Neural operator

Dr. Souvik Chakraborty
Department of Applied Mechanics
Indian Institute of Technology Delhi
Hauz Khas – 110016, Delhi, India.

E-mail: souvik@am.iitd.ac.in
Website: https://www.csccm.in/

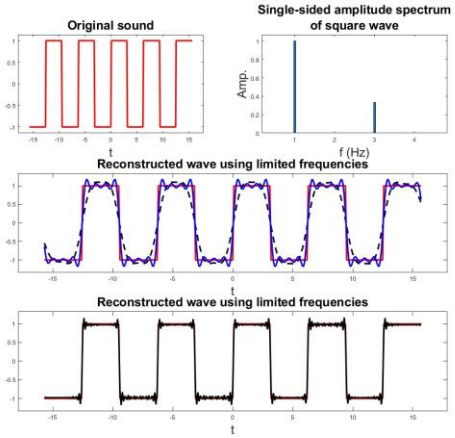
Recap

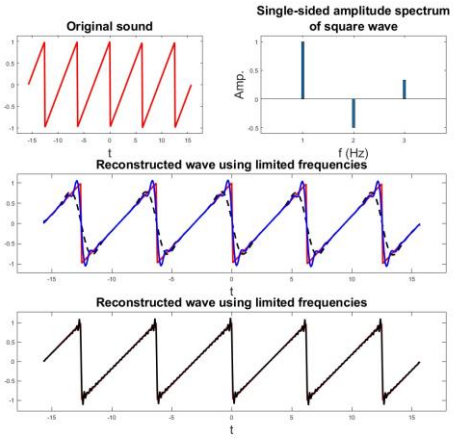
- An operator is defined as a mapping from a space of functions into another space of functions.

$$u = (K_l \circ \sigma_l \circ \cdots \circ \sigma_1 \circ K_0)v$$

$$u(x) = \int_\Omega k(x,y)\,\psi(y)dy + g(x)$$

$$F(\omega) = \int f(x)e^{-i\omega x}dx$$





$K(x,y)$ ← kernel

For stationary kernel,

$$k(x,y) = k(x-y)$$

$$\int k(x-y)\, f(y)\, dy \quad -(1)$$

$$F(\omega) = \int f(x)\, e^{-i\omega x}\, dx \quad -(2)$$

Substitute (1) into (2),

$$F(\omega) = \int\left(\int k(x-y)\, f(y)\, dy\right) e^{-i\omega x}\, dx$$

$$F(\omega) = \int \left(f(y) \int k(x-y) \right) e^{-i\omega x} \, dx \, dy$$

Let,

$$x - y = z \quad \Rightarrow \quad x = y + z$$

Fourier transform of the integral kernel

- The mathematical equation for the Fourier transform is as follows:

$$F(\omega) = \int f(x) e^{-i\omega x} dx$$

- The integral kernel takes the following form:

$$\int_\Omega k(x,y)\,\psi(y) dy = \int_\Omega k(x-y)\,\psi(y) dy \text{ (stationary kernel)}$$

- Therefore, the Fourier transform can be written as

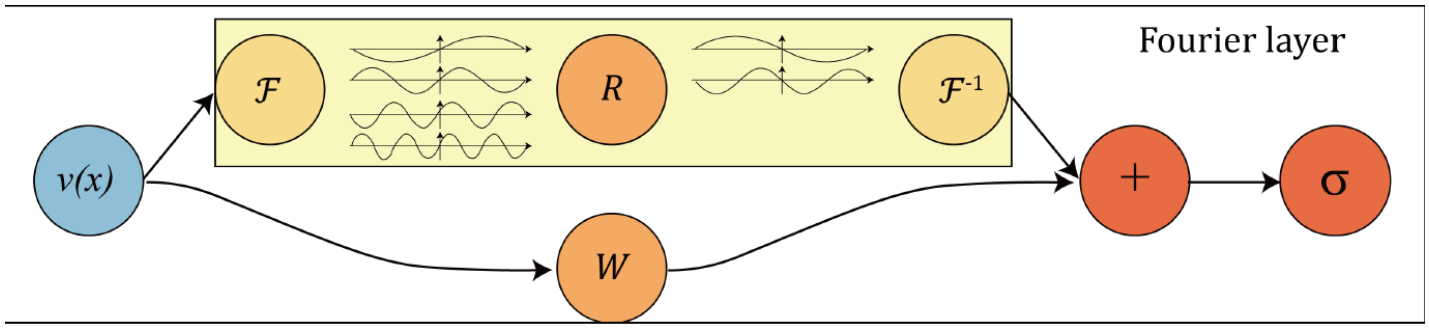
$$F(\omega) = \int_\Omega \left(\int_\Omega k(x-y)\psi(y) dy \right) e^{-i\omega x} dx = \int_\Omega \psi(y) \int_\Omega k(x-y) e^{-i\omega x} dx\, dy$$

- Consider $x - y = z$. Therefore,

$$F(\omega) = \int \psi(y) \int k(z) e^{-i\omega(y+z)} dz dy = \int \psi(y) e^{-i\omega y} dy \int k(z) e^{-i\omega z} dz$$
$$= \Psi(\omega) K(\omega)$$

- In the Fourier domain, the convolution integral results in pointwise multiplication.

Fourier layer

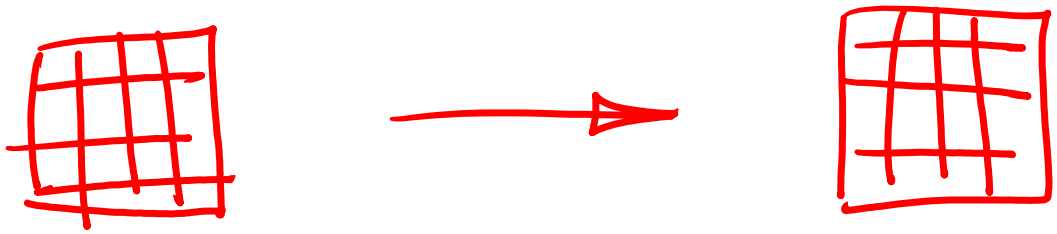


- The Fourier layer just consists of three steps:
  - Fourier transform $\mathcal{F}$
  - Linear transform of lower Fourier modes R
  - Inverse Fourier transform $\mathcal{F}^{-1}$

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

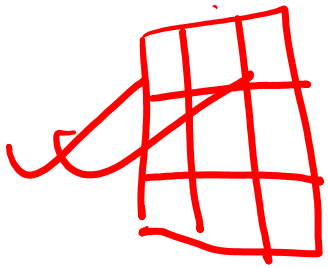
- We then add the output of the Fourier layer with the bias term Wv (a linear transformation) and apply the activation function.

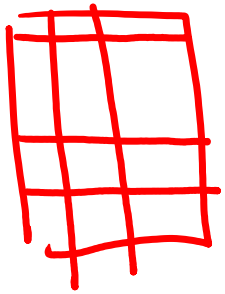
- In practice, it's usually sufficient to only take the lower frequency modes and truncate out these higher frequency modes. Therefore, we apply the linear transformation on the lower frequency modes and set the higher modes to zeros.

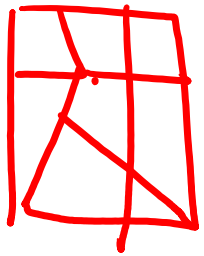


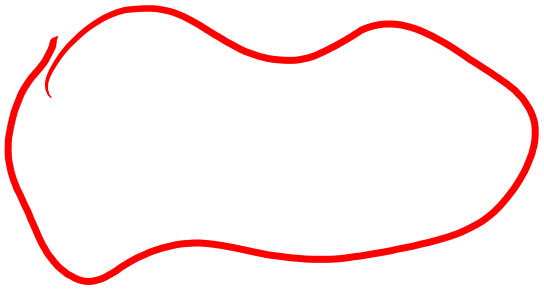
Fourier layer

- Notice the activation functions shall be applied on the spatial domain.
- They help to recover the Higher frequency modes and non-periodic boundary which are left out in the Fourier layers.
- Therefore, it's necessary to the Fourier transform and its inverse at each layer
- The Fourier layer on its own loses higher frequency modes and works only with periodic boundary conditions. However, the Fourier neural operator as a whole does not have these limitations.
- The Fourier layers are discretization-invariant, because they can learn from and evaluate functions which are discretized in an arbitrary way.
- Since parameters are learned directly in Fourier space, resolving the functions in physical space simply amounts to projecting on the basis of wave functions which are well-defined everywhere on the space. This allows us to transfer among discretization.
- If implemented with standard FFT, then it will be restricted to uniform mesh, but still resolution-invariant.









Geo-FNO

U-FNO

Limitation and motivation for WNO

Limitations:

- Fourier transformation only provide frequency information; no information on either space or time is available.
- FNO relies on the $g(x)$ for dealing with non-periodic boundary
- FNO performs poorly on irregular geometry (a case study shown later).

Solution: This can be addressed by learning the kernel integral in the wavelet space.

This is the motivation behind **Wavelet Neural Operator**.

Similar to FNO, the convolution integral in real space reduces to pointwise multiplication in the wavelet space.

We exploit wavelet transform and inverse wavelet transform (instead of Fourier transform and inverse Fourier transform) to construct the wavelet neural operator.

