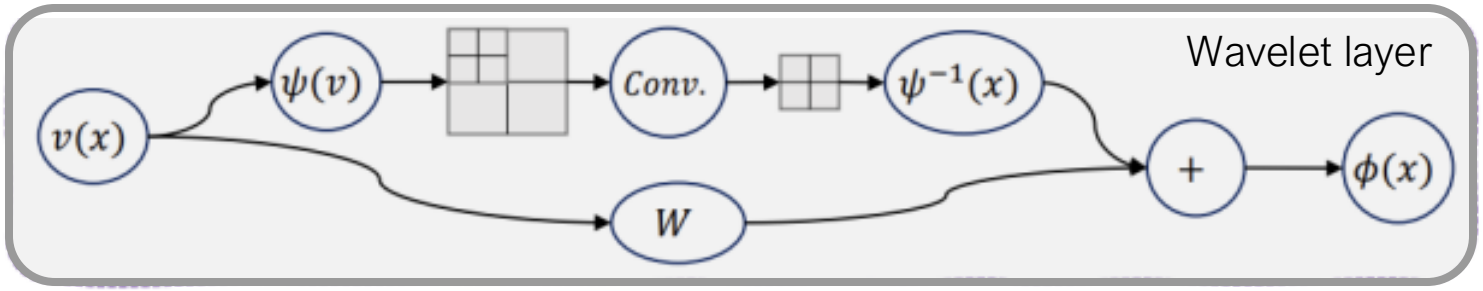


Schematic representation of Fourier layer (adopted from Ref. [4])

Procedure:

- Initialize training parameters $\boldsymbol{\theta} = (K_\phi(i\omega_1), \cdots, K_\phi(i\omega_1))$ in the frequency domain
- Obtain $\alpha(\omega)$ by performing FFT of input
- Perform the multiplication $U_1(\omega) = K_\phi(\omega)\alpha(\omega)$
- Perform IFFT of $U_1(\omega)$ to obtain the output of Fourier layer

[4] Li, Z. et al. (2020). Fourier neural operator for parametric partial differential equations.

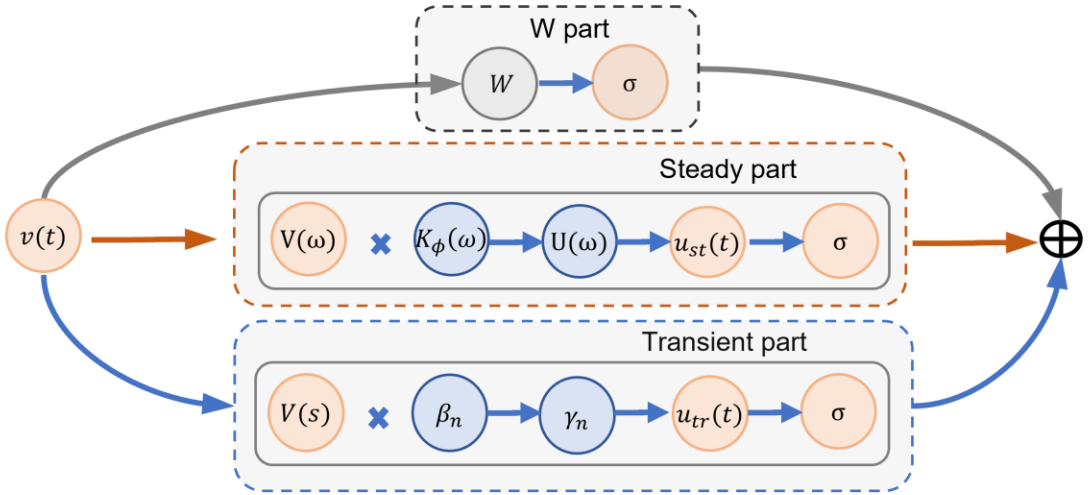


Schematic representation of wavelet layer (adopted from Ref. [5])

Procedure:

- Initialize training parameters $\boldsymbol{\theta} = (\mathcal{W}(k_{\phi 1}), \cdots, \mathcal{W}(k_{\phi N}))$ in the wavelet domain
- Obtain $\mathcal{W}(v)$ by performing wavelet transform of input
- Perform the multiplication $\mathcal{W}(u_1) = \mathcal{W}(k_\phi)\mathcal{W}(v)$
- Perform inverse wavelet transform of $\mathcal{W}(u_1)$ to obtain the output of wavelet layer

[5] Tripura, T. and Chakraborty, S. (2022). Wavelet neural operator: a neural operator for parametric partial differential equations.

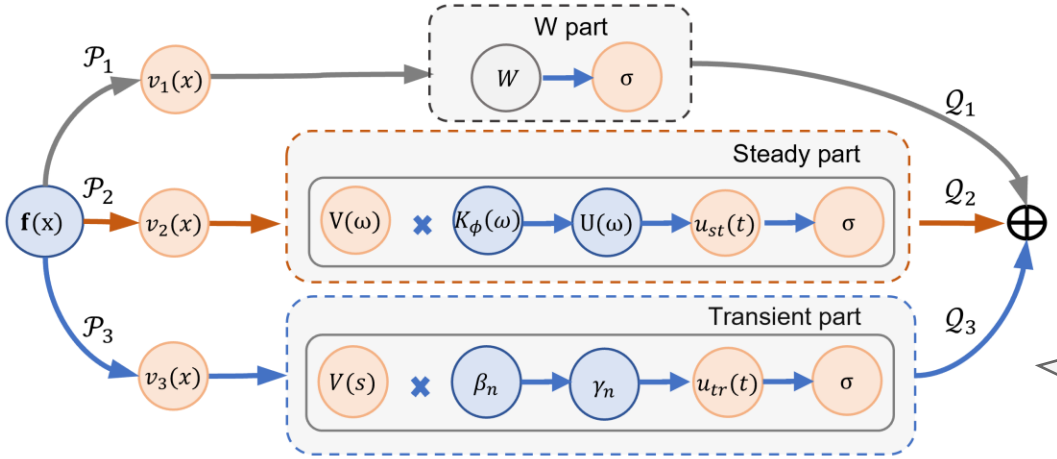


Schematic representation of Laplace layer (adopted from Ref. [3])

Procedure:

- Initialize training parameters $\boldsymbol{\theta} = (\mu_1, \cdots, \mu_N, \beta_1, \cdots, \beta_N,)$, where μ_n and β_n are trainable system poles and residues
- Obtain input poles $i\omega_\ell$ and residues α_ℓ by performing signal decomposition methods of input, such as FFT, Prony-SS
- Calculate output residues γ_n and λ_ℓ
- Compute output by $u(x) = \sum_{n=1}^{N} \gamma_n e^{\mu_n x} + \sum_{\ell=-\infty}^{\infty} \lambda_\ell e^{i\omega_\ell x}$

[3] Cao, Q., Goswami, S., & Karniadakis, G. E. (2023). LNO: Laplace neural operator for solving differential equations. .



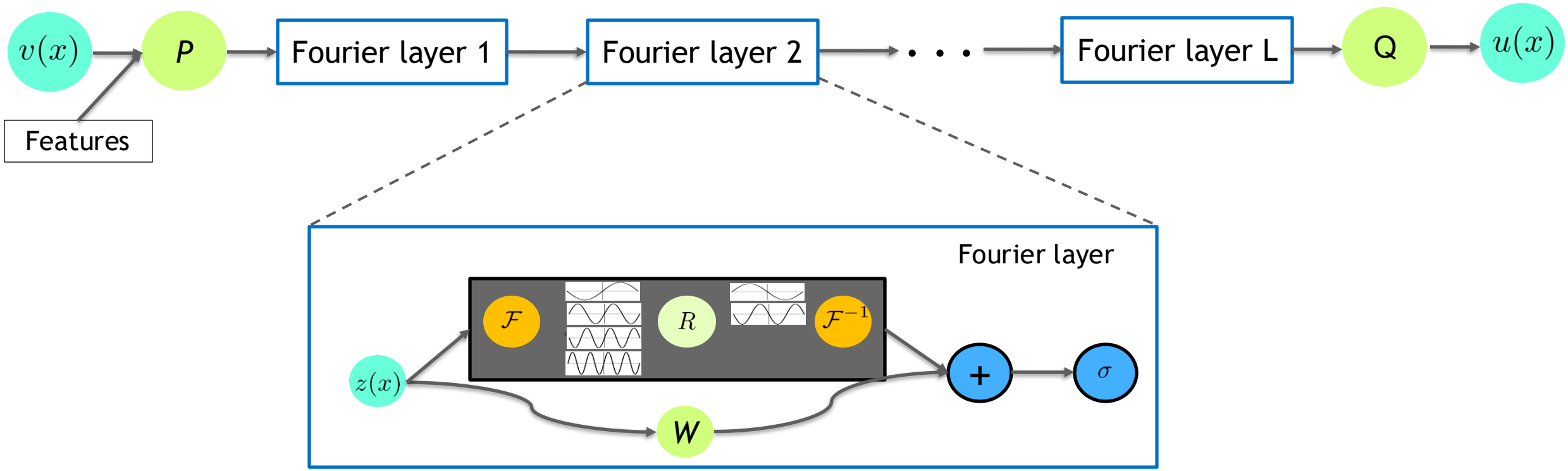
Schematic representation of advanced Laplace layer

- When transient part is zero, the advanced LNO becomes FNO.
- Three parts can have different lifting dimensions.

Procedure:

- Initialize training parameters $\boldsymbol{\theta} = \left(\mu_1, \cdots, \mu_N, \beta_1, \cdots, \beta_N, K_\phi(i\omega_1), \cdots, K_\phi(i\omega_1)\right)$
- Obtain input poles $i\omega_\ell$ and residues α_ℓ by performing FFT of input
- Perform the multiplication $U(\omega) = K_\phi(\omega)\alpha(\omega)$
- Compute steady-state output $u_{st}(x)$ by IFFT
- Calculate output residues γ_n
- Compute transient output by $u_{\mathrm{tr}}(x) = \sum_{n=1}^{N} \gamma_n e^{\mu_n x}$

Fourier Neural Operator (FNO) for Parametric Partial Differential Equations



Idea: The main network parameters are defined and learned in the Fourier space rather than the physical space.

- Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895. 2020 Oct 18.

Structure of FNO

Step 1: Function value $v(x)$ is lifted to a higher dimensional representation $z_0(x)$ by

$$z_0(x) = P(v(x)) \in \mathbb{R}^{d_z}$$

Transformation $P : \mathbb{R} \to \mathbb{R}^{d_z}$ is a shallow fully-connected NN or simply a linear layer. d_z is like the channel size in CNN.

