Caltech EAS Division of Engineering and Applied Science

Deep Operator Networks (DeepONets)

Caltech-EAS AI Bootcamp

AI BOOTCAMP

[1] Image credit: “Neural Network 3D Simulation” - Denis Dmitriev (YouTube)

Operators

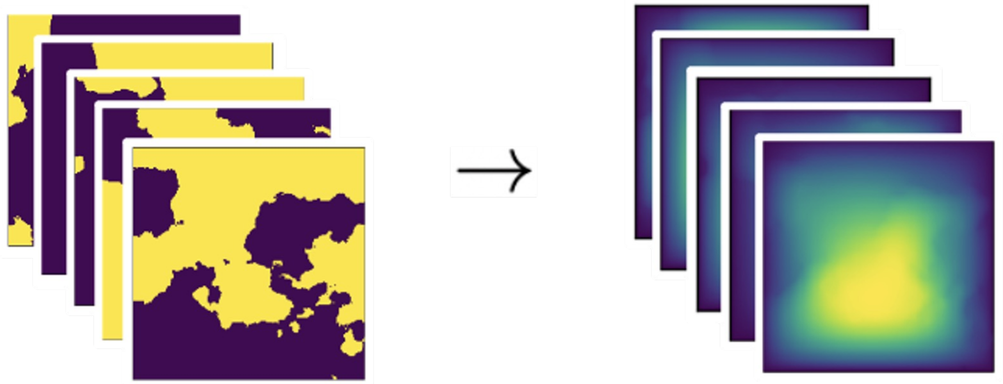
Operators are maps between function spaces.

Examples

- Arithmetic Operators: $+$ $-$ $*$ $/$
- Differential Operators: d/dx ∂_x ∇
- Integral Operators: $\int$ $\sum$

Operator Learning in the context of solving PDEs

Learn the solution operator from input functions to output solutions



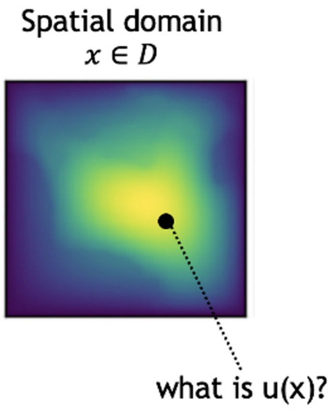
Input: coefficients

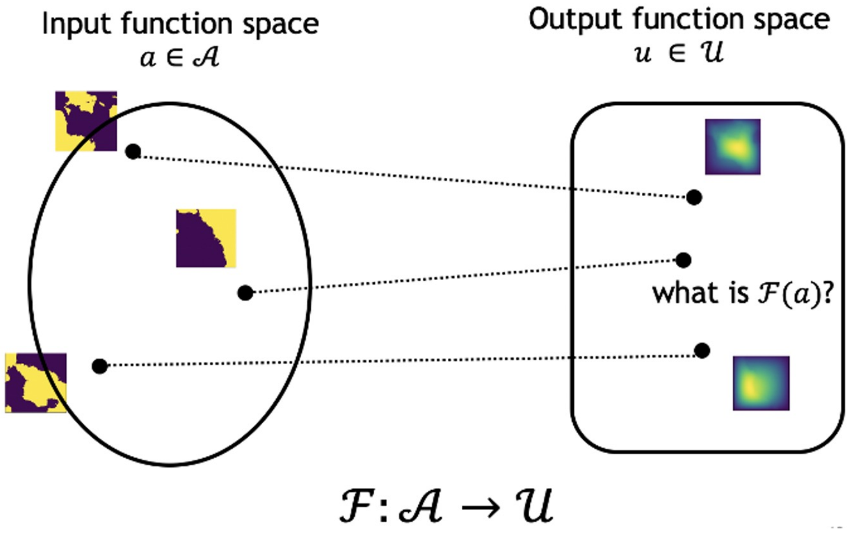
Output: solutions

Solve vs **Learn**

Solving for a PDE instance u
Approximate $u(x)$ in spatial space

Learn the solution operator





Linear Algebra Review

A set of linearly independent vectors $\{v_1, \ldots, v_n\}$ in a vector space V is a **basis** for V

if and only if any vector $u \in V$ can be written as a linear combination

$$u = \sum_{k=1}^{n} c_k v_k$$

A vector space can have several bases, and n is the **dimension** of the vector space V