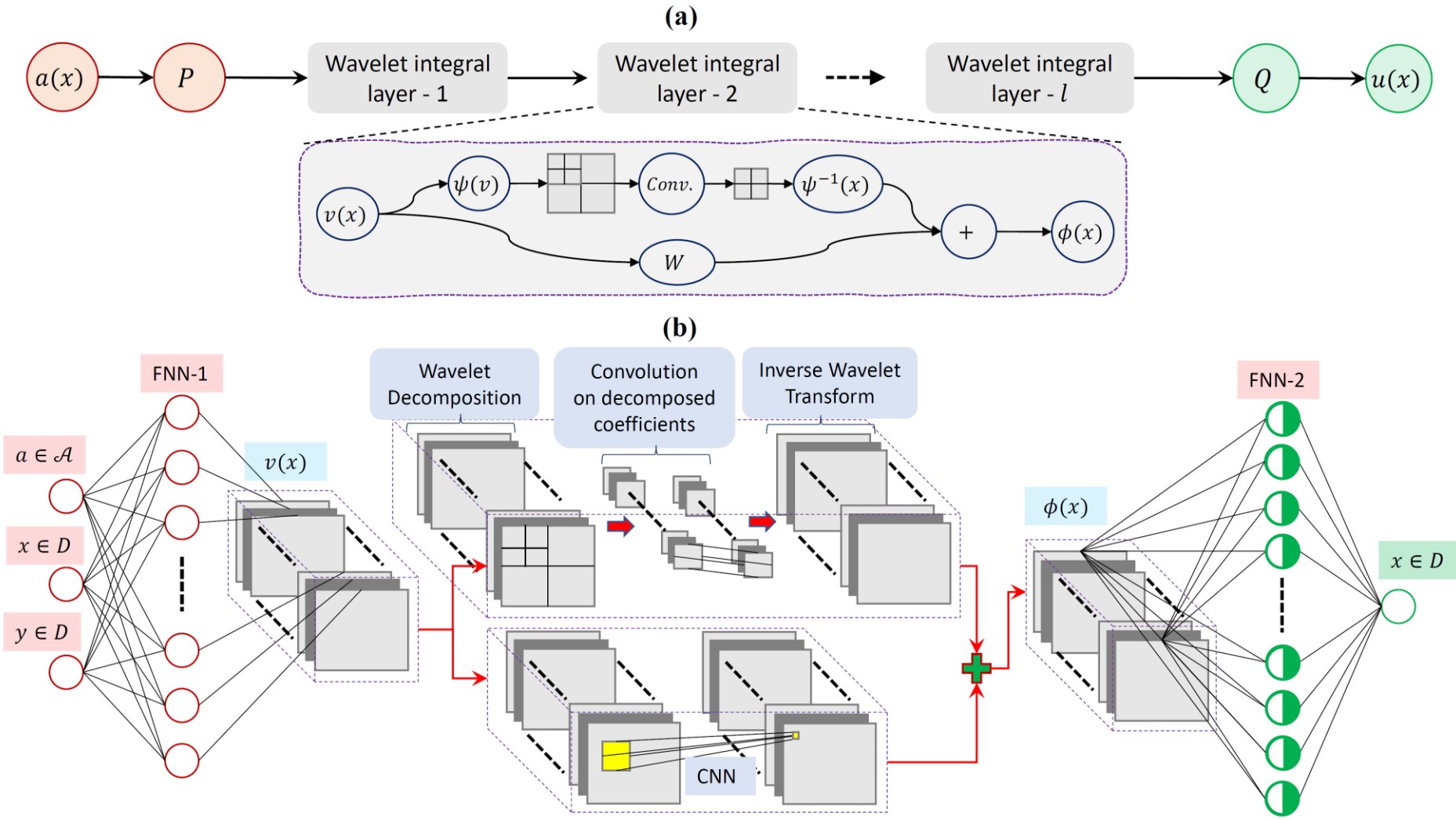
Note, Fourier transform is a special case of wavelet transform.

Wavelet transform

$$(\mathcal{W}\,\Gamma)_j(\alpha,\beta) = \int_D \Gamma(x)\frac{1}{|\alpha|^{0.5}}\psi\left(\frac{x-\beta}{\alpha}\right)dx$$

- $\psi(\cdot)$ is referred to as the mother wavelet.
- The desired wavelets can be obtained from mother wavelet by scaling and shifting.
- Similar to Fourier transform, there can be continuous wavelet transform and discrete wavelet transform.
- In the DWT, the coefficients in smaller scales are usually associated with high frequencies and the wavelet coefficients generally correspond to low-frequency information.
- There are different families of wavelet basis. A comprehensive list of the same can be found here.

WNO


(a)
$a(x)$
P
Wavelet integral layer - 1
Wavelet integral layer - 2
Wavelet integral layer - l
Q
$u(x)$
$v(x)$
$\psi(v)$
$Conv.$
$\psi^{-1}(x)$
W
$+$
$\phi(x)$
(b)
FNN-1
$a \in \mathcal{A}$
$x \in D$
$y \in D$
$v(x)$
Wavelet Decomposition
Convolution on decomposed coefficients
Inverse Wavelet Transform
CNN
$\phi(x)$
FNN-2
$x \in D$

Algorithm

Algorithm 1 Algorithm of the WNO

Input: N-samples of the pair $\{a(x) \in \mathbb{R}^{n_D \times d_a}, u(x) \in \mathbb{R}^{n_D \times d_u}\}$, coordinates $x \in D$, and network hyperparameters.

1: Stack the inputs: $\{a(x), x\} \in \mathbb{R}^{n_D \times 2d_a}$.
2: **for** epoch $= 1, \ldots,$ epochs **do**
3: Uplift the input using transformation $P(\cdot)$: $v_0(x) \in \mathbb{R}^{n_D \times d_v} = P\left(\{a(x), x\} \in \mathbb{R}^{n_D \times 2d_a}\right)$.
4: **for** $j = 1, \ldots, l$ perform the iterations: $v_{j+1}{=}G(v_j)$ **do**
5: Decompose the input using wavelet decomposition: $\mathcal{W}(v_j(x)) \in \mathbb{R}^{n_D/2^m \times d_v}$.
6: Parameterize the NN kernel k_ϕ in the wavelet space: $R_\phi * \mathcal{W}(v_j(x))$. ▷ Eq. (14)
7: Reconstruct the convoluted input: $v^1_{j+1}(x) = \mathcal{W}^{-1}\left(R_\phi * \mathcal{W}(v_j(x))\right)$. ▷ Eq. (10)
8: Perform the linear transform: $v^2_{j+1}(x) = W * v_j(x)$ using CNN.
9: Add the outputs of step 7 and 8: $\tilde{v}_{j+1}(x) \in \mathbb{R}^{n_D \times d_v} = \left(v^1_{j+1} + v^2_{j+1}\right)(x)$.
10: **if** $j \neq l$ **then**
11: Apply the activation to complete the iteration: $v_{j+1} \in \mathbb{R}^{n_D \times d_v} = g\left(\tilde{v}_{j+1}(x)\right)$. ▷ Eq. (2)
12: **end if**
13: **end for**
14: Apply an activated lifting transformation: $r(x) \in \mathbb{R}^{n_D \times d_r}{=}g\left(Q_1(v_l(x))\right)$ ($Q_1(\cdot) : \mathbb{R}^{d_v} \mapsto \mathbb{R}^{d_r}$ is a FNN).
15: Compute the final output: $\hat{u}(x) \in \mathbb{R}^{n_D \times d_u}{=}Q_2\left(r(x)\right)$ ($Q_2(\cdot) : \mathbb{R}^{d_r} \mapsto \mathbb{R}^{d_u}$ is a FNN).
16: Compute the loss: $\mathcal{L}\left(u, \hat{u}\right)$.
17: Compute the gradient of the loss: $\frac{\partial \mathcal{L}(u,\hat{u})}{\partial \theta_{NN}}$.
18: Update the parameters of the network using the gradient.
19: **end for**

Output: Solution space $u \in \mathcal{U}$, parameters of NN θ_{NN}.

FNO vs. WNO vs. DeepONet

Examples	Number of data		Wavelet	m	Network dimensions				$g(.)$
	Training	Testing			FNN1	FNN2	CNN	WNO	
Burgers(4.1)	1000	100	db6	8	64	128	4	4	GeLU
Darcy (rectangular)(4.2)	1000	100	db4	4	64	128	4	4	GeLU
Darcy (triangular)(4.3)	1900	100	db6	3	64	128	4	4	GeLU
Navier-Stokes(4.4)	1000	100	db4	3	26	128	4	4	GeLU
Allen-Cahn(4.5)	1400	100	db4	1	64	128	4	4	GeLU
Advection(4.6)	900	100	db6	3	96	128	4	4	GeLU

	Number of epochs					
Architectures	Burgers'	Darcy flow	Navier-Stokes	Allen-Cahn	Darcy (Notch)	Advection
DeepONet	500000	100000	100000	100000	20000	250000
POD-DeepONet	500000	100000	100000	100000	20000	250000
FNO	500	500	500	500	500	500
MWT	500	500	500	500	500	500
WNO	500	500	500	500	500	500