Step 2: L Fourier layers are applied iteratively to z_0. z_L is the output of the last Fourier layer, and the dimension of $z_L(x)$ is d_z.

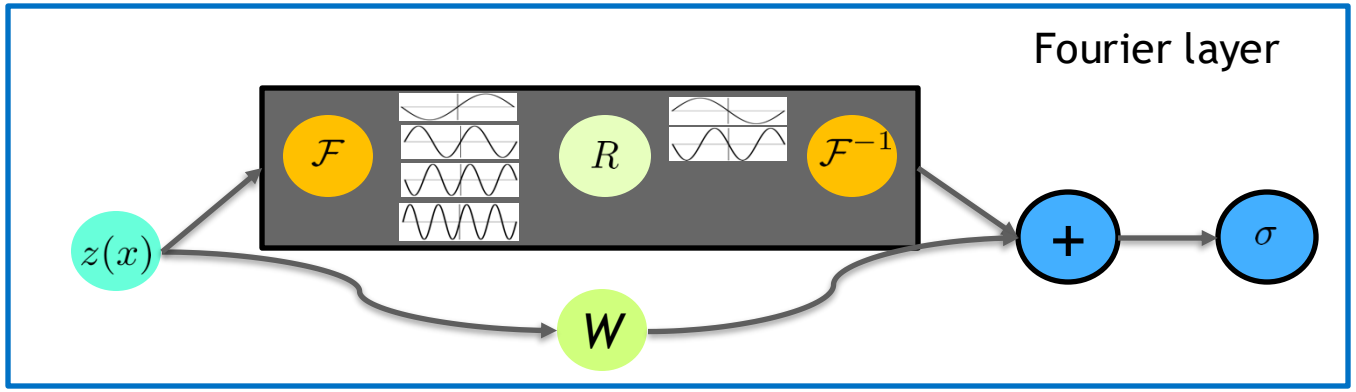
Step 3: Transformation $Q : \mathbb{R}^{d_z} \to \mathbb{R}$ is applied to project $z_L(x)$ to the output by

$$u(x) = Q(z_L(x))$$

Q is parameterized by a fully-connected NN.

- Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895. 2020 Oct 18.

Fourier Layer using Fast Fourier Transform (FFT)



For the output of the lth Fourier layer z_l with d_v channels:

Step 1: Compute the transform by FFT $\mathcal{F}$ and inverse FFT $\mathcal{F}^{-1}$:

$$\mathcal{F}^{-1}\left(R_l \cdot \mathcal{F}(z_l)\right)$$

$\mathcal{F}$ is applied to each channel of z_l separately; Truncate the higher modes of $\mathcal{F}(z_l)$, keeping only the first k Fourier modes in each channel. So $\mathcal{F}(z_l)$ has the shape $d_v \times k$.

Step 2: Apply a different (complex-number) weight matrix of shape $d_v \times d_v$ for each mode index of $\mathcal{F}(z_l)$ Have k trainable matrices, which form a weight tensor $R_l \in \mathbb{C}^{d_v \times d_v \times k}$. $R_l \cdot \mathcal{F}(z_l)$ has the same shape of $d_v \times k$ as $\mathcal{F}(z_l)$.

Step 3: Inverse FFT Need to append zeros to $R_l \cdot \mathcal{F}(z_l)$ to fill in the truncated modes.
Moreover, in each Fourier layer, a residual connection with a weight matrix $W_l \in \mathbb{R}^{d_v \times d_v}$. The output of the $(l+1)$th Fourier layer z_{l+1} is $\boldsymbol{z}_{l+1} = \sigma\left(\mathcal{F}^{-1}\left(R_l \cdot \mathcal{F}\left(\boldsymbol{z}_l\right)\right) + W_l \cdot \boldsymbol{z}_l + \boldsymbol{b}_l\right)$.

Implementation of Fourier Neural Operator (FNO)

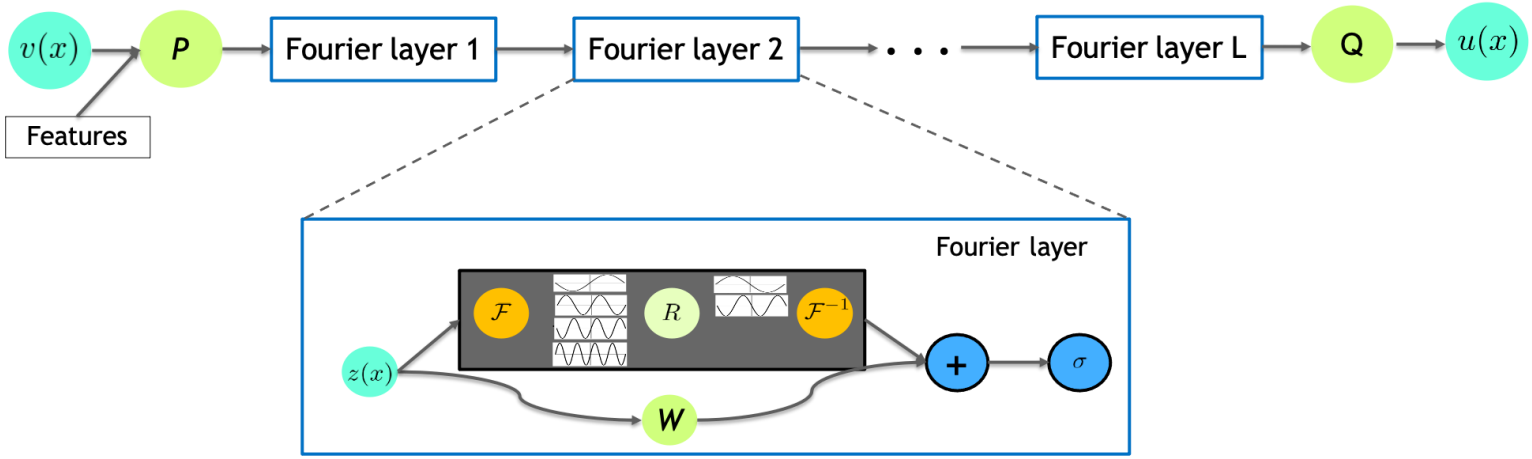
1. Consider the Burger's equation given by

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = \nu \frac{\partial^2 u}{\partial x^2}, \qquad x \in (0,1), \quad t \in (0,1]$$

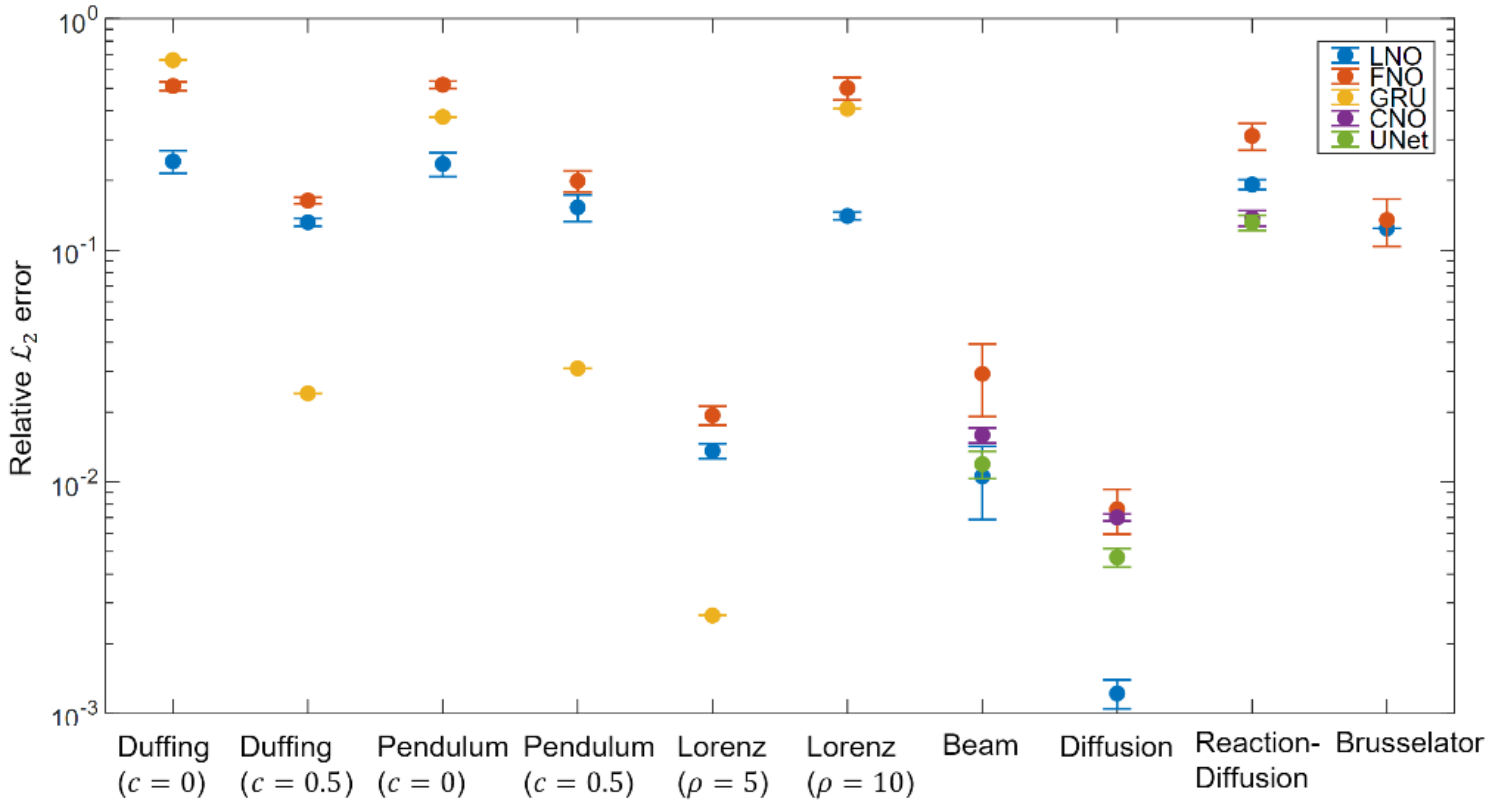
   and $\nu = 0.1$ with initial condition (IC) $u(x,0) = u_0(x)$ and periodic boundary condition.