$\{v_1, \ldots, v_n\}$ are called the **basis vectors**

$(c_1, \ldots, c_n)$ are the coordinates of the vector u in the basis $\{v_1, \ldots, v_n\}$

Function Spaces

Given a basis of functions $\{\varphi_k\}_{k=1}^{\infty}$ for a linear function space $\mathbb{F}$, we can represent any function $f \in \mathbb{F}$ as a linear combination

$$f = \sum_{k=1}^{\infty} c_k \varphi_k$$

$\{c_k\}_{k=1}^{\infty}$ are the “coordinates” of f in the basis $\{\varphi_k\}_{k=1}^{\infty}$

For numerical purposes, we usually truncate the series after N terms

Using Polynomials

Power Series $\sum_{k=0}^{\infty} c_k (x-a)^k$

→ basis example $\{1, x, x^2, x^3, x^4, \ldots\}$
then $f(x) = 3x^4 + 2x + 1$ can be represented as $f = (1,2,0,0,3,0,0,\ldots)$

Weierstrass Approximation Theorem: Given a function $f \in C([a,b],\mathbb{R})$ and $\varepsilon > 0$, there is a polynomial p such that $\|f-p\|_\infty < \varepsilon$ on $[a,b]$.

Taylor Polynomials. Let $f: \mathbb{R} \to \mathbb{R}$ be K times differentiable at $a \in \mathbb{R}$.

$$f(x) \approx \sum_{k=0}^{K} \frac{f^{(k)}(a)}{k!}(x-a)^k$$

Using Trigonometric Polynomials

Fourier series of the function $f(x)$ for $x \in [-L, L]$

$$f(x) = \sum_{k=0}^{\infty} A_k \cos\left(\frac{k\pi x}{L}\right) + \sum_{k=0}^{\infty} B_k \sin\left(\frac{k\pi x}{L}\right)$$

$$f(x) = \sum_{k=-\infty}^{\infty} C_k \exp\left(i\frac{k\pi x}{L}\right)$$

Basis $\left\{1, \cos\left(\frac{\pi x}{L}\right), \sin\left(\frac{\pi x}{L}\right), \cos\left(\frac{2\pi x}{L}\right), \sin\left(\frac{2\pi x}{L}\right), \cos\left(\frac{3\pi x}{L}\right), \sin\left(\frac{3\pi x}{L}\right), \ldots\right\}$

Theorem: The trigonometric polynomials are dense in the space of periodic continuous functions with the uniform norm

Universal Approximation Theorem for Functions

Theorem (Cybenko, 1989)

Given a continuous sigmoidal function σ, the finite sums of the form

$$\sum_{k=1}^{n} a_k \, \sigma\left(y_k^\top x + \theta_k\right)$$

are dense in $C(I_d)$.

Universal Approximation Theorem for Operators

Theorem 1 (Universal Approximation Theorem for Operator). *Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \ldots, n$, $k = 1, \ldots, p$, $j = 1, \ldots, m$, such that*

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} \right| < \epsilon \qquad (1)$$

holds for all $u \in V$ and $y \in K_2$.