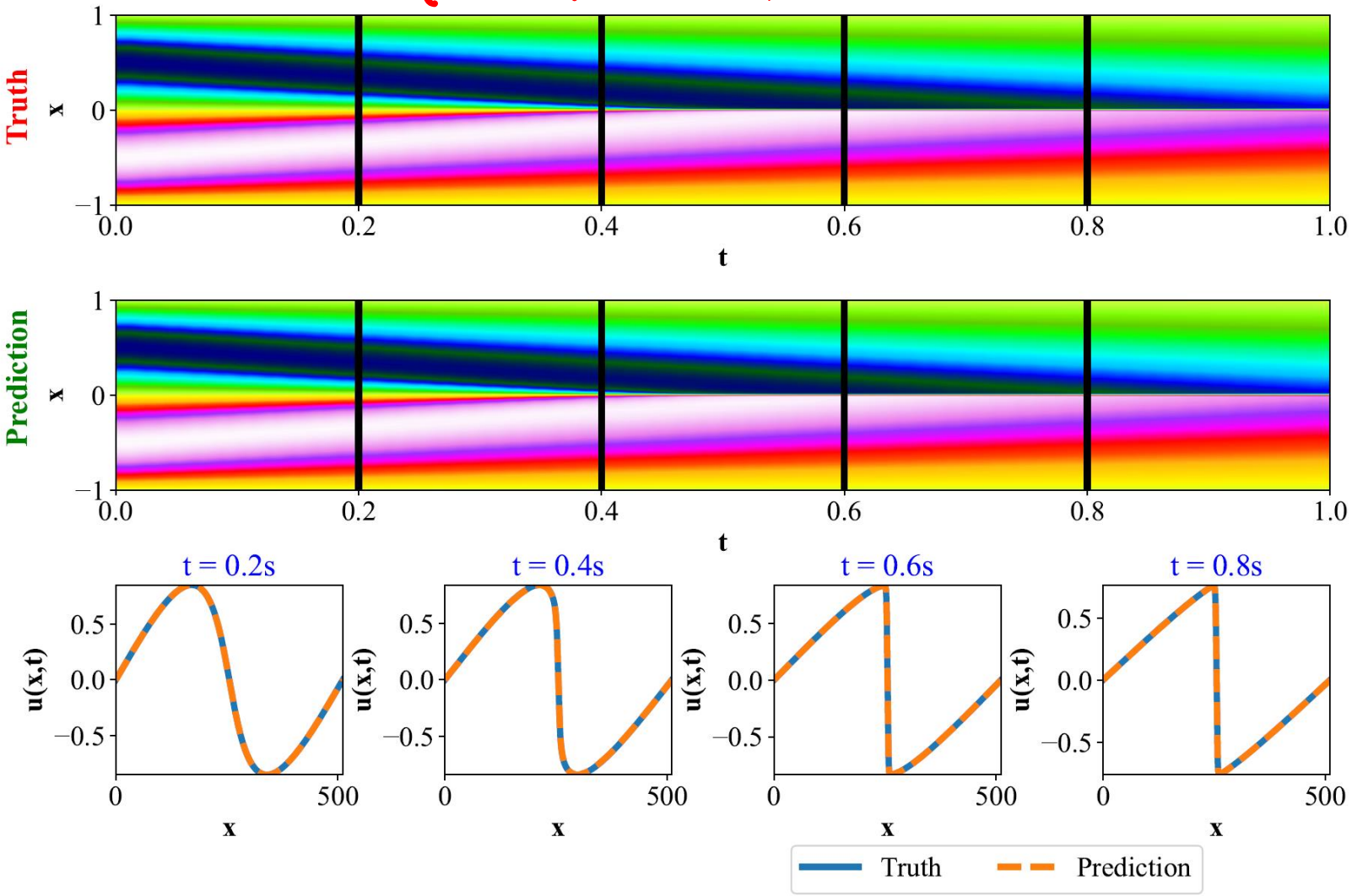
FNO vs. WNO vs. DeepONet

PDE	Network Architectures						
	GNO	DeepONet	FNO	MWT	POD-DeepONet‡	dgFNO+†	WNO
Burgers' equation	≈ 6.15 %	≈ 2.15 %	≈ 1.60 %	≈ 0.19 %	≈ 1.94 %	-	≈ 1.75 %
Darcy (rectangular)	≈ 3.46 %	≈ 2.98 %	≈ 1.08 %	≈ 0.89 %	≈ 2.38%	-	≈ 0.84 %
Darcy (triangular)	-	≈ 2.64 %	-	≈ 0.87 %	≈ 1.00 %	≈ 7.82%	≈ 0.77 %
Navier-Stokes	-	≈ 1.78 %	≈ 1.28 %	≈ 0.63 %	≈ 1.36 %	-	≈ 0.31 %
Allen-Cahn	-	≈ 17.7 %	≈ 0.93 %	≈ 4.84 %	-	-	≈ 0.21 %
Wave-advection	-	≈ 0.32 %	≈ 47.7 %	≈ 10.22 %	≈ 0.40 %	≈ 0.60 %	≈ 0.62 %

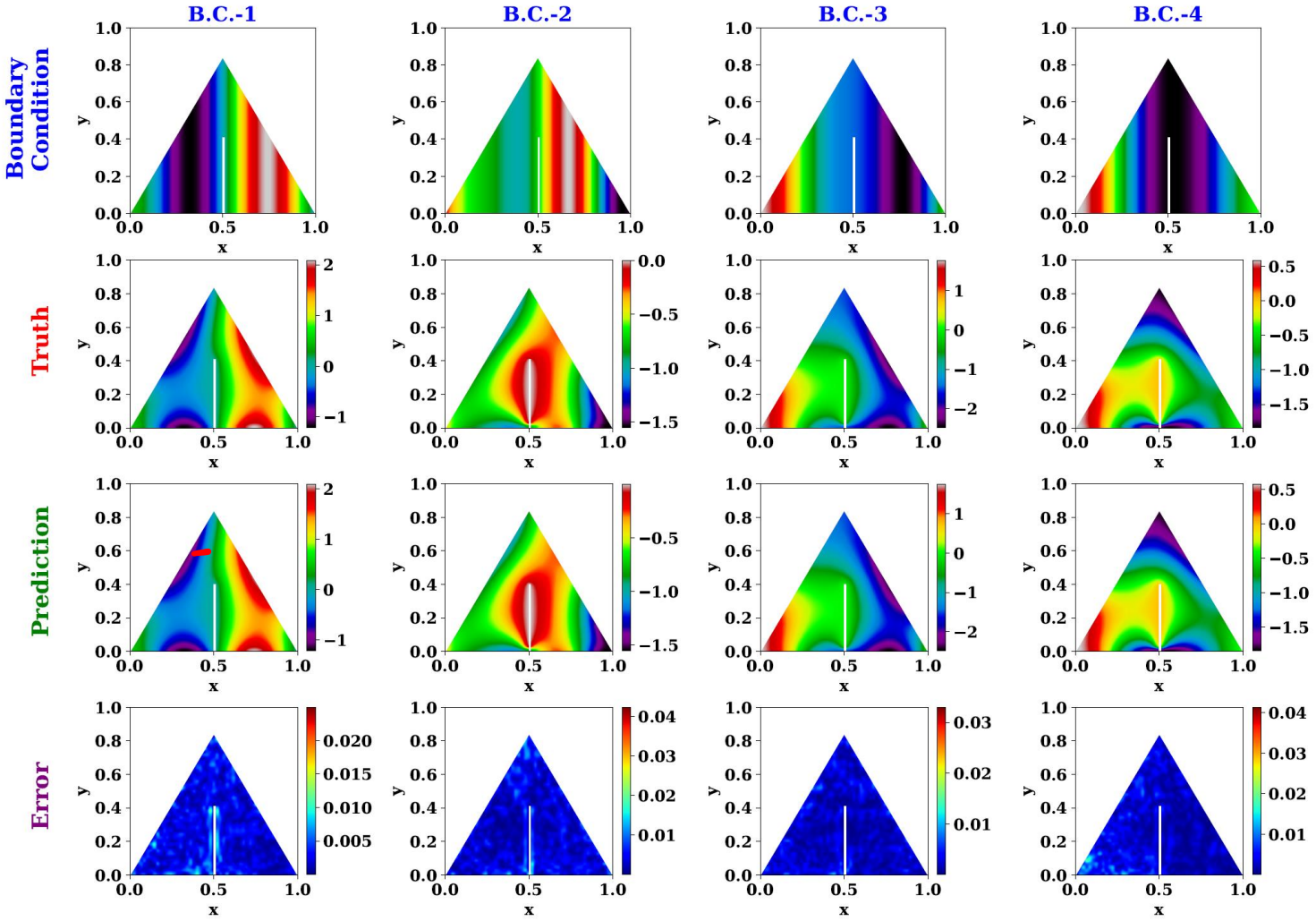
‡ POD-DeepONet is the modified version of DeepONet, proposed in Ref. [26]. † Note that in complex geometric conditions, FNO does not work. Thus we use dgFNO+, which is a modified version of FNO (Ref. [26]).