2. We aim to learn the operator mapping the initial condition to the solution at time one $u_0 \to u(\cdot, 1)$

3. The data is generated for different initial condition $u_0 \sim \mathcal{N}(0, 625(-\Delta + 25\boldsymbol{I})^{-2})$



Dynamical Systems & PDEs：Relative $\mathcal{L}_2$ errors of several examples



Relative $\mathcal{L}_2$ error in the test cases for all the ODE and PDE cases and for different scenarios considered in each example. The plot shows the mean and the standard deviation of the error that has been computed based on five independent training trials.

Differences between DeepONet and FNO

Properties	DeepONet	FNO
Input domain D & Output domain D'	Arbitrary	Cuboid, $D = D'$
Discretization of output function u	No	Yes
Mesh	Arbitrary	Grid
Prediction location	Arbitrary	Grid points
Full field observation data	No	Yes

A Comprehensive and Fair Comparison of Two Operator-Regression Networks (with practical extensions) based on FAIR Data

DeepOnet@Brown vs. FNO@Caltech

Problems
Burgers' equation
5 Darcy problems in a rectangular domain and complex geometries
Multiphysics electroconvection problem
Earthquake problem
3 Advection problems
Linear instability waves in high-speed boundary layers
Compressible Euler equation with non-equilibrium chemistry
Predicting surface vorticity of a flapping airfoil
Navier-Stokes equation in the vorticity-velocity form
2 problems of regularized cavity flows

- Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z, Karniadakis GE. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. arXiv preprint arXiv:2111.05512. 2021 Nov 10.

dFNO+: Operators with inputs/outputs defined on different domains

FNO limitation: input function domain D and output function domain D' is the same.

Case I: The output space is a product space of the input space and another space D_O, i.e., $D' = D \times D_O$

Example: PDE solution operator mapping from IC to the solution in the whole domain

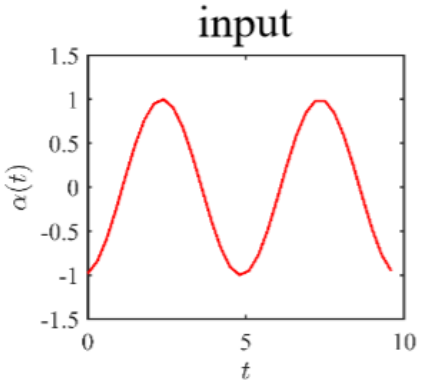
$$\mathcal{G} : v(x) = u(x, 0) \mapsto u(x, t)$$

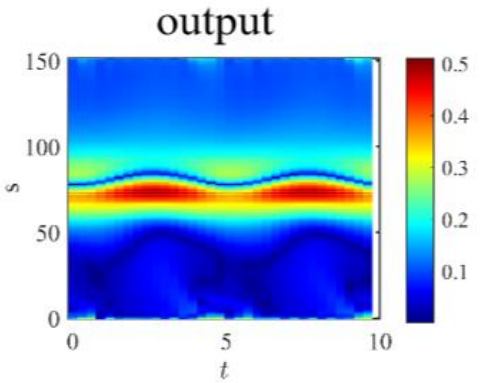
where $x \in [0, 1]$ and $t \in [0, T]$. Here, $D = [0, 1], D' = [0, 1] \times [0, T]$, and thus $D_0 = [0, T]$.

- Method 1: Extend the input domain by adding the extra coordinate t, defining $\tilde{v}$ as

$$\tilde{v}(x, t) = v(x).$$

  Then, FNO is used to learn the mapping from $\tilde{v}(x, t)$ to $u(x, t)$.
- Method 2: Shrink the ouput domain via RNN for time marching. Hard to train and unstable.





dFNO+: Operators with inputs/outputs defined on different domains

Case II: The input space is a subset of the output space, i.e., $D \subset D'$. in general we can extend v from D to D' by padding zeros in the domain $D' \backslash D$.

Example when padding is not efficient:
Consider a PDE defined on a rectangular domain $(x, y) \in D' = [0, 1]^2$, and the operator is the mapping from the Dirichlet boundary condition v defined in the four boundaries $(D = 1\{0, 1\} \times [0, 1] \cup [0, 1] \times \{0, 1\})$ to the solution $u(x, y)$ inside the rectangular domain.

Better strategy: First unfold the curve of v into a 1D function $\tilde{v}$ defined in $[0, 4]$:

$$\tilde{v}(\tilde{x}) = \begin{cases} v(\tilde{x}, 0), & \text{if } \tilde{x} \in [0, 1] \text{ (bottom boundary)} \\ v(1, \tilde{x} - 1), & \text{if } \tilde{x} \in [1, 2] \text{ (right boundary)} \\ v(3 - \tilde{x}, 1), & \text{if } \tilde{x} \in [2, 3] \text{ (top boundary)} \\ v(0, 4 - \tilde{x}), & \text{if } \tilde{x} \in [3, 4] \text{ (left boundary)} \end{cases}$$

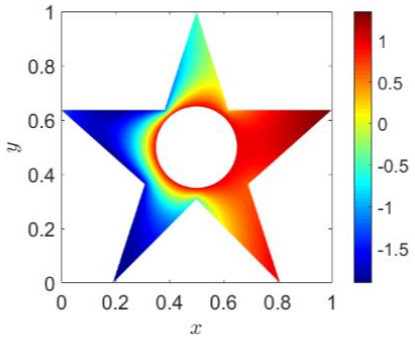
Then use the method in Case I to learn the operator from $\tilde{v}$ in 1D to u in 2D.

gFNO+ for Operators with inputs/outputs defined on a complex geometry

FNO uses FFT, which requires the input and output functions to be defined on a Cartesian domain with a lattice grid mesh. Two issues:

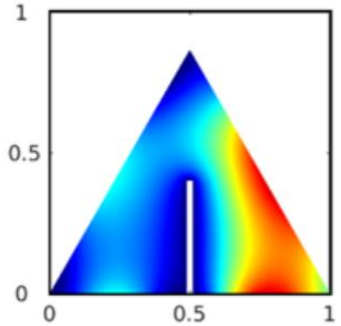
(1) non-Cartesian domain. Solution: "nearest neighbor" extension

(2) non-lattice mesh. Solution: interpolation to a lattice grid mesh



Nearest Neighbor extension:
Define the Cartesian domain $\tilde{D}$, which is the minimum bounding box of D

$$\tilde{v}(x) = \begin{cases} v(x), & \text{if } x \in D \\ v_0(x), & \text{if } x \in \tilde{D} \backslash D \end{cases}$$



The choice of $v_0(x)$ is not unique, such as $v_0(x) = 0$, i.e., zero padding.
Recommend: **nearest neighbor** so that $\tilde{v}(x)$ is continuous on the boundary of D.
for $x \in \tilde{D} \backslash D, \quad v_0(x) = v\left(x_0\right), \quad$ where $x_0 = \min_{p \in D} \|p - x\|$,

DeepONet and FNO with feature expansion

Features of the system: prior knowledge about the underlying system. For example, the prior knowledge of output functions being oscillating in nature or the functions being fast decaying.

For better approximation: encode the knowledge by modifying the network architecture.

DeepONet: Feature expansion either in the trunk net or in the branch net.

- Feature expansion in the trunk net: If features are functions of ξ
- For example, harmonic feature expansion

$$\xi \mapsto (\xi, \cos(\xi), \sin(\xi), \cos(2\xi), \sin(2\xi), \ldots)$$

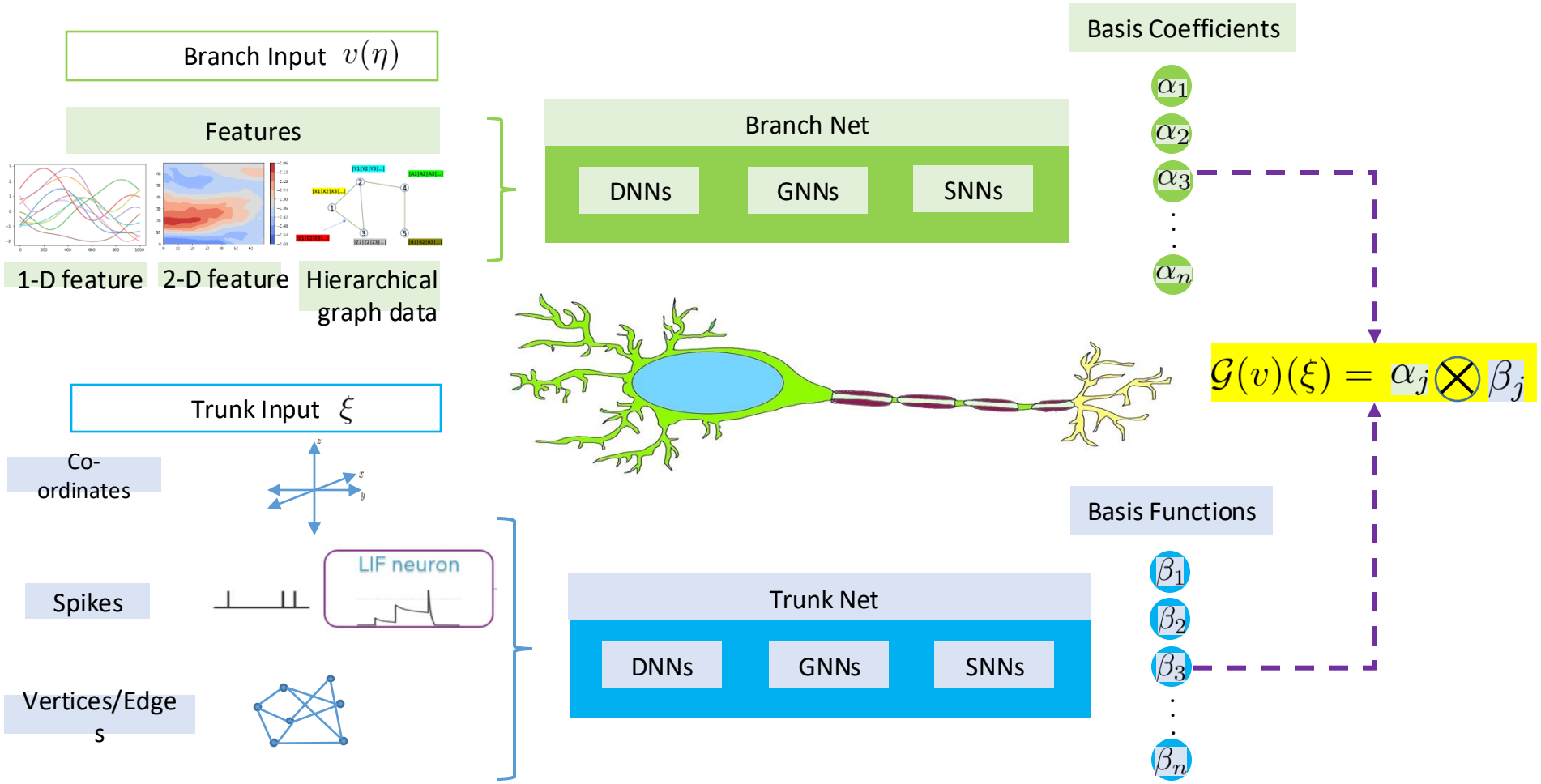
- Feature expansion in the branch net: If the feature is a function of x