Burger's equation

$u_t + uu_x = \nu u_{xx}$

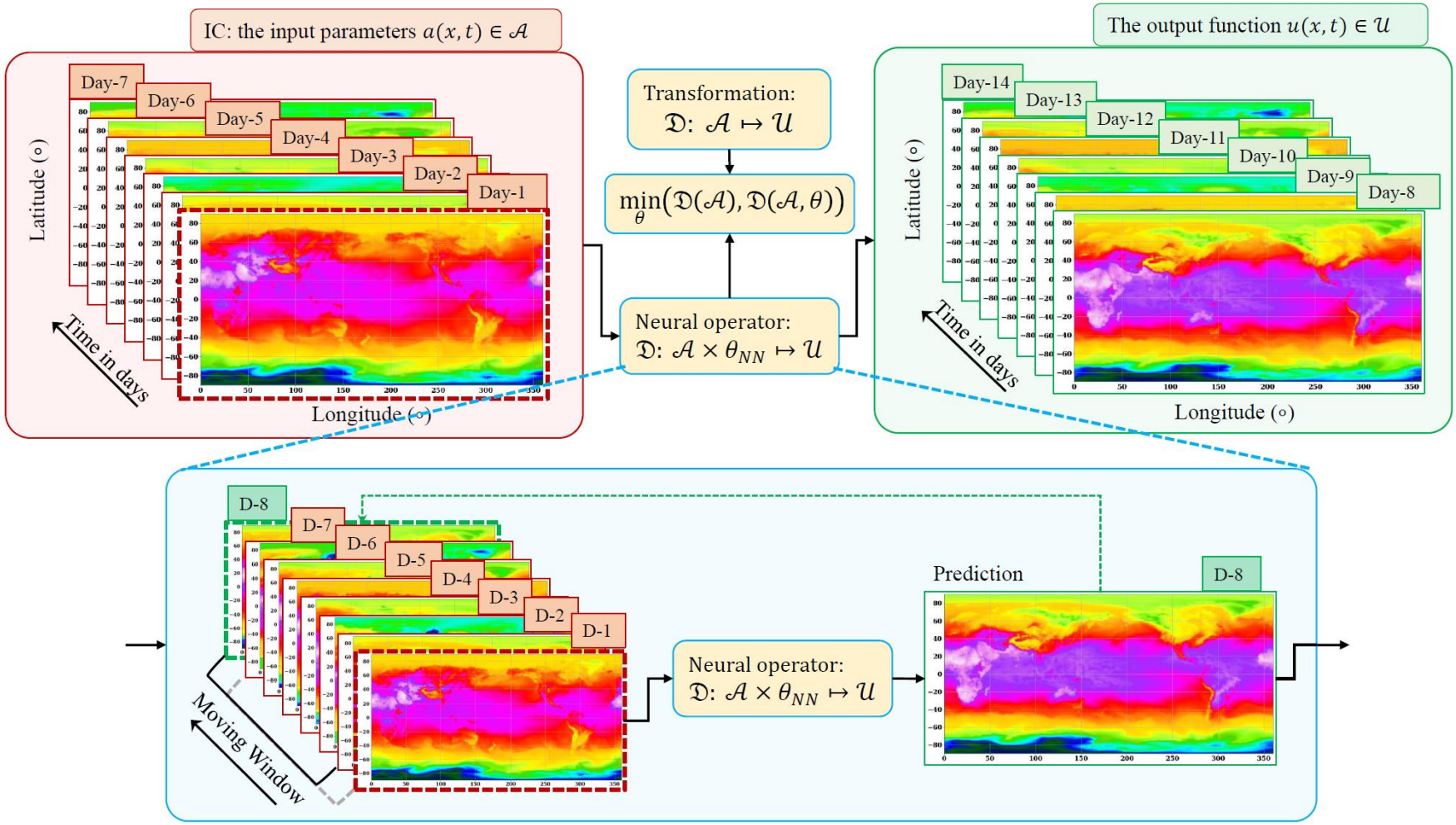


Darcy Flow

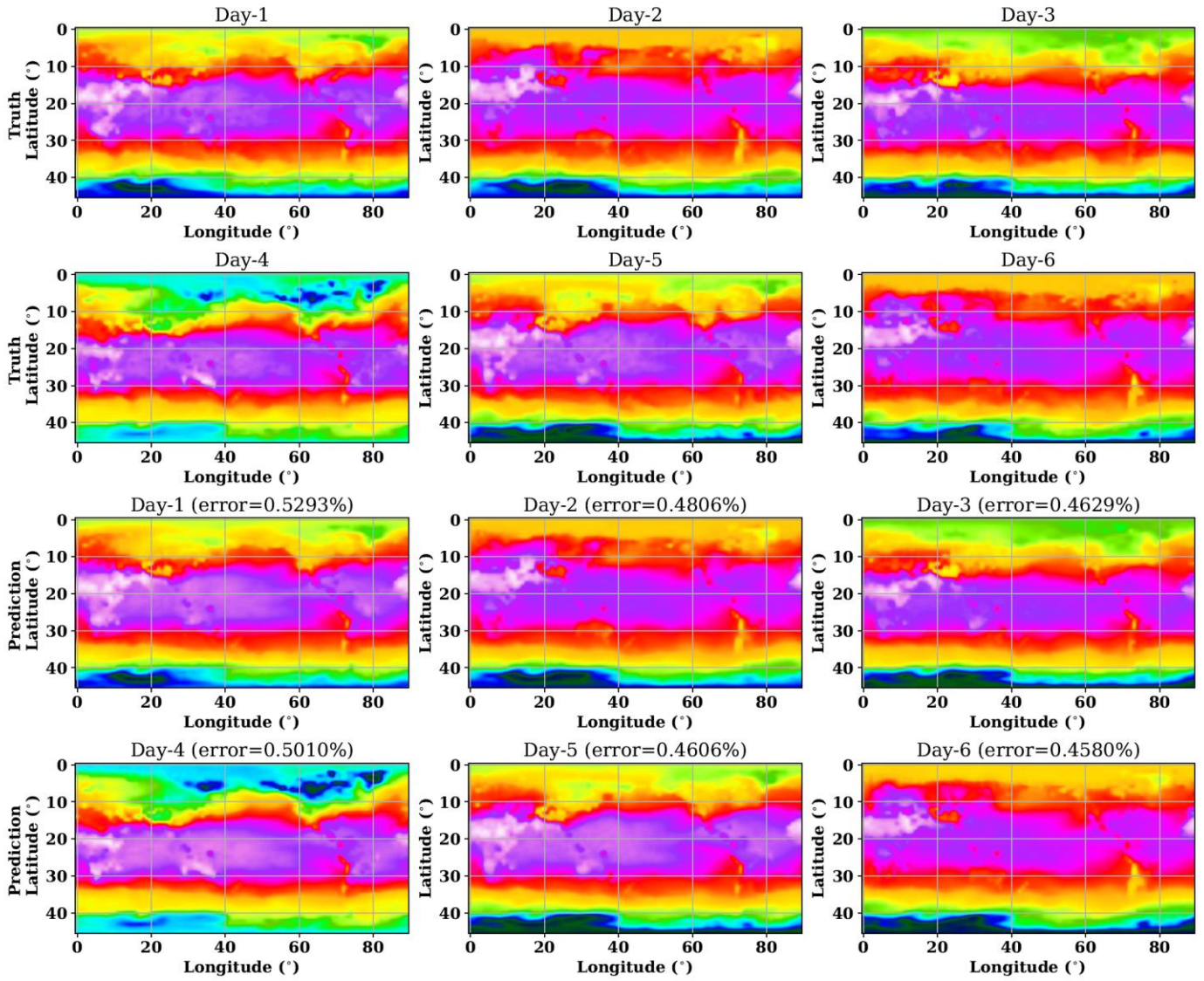

B.C.-1
B.C.-2
B.C.-3
B.C.-4
Boundary Condition
Truth
Prediction
Error
x
y

Digital twin of Earth's temperature

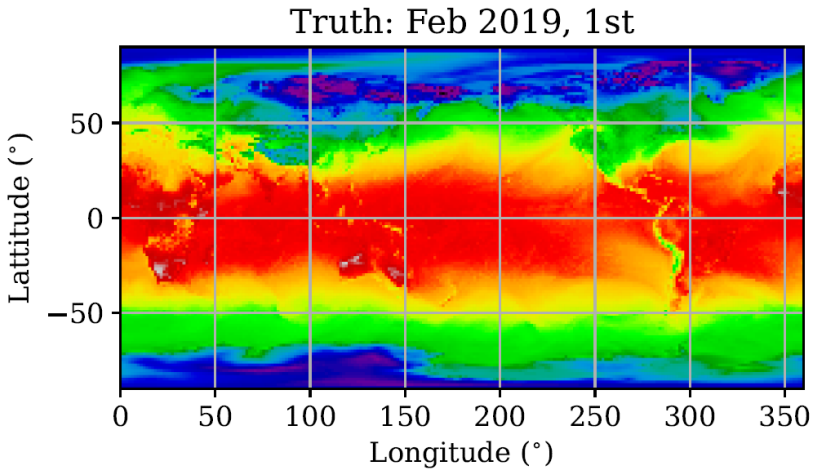
Framework

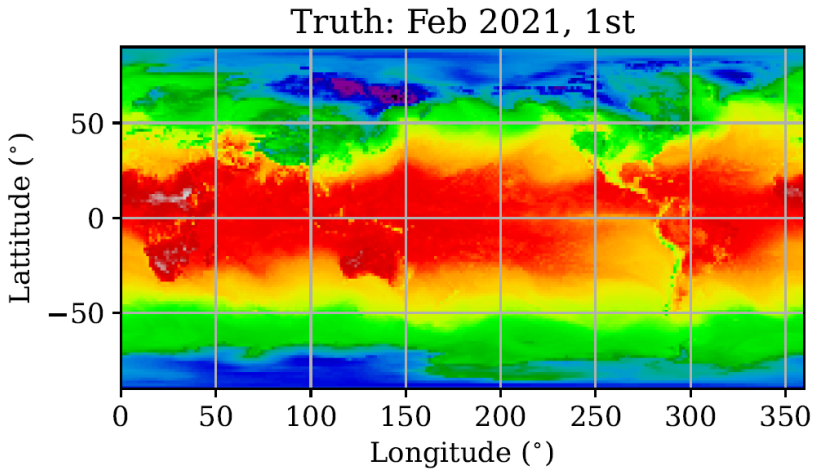


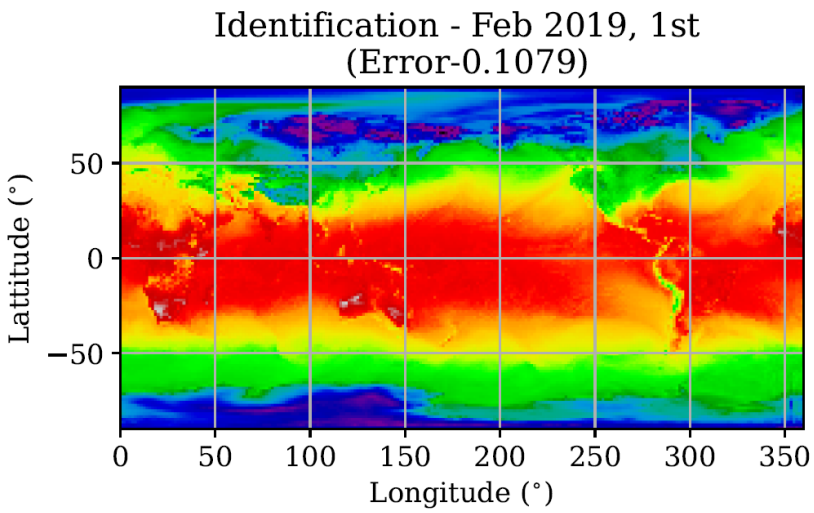
Prediction (Daily prediction)

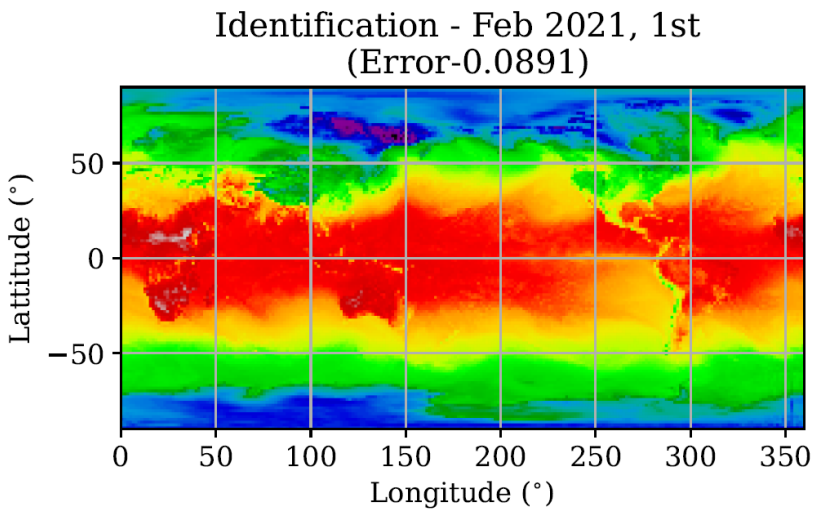


Prediction (Monthly average)



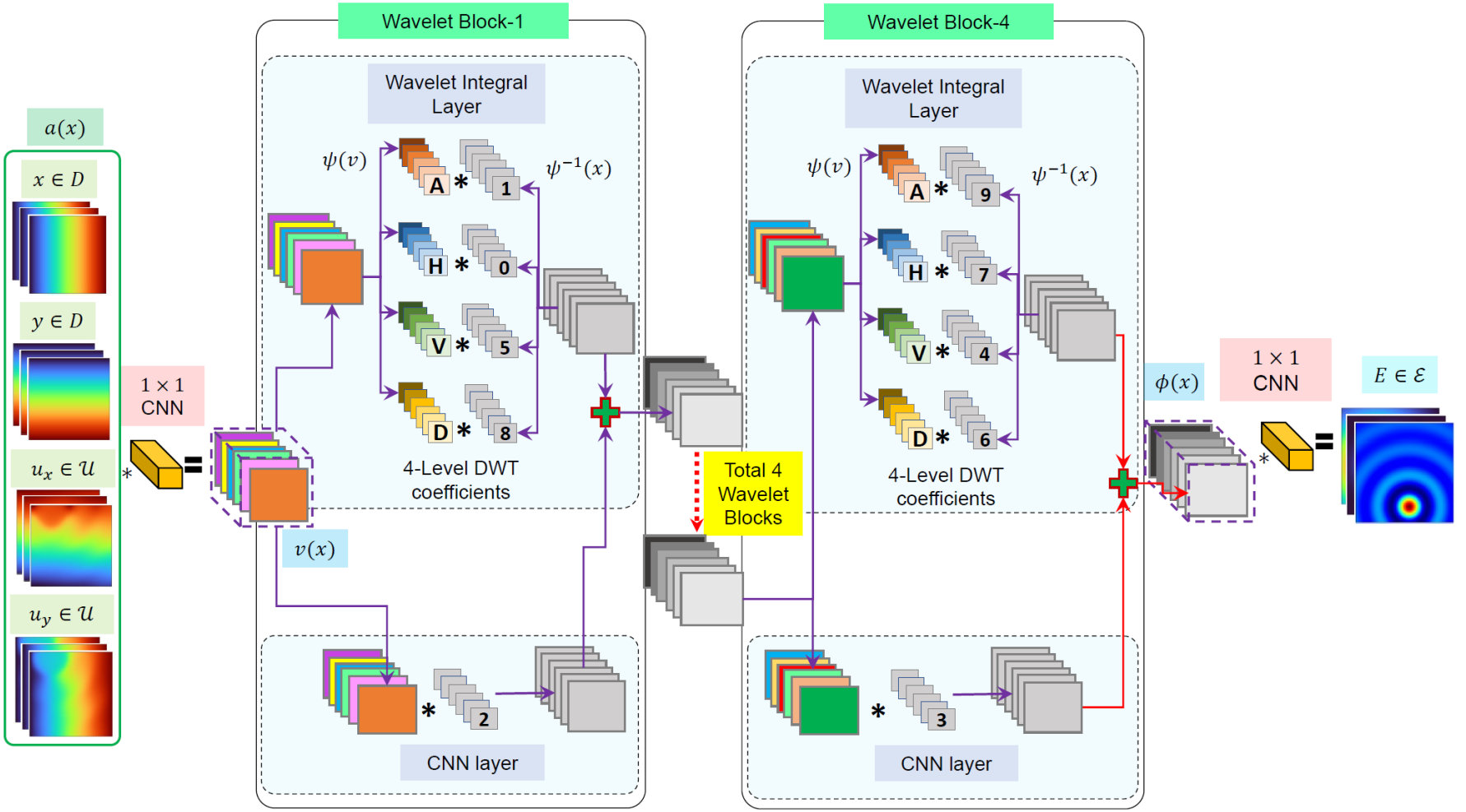




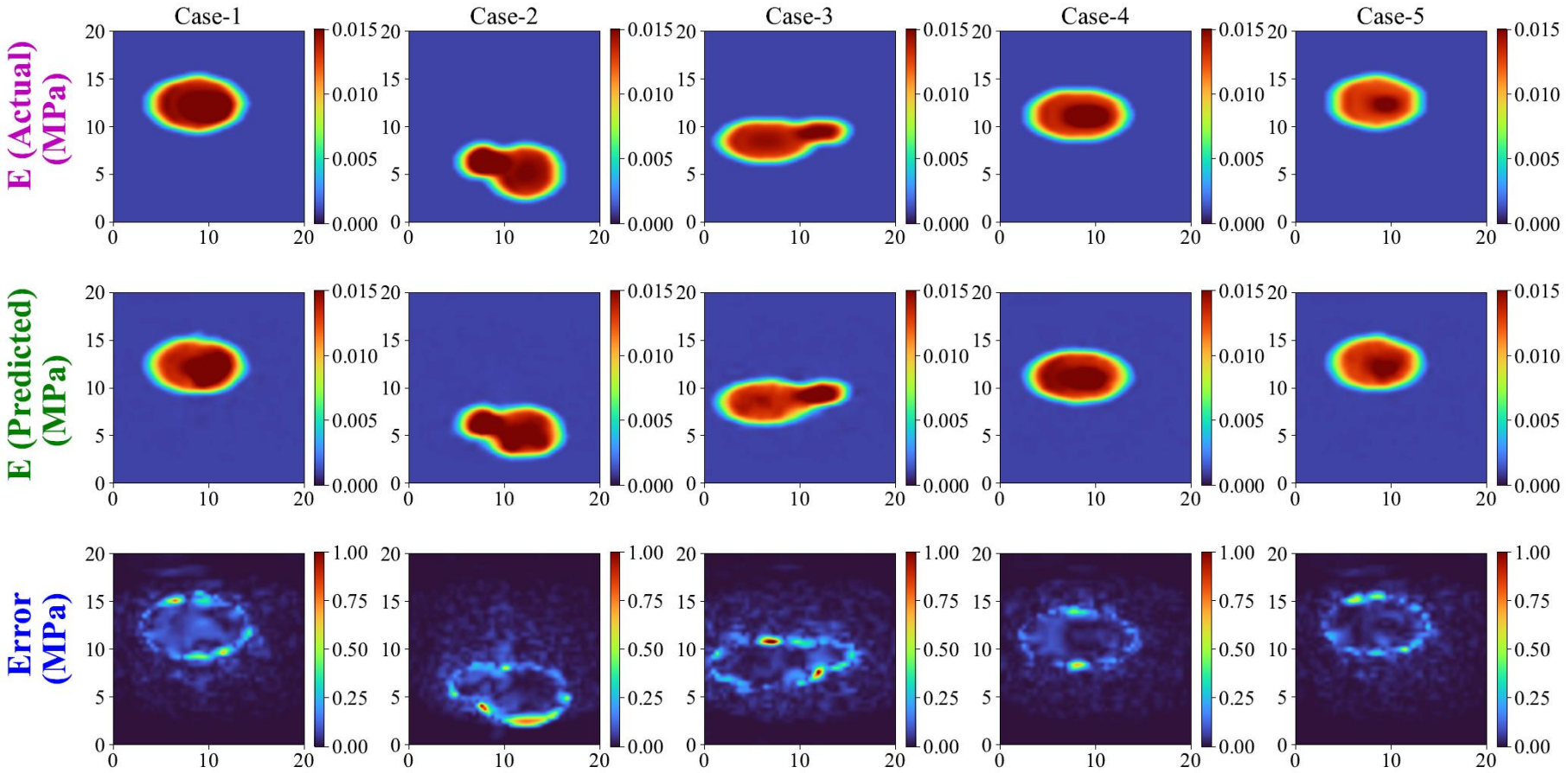


Tumor detection from USG data

Framework

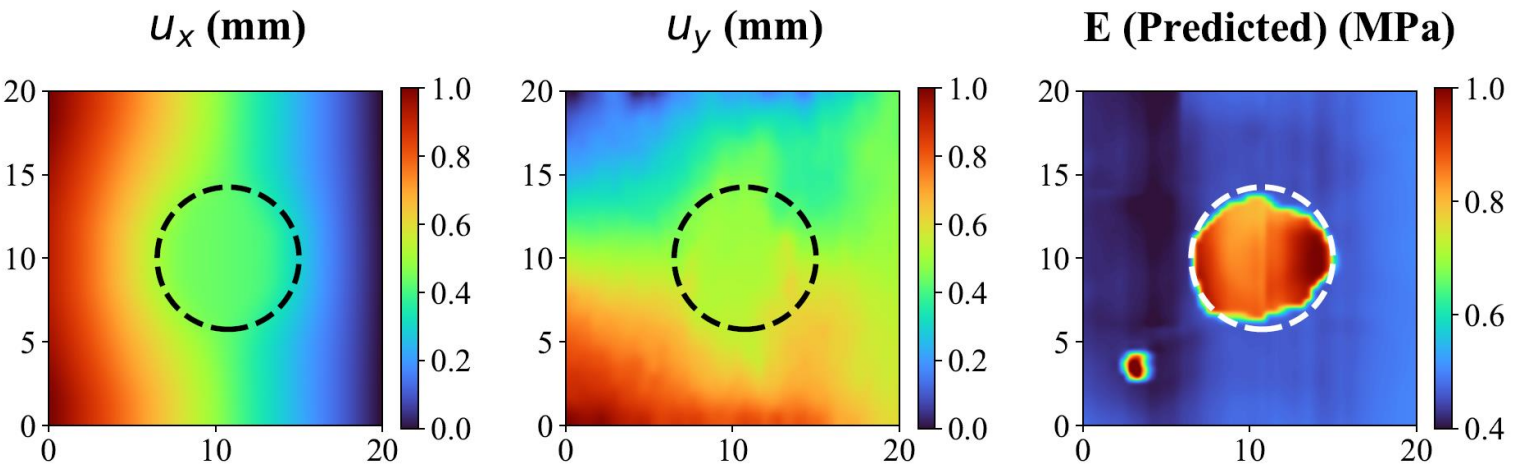


Tumor detection from USG data



Tumor detection from USG data

Noise (in % of the stdev.)	SNR (dB)	L^2 relative error (%)	
		WNO	CNN encoder-decoder
1	68	0.4199	2.1210
2	62	0.4212	2.1249
5	54	0.5166	2.1735
10	48	0.5628	2.2251



Limitations of WNO

- All neural operators are kind of data hungry.
- Although the network architecture for FNO and WNO are, in some sense, inspired from physics, explicit information on the physics is not included.
- With discrete wavelet transform, we gain efficiency but loose out on accuracy.
- WNO is deterministic – no information on epistemic uncertainty due to limited and noisy data

Solutions

- Multi-fidelity WNO.
- PI-WNO
- C-WNO
- RP-WNO

$$D_L = \{x_i, Y_{Li}\}_{i=1}^{N_L}$$

$$D_H = \{x_j, Y_{Hj}\}_{j=1}^{N_H}$$

Basic assumption :-

Correlated.

$$Y_H(x) = \rho(x) Y_L + \varepsilon(x) \leftarrow \text{Linear}$$

$$Y_H(x) = G(x, Y_L) + \varepsilon(x)$$

$$= Y_L + G_1(x, Y_L) + \varepsilon(x)$$

MF-WNO

- In multi-fidelity methods, we have data from two sources – one more accurate (high fidelity) and one less accurate (low fidelity).