FNO: Features are applied by using them as extra network inputs

- For example, the coordinates x and feature $f(x)$, then the FNO input is

$$(x, v(x), f(x)) \in \mathbb{R}^{d+2}$$

DeepOnet provides a plurality of Neural Operators



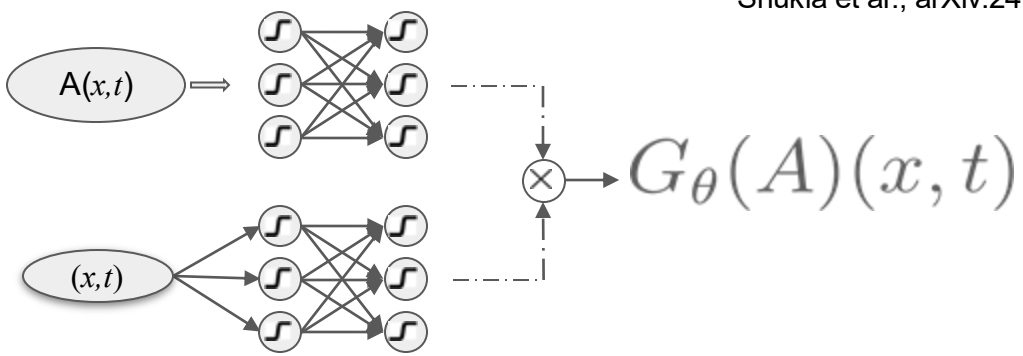
DeeOnet with Kolmogorov Arnold Network (KAN) on the trunk: DeepOKAN

DeepONet

KAN refs: github.com/KindXiaoming/pykan
Shukla et al., arXiv:2406.02917

[416,200,100,100,32] *branch net*

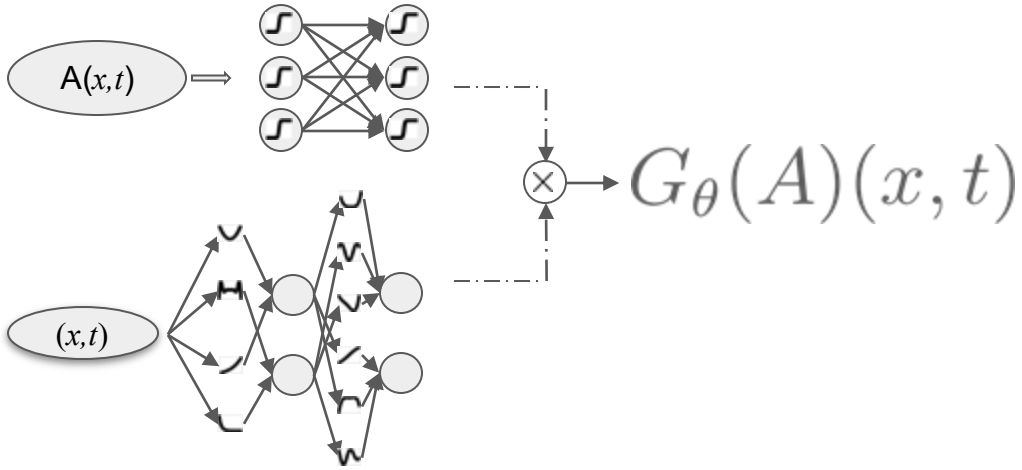
[3,16,24,32] *trunk net*



DeepONet+KAN

[416,200,100,100,32] *branch net*

[3,32,32,32]
Chebyshev polynomial order 7 *trunk net*



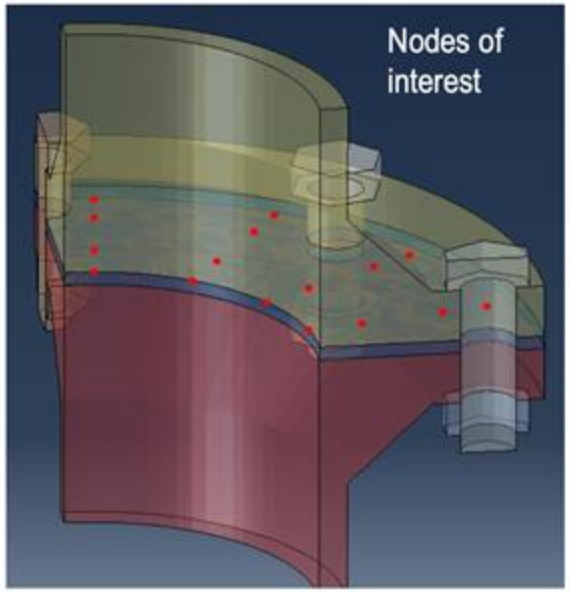
Thermal Reactor Data

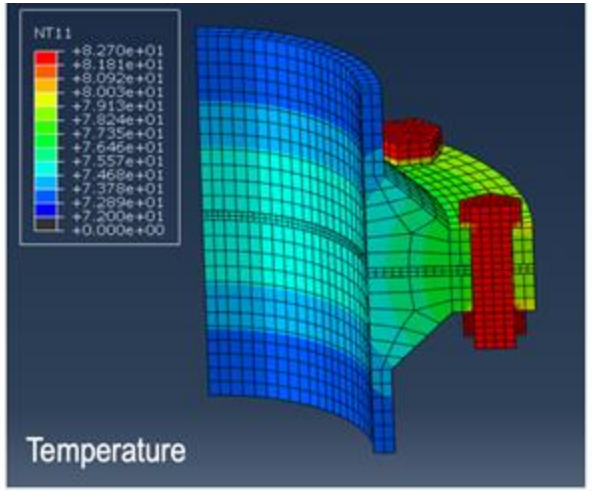
43 cases, each case has temperature data at 16 nodes in time domain [0, 2500s]

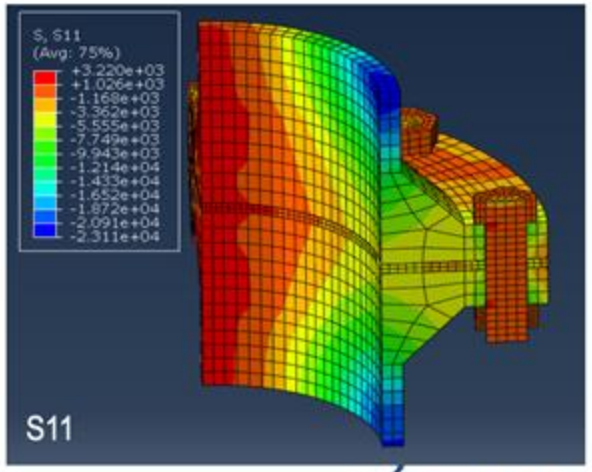
PROBLEM STATEMENT – PRESSURE VESSEL

Staggered Heat transfer – Stress Analysis (cont'd)

- **Stress Analysis**
  - **Reads temperature distribution** from heat transfer analysis
  - **Fixed time dependent internal pressure** on the inside of the vessel (highlighted)
    - These amplitudes ARE NOT used as input features
  - Contact interaction between parts
  - Bolt preload
  - Initial steady state
  - **Transient (step of interest – training prediction)**
- **Outputs**
  - A subset of nodes (ideally, the entire field)
  - Nodal temperature
  - S11 (ideally Sij)







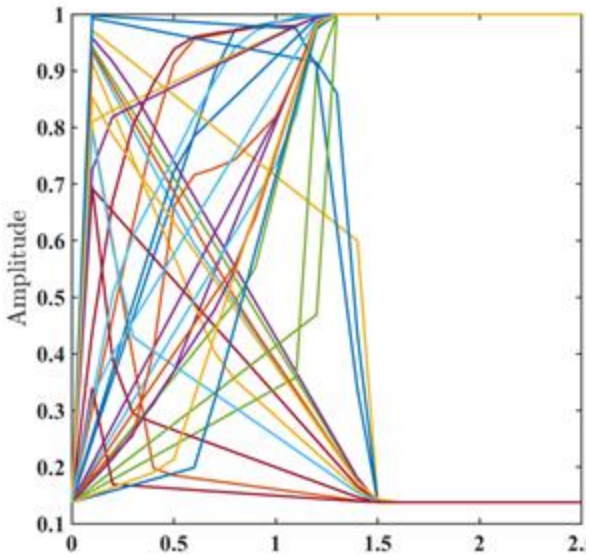
DASSAULT SYSTEMES

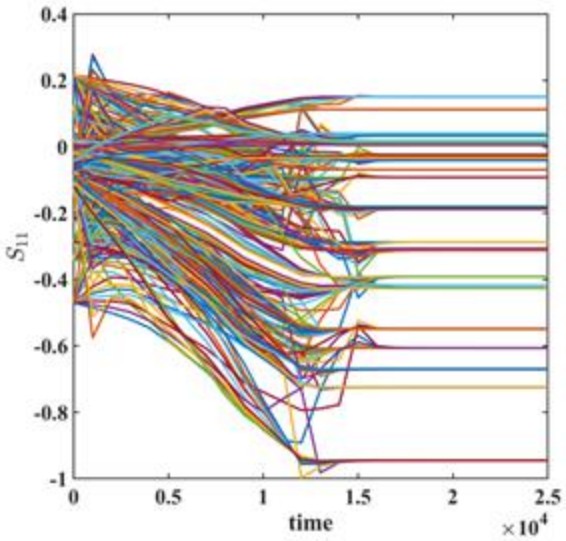
DeepONet setup

find a mapping from amplitude(A(x,*t*)) to stress (S11(x,*t*))

Data: 43 cases; 34 for training; 9 for testing (randomly selected)