- The major theme in multi-fidelity learning is the exploitation of correlation between the HF data, which is quite accurate but available in a smaller amount, and the LF data, which is inaccurate but is available in a larger amount

- A popular strategy in this regards is

$$y_H = \rho(x) y_L + \delta(x)$$

However, this only account for linear correlation between LF and HF

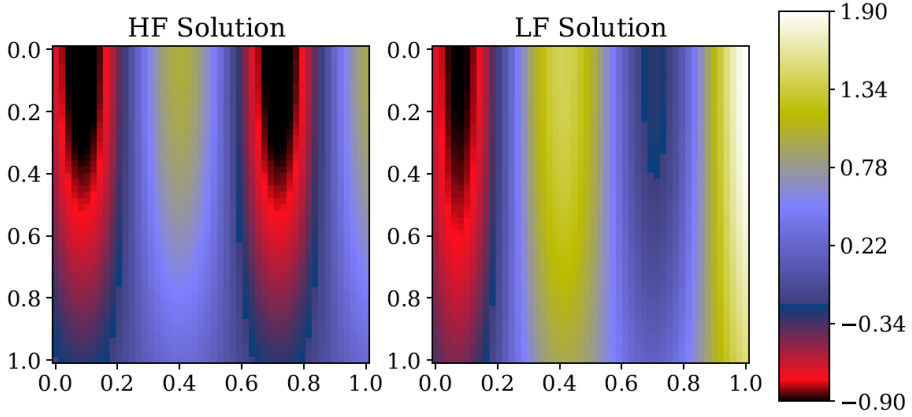
- A generic scheme can take the following form

$$y_H = \mathcal{G}(y_L, x) + \delta(x)$$

MF-WNO example – 2 dimensional problem with correlation

$$D_L(a)(x,y) = \cos(a)\cos(y) + x$$

$$D_H(a)(x,y) = \cos(a)\,[\cos(y)]^2 \qquad a = kx - 4$$



$\mathcal{Q}_{train}$ Size	MSE	
	MFSM	HFSM
10	1.2043×10^{-8}	1.7359×10^{-5}
8	1.3134×10^{-7}	7.7837×10^{-5}
6	3.5792×10^{-7}	3.3641×10^{-4}
2	3.3927×10^{-5}	2.3504×10^{-3}
MSE_{LF}	3.2892×10^{-1}	

MF-WNO example

Equation

$$-\nabla \cdot \big(a(x,y)\nabla u(x,y)\big) = 0, \qquad x, y \in (0, \mathbb{R})$$

Boundary conditions

$$u = 0, \qquad \forall x = 1$$
$$u = 1, \qquad \forall x = 0$$
$$\frac{\partial u}{\partial n} = 0, \qquad \forall y = 0 \text{ and } y = 1$$

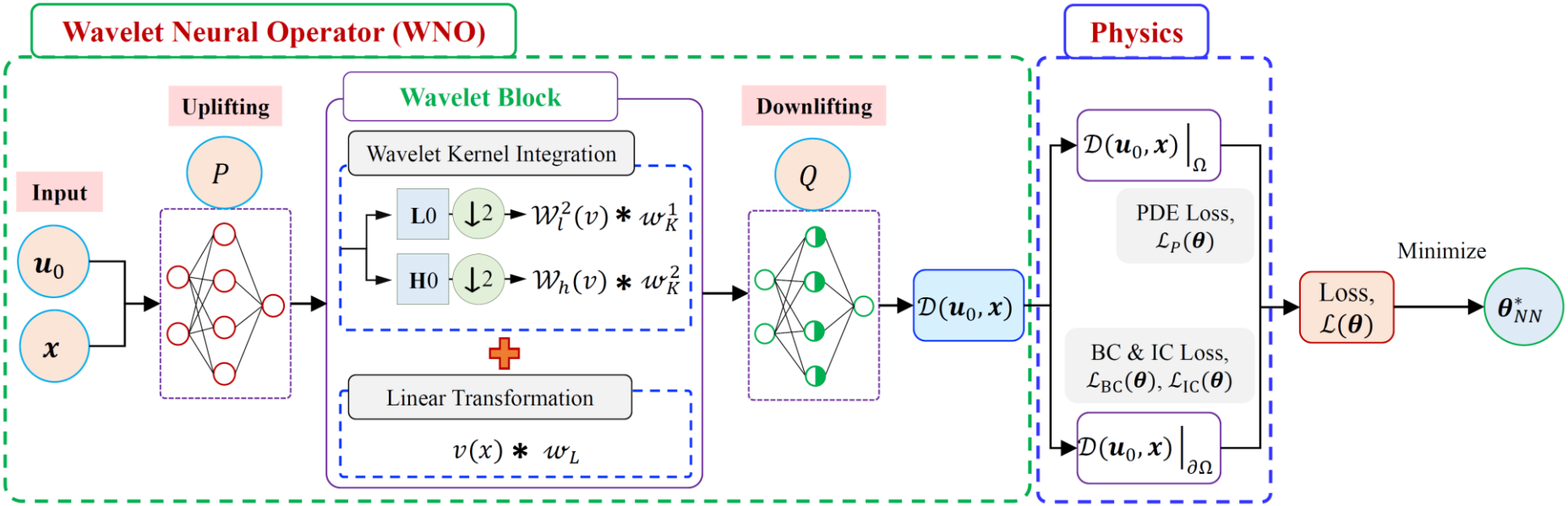
Input modelling

$$\log a(x,y) \sim GP\left(a(x,y) | m(x,y), \mathcal{K}\big((x,y),(x',y')\big)\right)$$

Results

$\mathcal{Q}_{train}$ Size	MSE	
	MFSM	HFSM
25	1.4001×10^{-4}	3.0188×10^{-3}
15	2.7293×10^{-4}	5.5890×10^{-3}
8	4.5502×10^{-4}	1.3692×10^{-2}
5	7.2992×10^{-4}	2.5784×10^{-2}
MSE_{LF}	1.3793×10^{-3}	

PI-WNO



Challenge:

- Computing gradient

Solution:

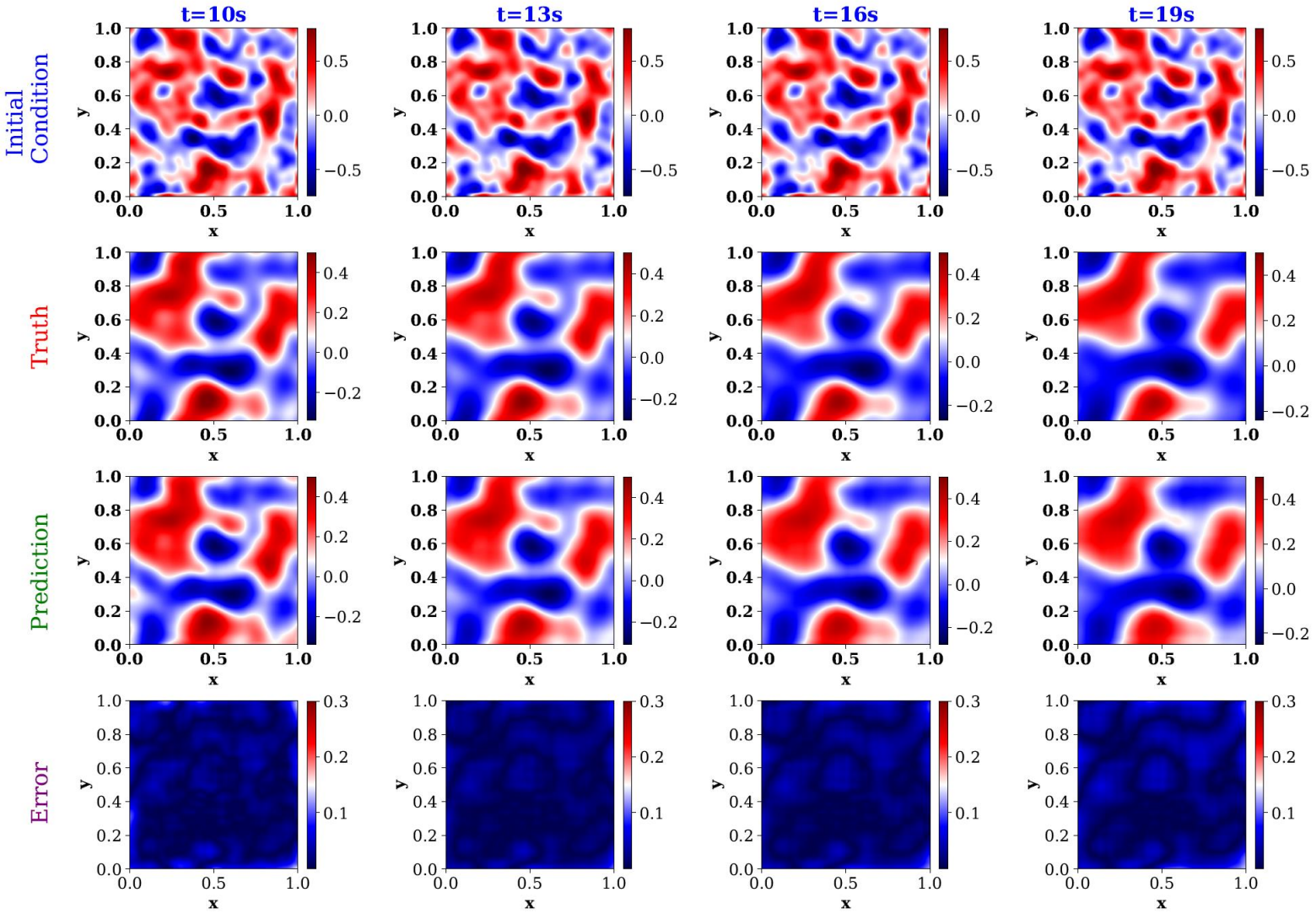
- Stochastic projection

$$\hat{G}(x = \bar{x}) = \frac{\partial u(\bar{x}, \omega)}{\partial x} = \frac{\frac{1}{N_t}\sum_{i=1}^{N_b}\big(u(x_i, \omega) - u(\bar{x}, \omega)\big)(x_i - \bar{x})^T}{\frac{1}{N_t}\sum_{i=1}^{N_b}(x_i - \bar{x})(x_i - \bar{x})^T}$$