Trunk net: **x → (*x*,*z*), node coordinates** **t → time**





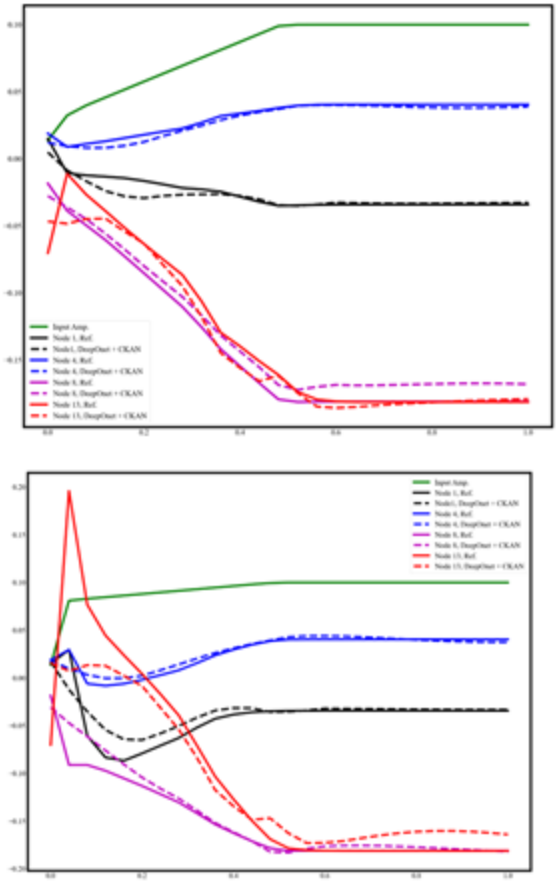
Branch net: **A(x,*t*)**

Output: **S11(x,*t*)**

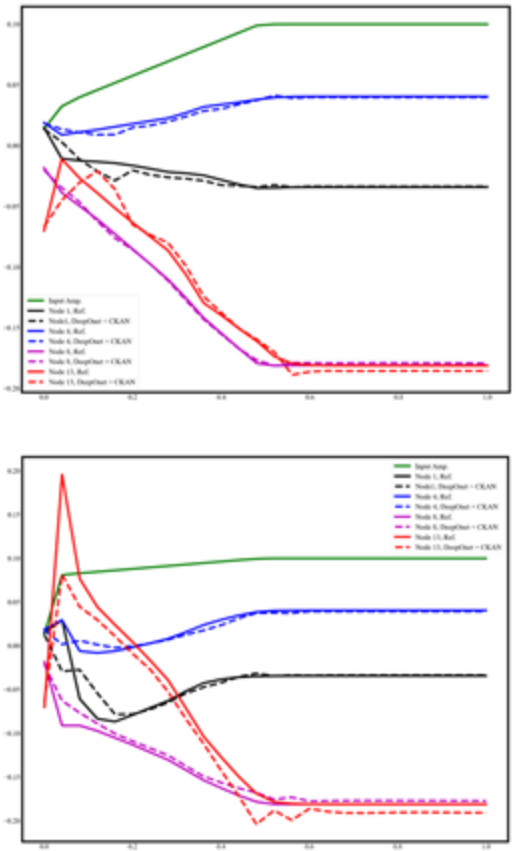
A(x,*t*) and S11(x,*t*) are uniformly interpolated into 26 points in time domain [0, 25000s]
A(x,*t*) is rescaled by 1/508
S11(x,*t*) is rescaled by 8000

Comparison between MLP and KAN on S11 stress

DeepOnet
test error: 6.63%



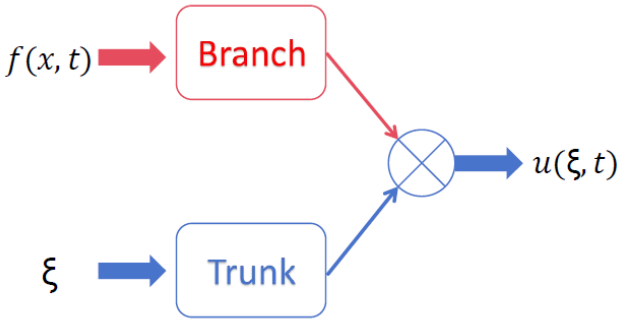
DeepOnet+KAN
Test error: 2.60%



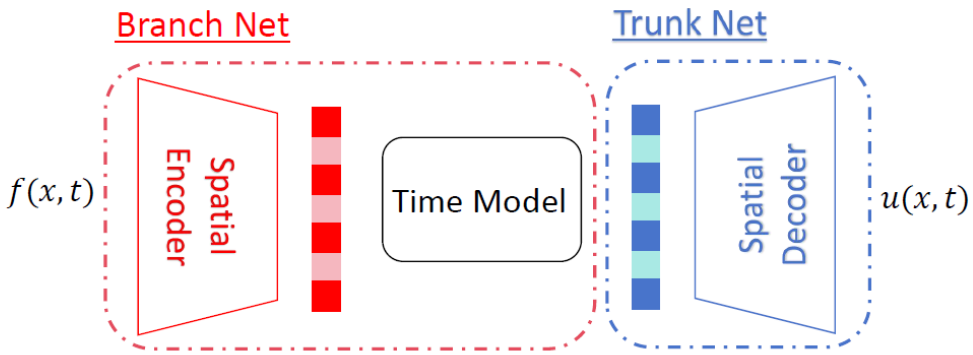
Model Architectures: DeepONet + Time Model

DeepO-Mamba

(A) DeepONet Structure



(B) DeepONet + Time Model Structure



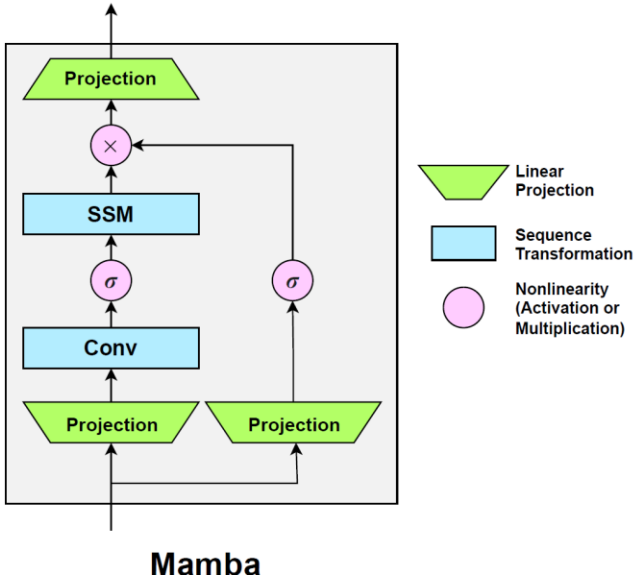
- The encoder + Mamba part can be regarded as the branch net, while the decoder can be regarded as the trunk net in the DeepONet framework.

State Space Model (SSM) and Mamba

SSM is a sequence-to-sequence (seq2seq) mapping:

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t). \tag{1}$$
$$y(t) = \boldsymbol{C}x(t). \tag{2}$$

It maps the input sequence $u(t)$ to the output sequence $y(t)$ via the hidden state $x(t)$.
Modern SSM like Mamba is based on the above simple equation, and improves the model via input-dependent $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ parameterization, ML system design, and implementation consideration (group norm, skip connection, etc.).
The entire SSM is constructed by stacking multiple SSM blocks, where the SSM dynamical system mapping is combined with nonlinearity and another module, such as feedforward and convolution networks.



Mamba

DeepO-Mamba: BELTRAMI FLOW

Type	Model	Params	Time	Memory	Relative L_2 Error
DeepONets	DeepO-GRU	460,672	11 min	15239MiB	1.708E-01±1.539E-02
	DeepO-LSTM	592,256	11 min	15193MiB	1.188E-01±2.054E-02
	DeepO-Mamba	504,704	**6 min**	15023MiB	**8.761E-04±3.956E-04**
	DeepO-Oformer-V	1,313,152	6 min	15151MiB	Diverge
	DeepO-Oformer-G	1,312,640	6 min	15149MiB	Diverge
	DeepO-Oformer-F	1,312,640	6 min	15149MiB	Diverge
	DeepO-Transformer	462,208	7 min	15099MiB	2.297E-02±3.197E-03
	DeepO-GNOT	2,824,578	12 min	15257MiB	Diverge
FNOs	FNO4d	11,944,427	111 min	51107MiB	1.499E-02±1.581E-03
	FFNO4d	181,155	901 min	71793MiB	1.382E-02±1.205E-03
	FNO3d+GRU v1	749,891	262 min	64215MiB	3.328E-02±3.455E-03
	FNO3d+GRU v2	749,891	263 min	64215MiB	3.766E-02±4.022E-03
	FNO3d+LSTM v1	750,435	202 min	61811MiB	3.038E-02±3.290E-03
	FNO3d+LSTM v2	750,435	203 min	61811MiB	3.626E-02±4.785E-03
	FNO3d+Mamba v1	757,859	576 min	77773MiB	2.196E-02±2.470E-03
	FNO3d+Mamba v2	757,859	577 min	77773MiB	2.034E-02±1.892E-03

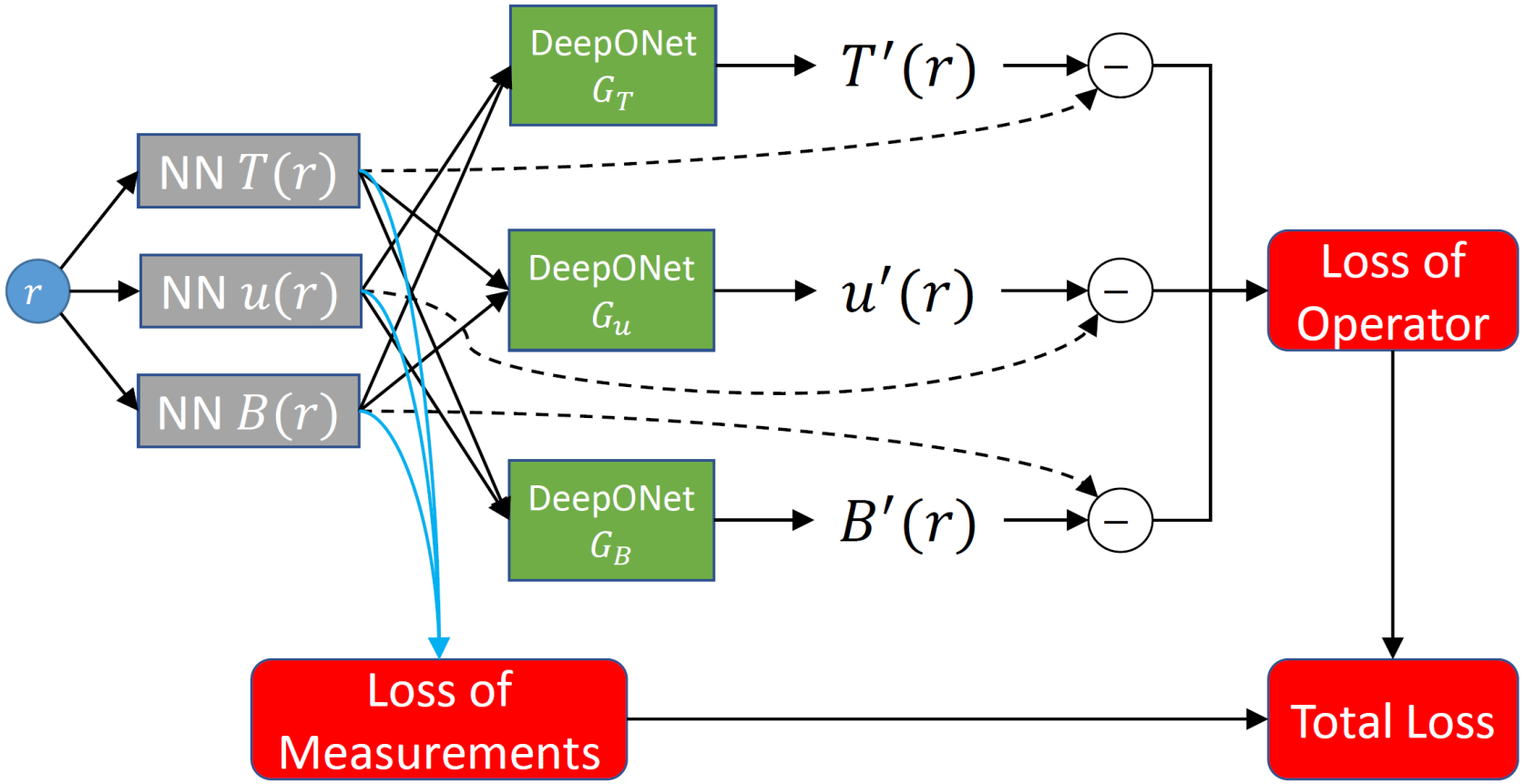
- Our latent approach cleverly encodes the high-dimensional PDE solutions discretized on the 17^3 3D dense spatial grid to very low-dimensional vectors, facilitating SSMs to learn the temporal dynamics.

- In contrast, FNO requires inputting the entire huge 3D tensor corresponding to the discretized 3D grid, incurring considerable cost and slow training.

- In DeepO-X, Mamba clearly outperforms to propagate the entire temporal dynamics in the complicated latent space.

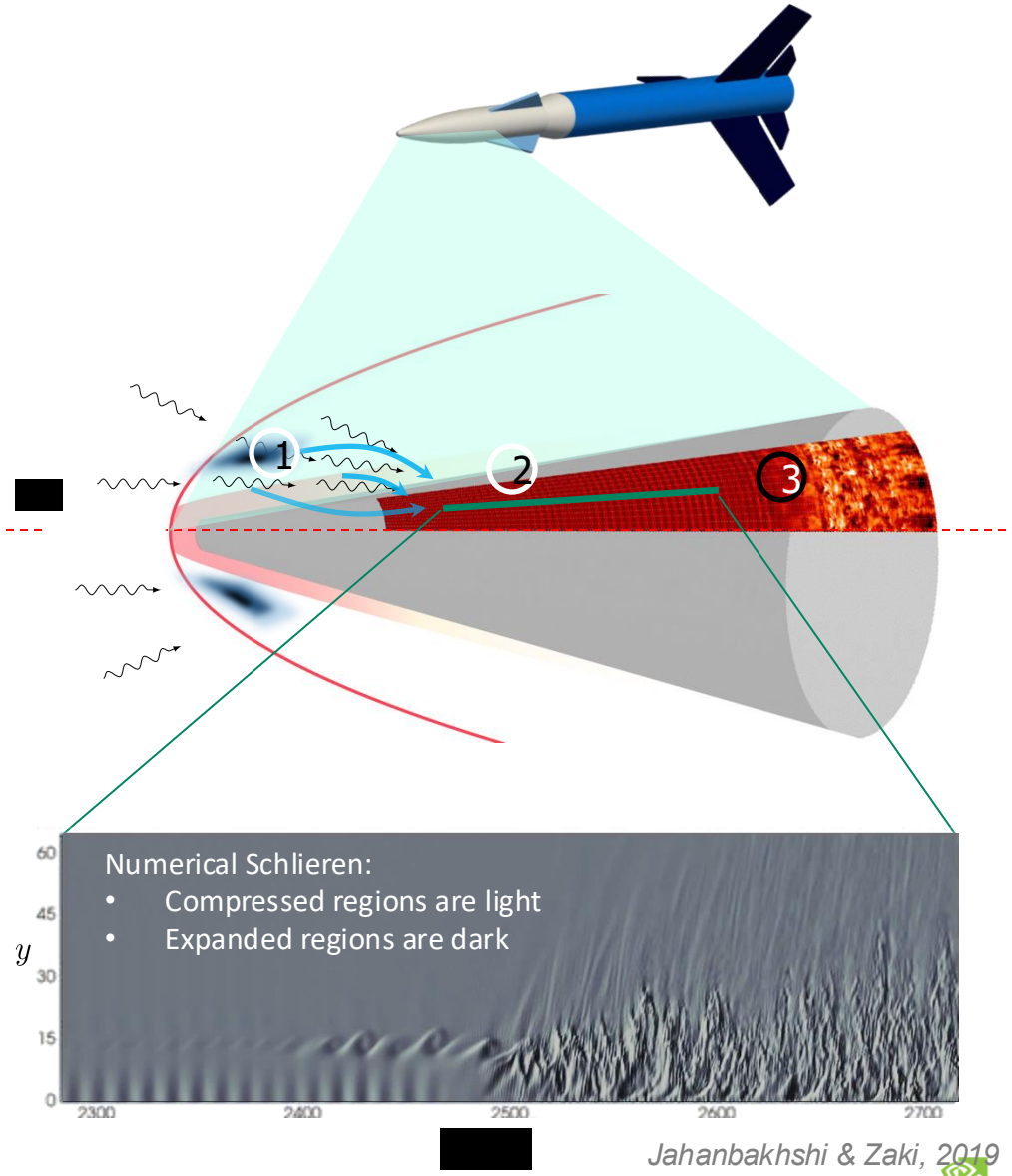
- In FNO+X, the temporal model's effect seems minor as it only propagates partial temporal dynamics on each fixed spatial grid but not the entire dynamics.arial

Deep Neural Operators as Foundation Models for Digital Twins



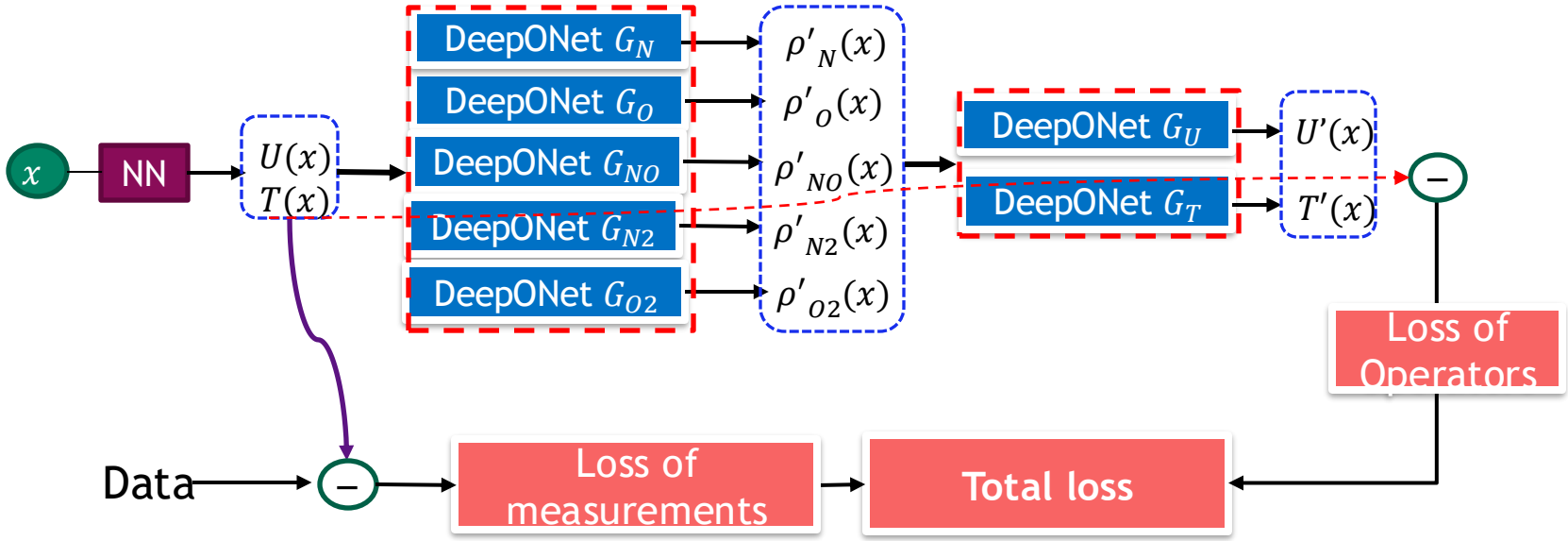
DeepONets for Compressible Non-Equilibrium Flows

- A complex, multi-physics and multi-scale problem with scarce data
- Instability waves interact with the boundary layer on the flight vehicle and can amplify exponentially.
- At high Mach numbers, the gas dissociation/association rates are commensurate with the flow.
- Once finite amplitude perturbations form, nonlinear interaction spur the generation of harmonics and ultimately break down to turbulence.
- Many instability waves are possible, each with unique characteristics.
- The flow state is extremely sensitive to disturbance environment.