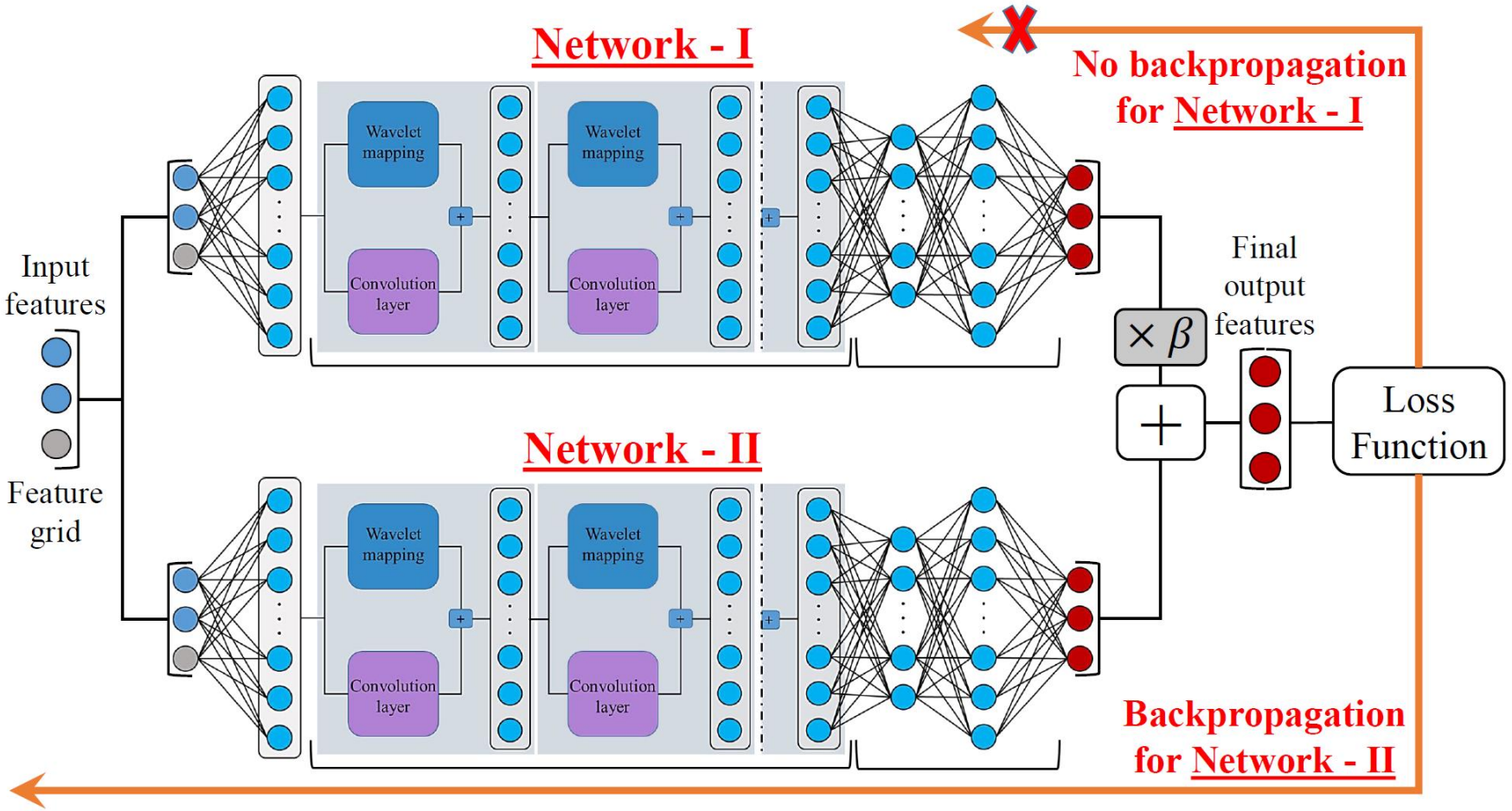
PI-WNO example

	PDEs	WNO		
		PIWNO	**Data-driven**	**Data + physics**
Relative MSE	Burger's	$0.0518 \pm 0.041\%$	$0.1552 \pm 0.17\%$	$0.0362 \pm 0.028\%$
$\mathbf{N_s}$		–	300	300
Relative MSE	Nagumo	$0.0778 \pm 0.088\%$	$0.0254 \pm 0.023\%$	$0.0202 \pm 0.020\%$
$\mathbf{N_s}$		–	800	800
Relative MSE	Non-homogeneous	$0.0469 \pm 0.015\%$	$0.0549 \pm 0.020\%$	0.0371 ± 0.020
$\mathbf{N_s}$	Poisson's	–	500	500
Reltive MSE	Allen-Cahn	$2.73 \pm 0.98\%$	$2.98 \pm 0.83\%$	2.37 ± 0.70
$\mathbf{N_s}$		–	600	600

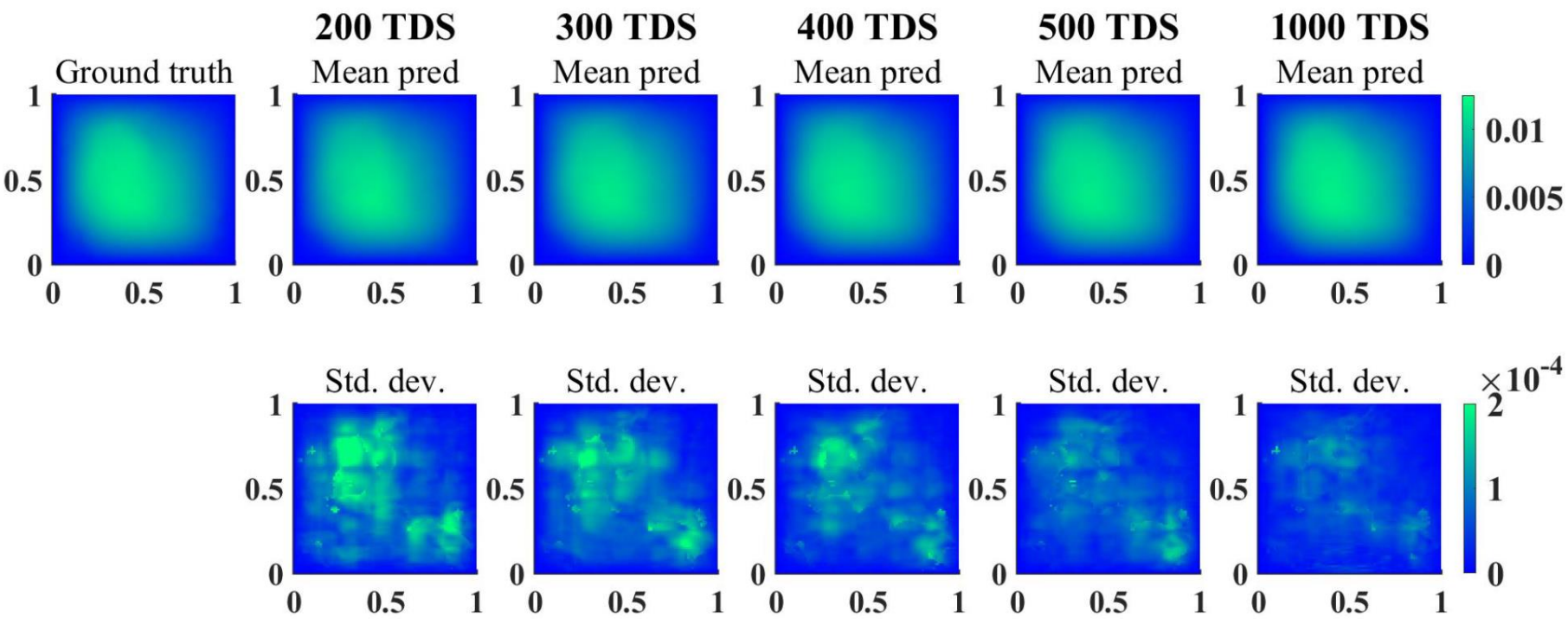
PI-WNO example



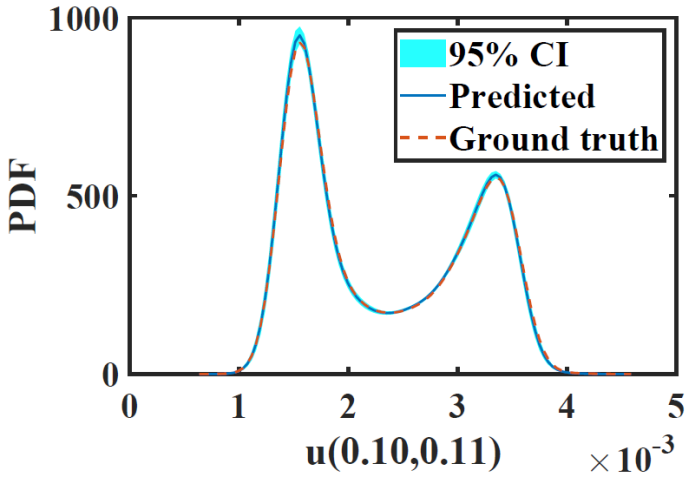
RP-WNO


Network - I
No backpropagation for Network - I
Wavelet mapping
Convolution layer
Wavelet mapping
Convolution layer
Input features
Feature grid
Final output features
× β
+
Loss Function
Network - II
Wavelet mapping
Convolution layer
Wavelet mapping
Convolution layer
Backpropagation for Network - II