Jahanbakhshi & Zaki, 2019

Compressible Navier-Stokes with finite-rate chemistry



Minimize $\mathcal{L}$,

$$\mathcal{L} = \omega_D \frac{1}{n_d}\mathcal{L}_{\text{data}} + \omega_O \frac{1}{n_O}\mathcal{L}_{op} + \omega_R \mathcal{L}_{reg} + \omega_G \frac{1}{n_G}\mathcal{L}_G$$

$$\mathcal{L}_{\text{data}} = \sum_{j=1}^{n_d} \left\| U^j_{\text{data}} - U(x_j) \right\|^2 + \sum_{j=1}^{n_d} \left\| T^j_{\text{data}} - T(x_j) \right\|^2$$

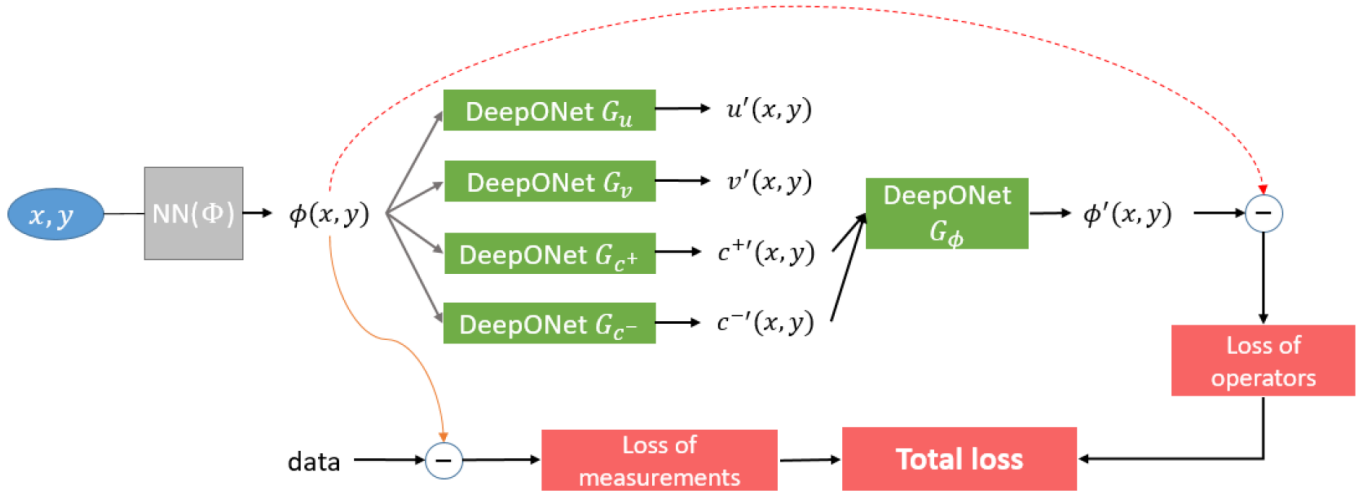
$$\mathcal{L}_{\text{op}} = \sum_{j=1}^{n_{op}} \|U^*(x_j) - U(x_j)\|^2 + \sum_{j=1}^{n_{op}} \|T^*(x_j) - T(x_j)\|^2$$

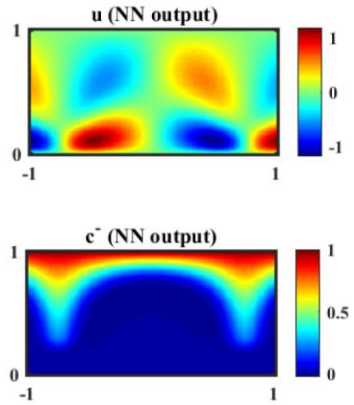
$$\mathcal{L}_G = \sum_{j=1}^{n_G} \|\rho^* U^*(x_j) - \text{ Const } .\|^2$$

$$\mathcal{L}_{reg} = ||\theta||_2^2$$
where, $\rho^* = \rho^*_{N_2} + \rho^*_{0_2} + \rho^*_N + \rho^*_0 + \rho^*_{N0}$

- Mao Z, Lu L, Marxen O, Zaki TA, Karniadakis GE. DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Journal of Computational Physics. 2021 Dec 15;447:110698.

Simulation of Electro-Convection





Governing Equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla p + \nabla^2 \mathbf{u} + \mathbf{f_e},$$
$$\nabla \cdot \mathbf{u} = 0,$$
$$-2\epsilon^2 \nabla^2 \phi = \rho_e,$$
$$\frac{\partial c^{\pm}}{\partial t} = -\nabla \cdot \mathbf{J}^{\pm},$$
$$\mathbf{J}^{\pm} = c^{\pm}\mathbf{u} - \nabla c^{\pm} \mp c^{\pm}\nabla\phi.$$

- A neural network is used to approximate the solution of ϕ
- u, v, c^+, c^- are hidden outputs
- Loss function:

$$\arg\min_{\Phi} \mathcal{L} = \mathcal{L}_{op} + \mathcal{L}_{\text{data}} + \mathcal{L}_2(\Phi)$$

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d}\sum_{i=1}^{N_d} \left\| \phi_{\text{data}}\left(x^i, y^i\right) - \phi\left(x^i, y^i\right) \right\|^2$$

$$\mathcal{L}_{op} = \frac{1}{N_{op}}\sum_{i=1}^{N_{op}} \left\| \phi'\left(x^i, y^i\right) - \phi\left(x^i, y^i\right) \right\|^2$$

- Cai S, Wang Z, Lu L, Zaki TA, Karniadakis GE. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. Journal of Computational Physics. 2021 Jul 1;436:110296.

Performance Metrics

Problem	# of physical fields	# of scales 10^n	Accuracy 10^{-m}	Speed-up	Reference
Electroconvection	6	3	3	10,000X	J. Comp. Phys. 2021
Hypersonic flow	7	8	3	100,000X	J. Comp. Phys. 2021
High-speed boundary layer	5	2	3	40,000X	***arXiv:2105.08697***

Important considerations

N_s: Number of simulations to generate data
C_s: Cost of performing one simulation

C_t: Cost of training DeepONet
C_e: Cost of evaluating DeepONet

New metrics

- Training cost ratio $R_t = C_t / N_s C_s$
- Evaluation cost ratio $R_e = C_e / C_s$
- Break-even number $N_e = N_s + C_t / C_s$

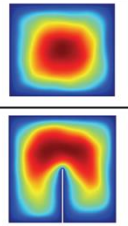
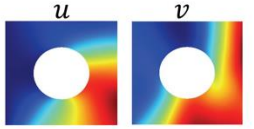
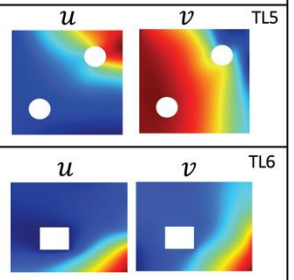
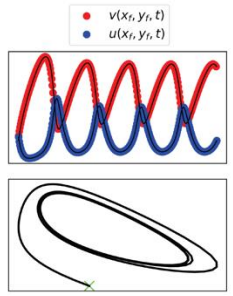
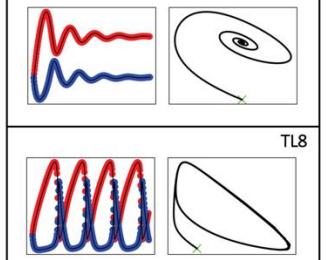
Compressible Boundary Layer

- $R_t = 3.39$
- $R_e = 2.7 \times 10^{-5}$
- $N_e = 250$

▪ Mao Z, Lu L, Marxen O, Zaki TA, Karniadakis GE. DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Journal of Computational Physics. 2021 Dec 15;447:110698.

DeepOnet as Foundation Model – Transfer Learning

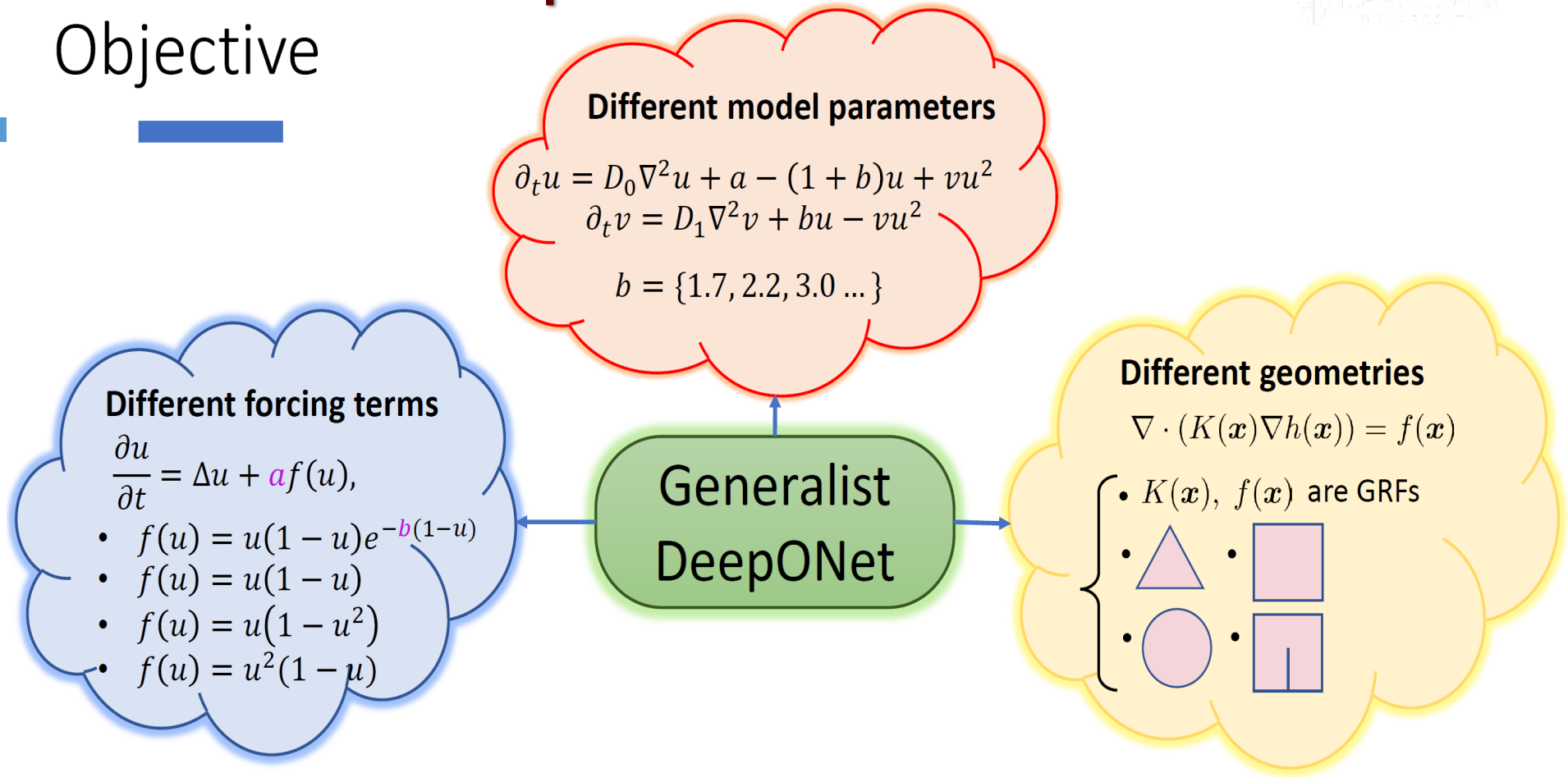
- **Transfer-Learning** has continuous learning but stops after a few attempts; it does not care about the performance of the source task when transferring knowledge (hence only forward transfer).