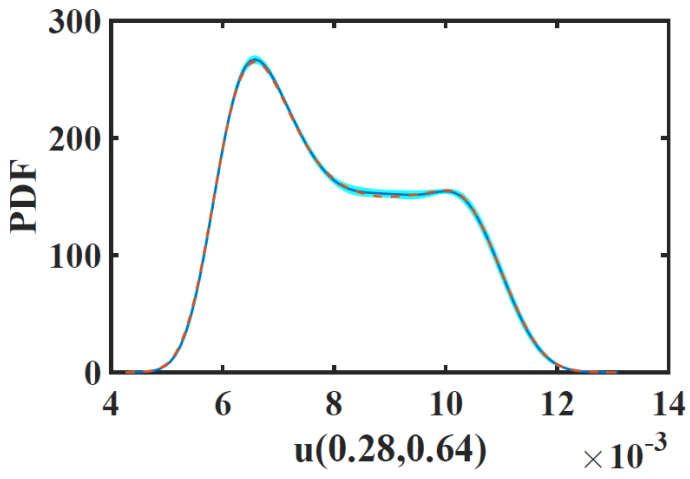
RP-WNO example: Darcy flow



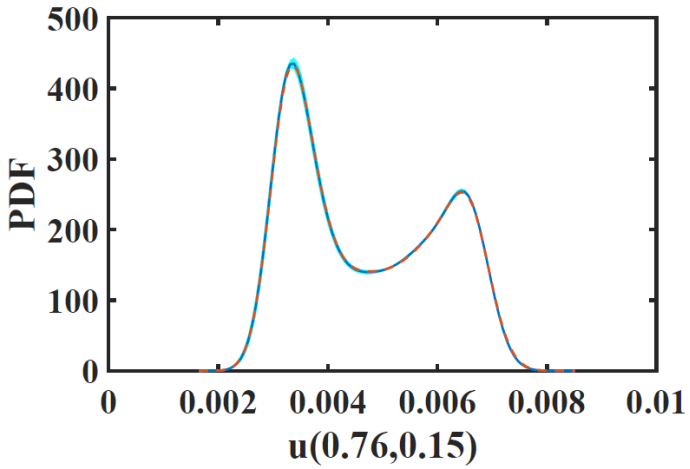
RP-WNO example: Darcy flow



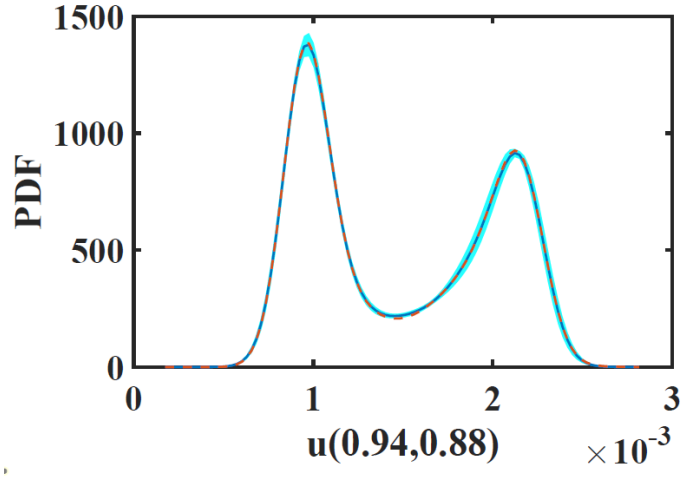
(a) PDF computed at $x = 0.10$ and $y = 0.11$.



(b) PDF computed at $x = 0.28$ and $y = 0.64$.

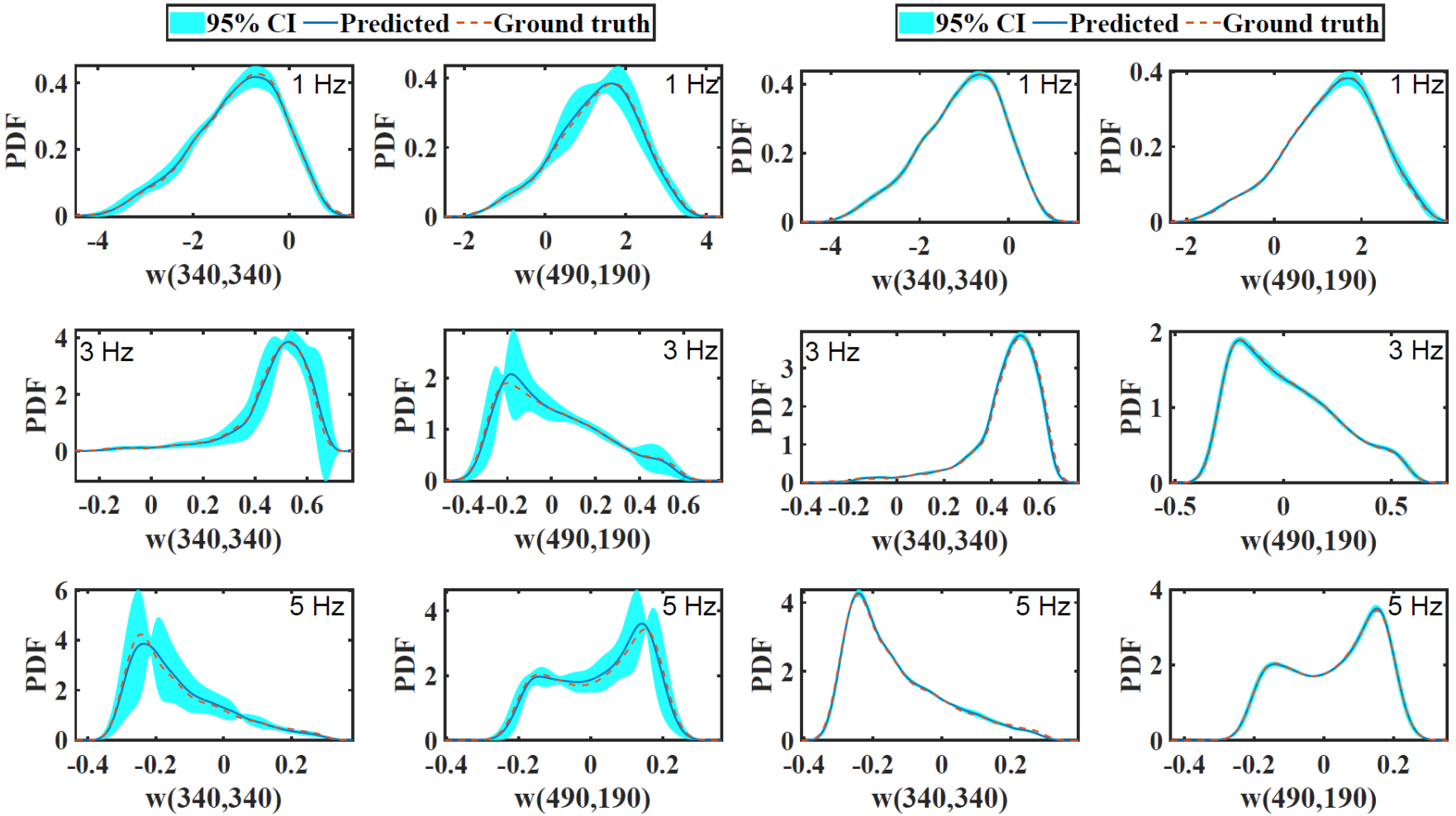


(c) PDF computed at $x = 0.76$ and $y = 0.15$.



(d) PDF computed at $x = 0.94$ and $y = 0.88$.

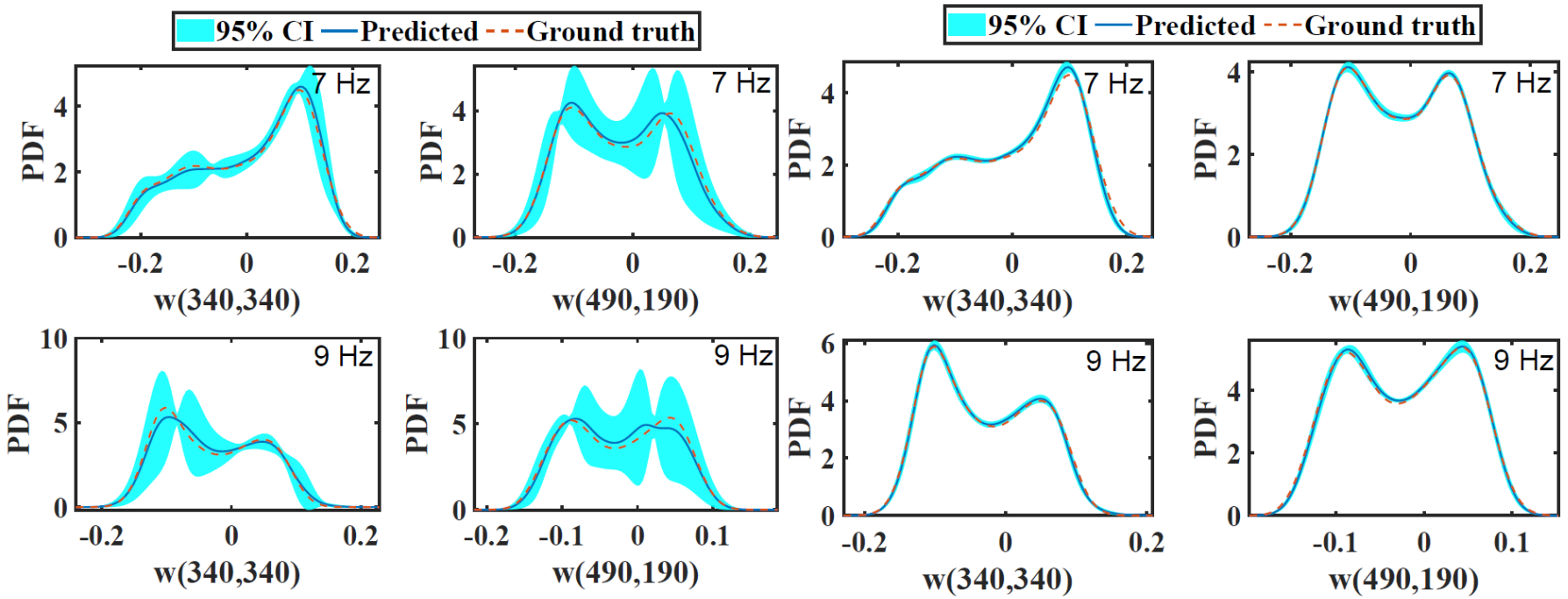
RP-WNO example: Helmholtz equation



(a) Discrete wavelet transform

(b) Continuous wavelet transform

RP-WNO example: Helmholtz equation



(a) Discrete wavelet transform

(b) Continuous wavelet transform