Application	Input Function	Model Output	Domain Visualization	
			Source	Target/s
Darcy Flow	Random input conductivity field $K(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$ $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(x-x')^2}{2l^2}\right]$ $l = 0.25, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]^2$	$\nabla.\left(K(\boldsymbol{x})\nabla h(\boldsymbol{x})\right) = 1$ $h(\boldsymbol{x}) = 0 \quad \forall \quad \boldsymbol{x} \in \partial\Omega$ $\mathcal{G}_\theta: K(\boldsymbol{x}) \rightarrow h(\boldsymbol{x})$		TL1 TL2 TL3 TL4
Elasticity Model	Random boundary conditions $\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$ $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}')\hat{}2}{2l^2}\right]$ $l = 0.12, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]$	$\nabla.\boldsymbol{\sigma} + \boldsymbol{f}(\boldsymbol{x}) = 0$ $(u, v) = 0 \; \forall \; x = 0$ $\mathcal{G}_\theta: f(\boldsymbol{x}) \rightarrow (u, v)$ u: X-Displacement v: Y-Displacement Material properties $E_S = 300 \cdot 10^5$, $\nu_S = 0.3$ $E_{T_1} = 410 \cdot 10^3, \nu_{T_1} = 0.35$ $E_{T_2} = 410 \cdot 10^3, \nu_{T_2} = 0.45$		 TL5 TL6
Brusselator Diffusion-Reaction System	Random initial condition $h_2(\boldsymbol{x}) \sim \mathcal{GP}(h_2(\boldsymbol{x})\|\mu(\boldsymbol{x}), \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$ $v(x, y, t = 0) = h_2(x, y) \geq 0$ $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}')\hat{}2}{2l^2}\right]$ $l_x = 0.12, l_y = 0.4, \sigma^2 = 0.15$	$\frac{\partial u}{\partial t} = D_0 \nabla^2 u + a - (1-b)u + vu^2$ $\frac{\partial v}{\partial t} = D_1 \nabla^2 v + bu - vu^2$ $x \in [0,1]^2, t \in [0,1]$ $\mathcal{G}_\theta: h_2(x, y) \rightarrow v(x, y, t)$ Model parameter $b_S = 2.2$ $b_{T_1} = 1.7$, $b_{T_2} = 3.0$		 TL7 TL8

- **Deep transfer operator learning for partial differential equations under conditional shift for partial differential equations under conditional shift , Goswami et al,** *Nature Machine Intelligence (December, 2022)*

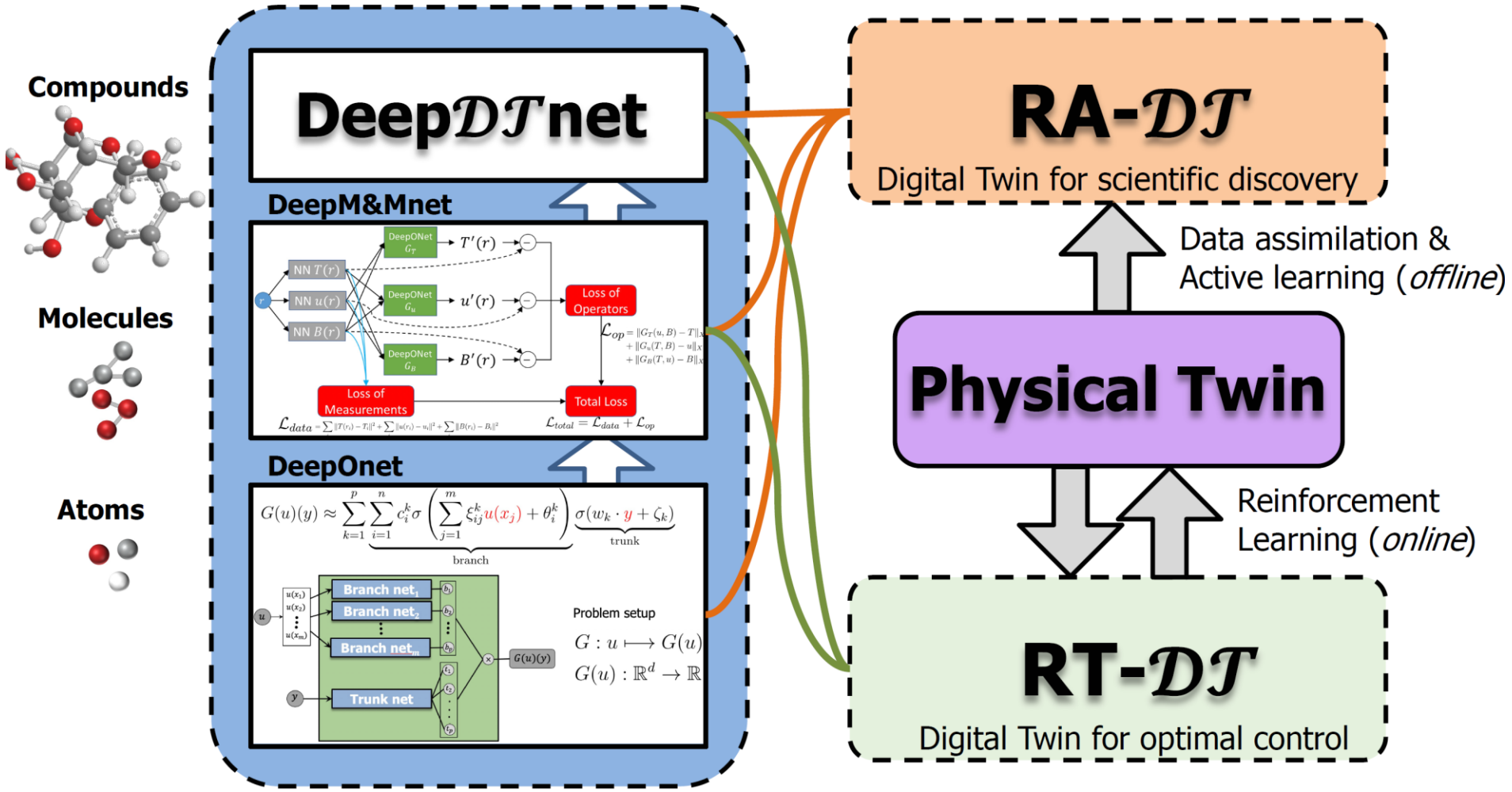
Multi-Task Neural Operators

Objective



- Multi-task regressor model for solving multiple PDEs at once;
- Improve accuracy and efficiency compared to individual DeepONet
- Network will use same trained parameters to solve PDEs
- **How?**: Incorporate information to the input data or modify trunk net

Building DeepDTnet for Digital Twins from Bottom Up



Summary

- Neural networks are universal approximators of functions, functionals and nonlinear operators
- DeepOnet was designed based on theorem of Chen & Chen and it was extended to deep NNs
- DeepOnet has some resemblance to biological neurons
- DeepOnet converges exponentially fast with the training data but in practice it saturates for big data
- In addition to learning mathematical operators, DeepOnet can learn multiscale operators
- Physics-informed DeepOnet can enhance accuracy and generalization but hybrid training is more robust
- Fourier neural operator (FNO) is a subcase of the DeepOnet framework
- Neural operators can be used as building blocks for multiphysics problems and digital twins
- The accuracy of neural operators can be enhanced with expansion features added to the branch and trunk nets
- Rigorous theory for neural operators has already been developed

Advantages and Limitations of Different Methods

	WNO	FNO	LNO	Advanced LNO
Advantages	• Accurate tracking of patterns in spatial domain • Can handle PDEs with discontinuities and abrupt changes in the solution domain and the boundary	• Fastest with similar training parameters • Highest accuracy for interpolation in simple cases	• Consider both transient and steady-state response • Highest accuracy for extrapolation • Highest accuracy for no damping systems and systems with multiple equilibrium • Predict outputs at any arbitrary locations	• Include all advantages from FNO and LNO
Limitations	• Less accurate than FNO for interpolation in simple cases • Needs to try different types of discrete wavelets	• Only steady-state response • Input and output have same discretization	• Less accurate than FNO for interpolation in simple cases due to strict formulation of training parameters	• More number of parameters need to be trained and tuned.

References

- Cai S, Wang Z, Lu L, Zaki TA, Karniadakis GE. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. Journal of Computational Physics. 2021 Jul 1;436:110296.
- Goswami S, Yin M, Yu Y, Karniadakis GE. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. Computer Methods in Applied Mechanics and Engineering. 2022 Mar 1;391:114587.
- Lanthaler S, Mishra S, Karniadakis GE. Error estimates for DeepOnets: A deep learning framework in infinite dimensions. arXiv preprint arXiv:2102.09618. 2021 Feb 18.
- Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895. 2020 Oct 18.
- Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence. 2021 Mar;3(3):218-29.
- Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z, Karniadakis GE. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. arXiv preprint arXiv:2111.05512. 2021 Nov 10.
- Mao Z, Lu L, Marxen O, Zaki TA, Karniadakis GE. DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Journal of Computational Physics. 2021 Dec 15;447:110698.

References

- del Aguila Ferrandis J, Chryssostomidis C, Triantafyllou M, Karniadakis, G.E. Learning functionals via LSTM neural networks for predicting vessel dynamics in extreme sea states, Proc. R. Soc. A.47720190897, 2021.

- Lin C, Li Z, Lu L, Cai S, Maxey M, Karniadakis GE. Operator learning for predicting multiscale bubble growth dynamics. The Journal of Chemical Physics. 2021 Mar 14;154(10):104118.

- Lin C, Maxey M, Li Z, Karniadakis GE. A seamless multiscale operator neural network for inferring bubble dynamics. Journal of Fluid Mechanics. 2021 Dec; 929.

- Marcati C, Schwab C. Exponential convergence of Deep Operator Networks for elliptic partial differential equations. arXiv preprint arXiv:2112.08125. 2021 Dec 15.

- Chen TP and Chen H, Approximations of continuous functionals by neural networks with application to dynamic systems, IEEE Transactions on Neural Networks, 910-918, 4(6), 1993.

- Chen TP and Chen H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Transactions on Neural Networks, 6(4):911–917, 1995.

- Wang S, Wang H, Perdikaris P. Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. arXiv preprint arXiv:2103.10974. 2021 Mar 19.

Deep Learning for Science and Engineering Teaching Kit

Thank You