Chen & Chen 1995

DeepONets

We want to learn a representation $\mathcal{G}_\theta$ of a mapping $\mathcal{G}$ which

- takes an input function u
- returns an output function $\mathcal{G}(u)(\cdot)$

such that

$$\mathcal{G}_\theta(u)(y) \approx \mathcal{G}(u)(y) \quad \forall u, y$$

In practice, the input function u is represented in a discrete way, typically through function values $[u(x_1), \dots, u(x_s)]$ at a finite set of sensor locations $\{x_1, \dots, x_s\}$

DeepONets

DeepONet consists of
- a **Branch Network** encoding the input function values $[u(x_1), \ldots, u(x_s)]$
- a **Trunk Network** encoding the location y where the output function is evaluated

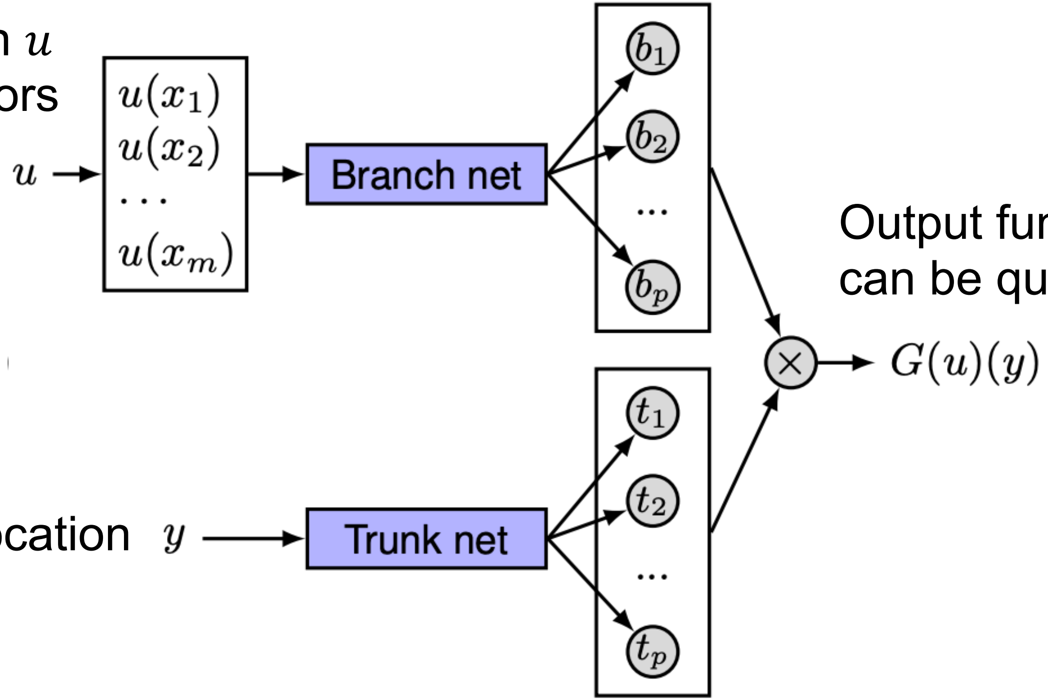
The output of the operator representation is then given by

$$\mathcal{G}_\theta(u)(y) = \sum_{k=1}^{p} b_k(u) t_k(y)$$

where $\{b_1, \ldots, b_p\}$ and $\{t_1, \ldots, t_p\}$ are the outputs of the branch and trunk networks in some p-dimensional latent space, respectively

DeepONets

Input function u
at fixed sensors
$\{x_1, \ldots, x_m\}$



Query location y

Output function $G(u)$
can be queried at any location y

Guarantees

Theorem 1.1. *Suppose* $D_u \subset \mathbb{R}^{d_u}$ *and* $D_v \subset \mathbb{R}^{d_v}$ *are compact sets. Let* $\mathcal{G}$ *be a nonlinear continuous operator mapping a subset of* $C(D_u)$ *into* $C(D_v)$. *Then, given* $\varepsilon > 0$, *there exists a DeepONet* $\mathcal{G}_\theta$ *such that* $|\mathcal{G}(u)(y) - \mathcal{G}_\theta(u)(y)| < \varepsilon$ *for all* $u \in \mathcal{U}$, $y \in D_v$.

Theorem 1.2. *Let* μ *be a probability measure on* $C(D_u)$, *and* $\mathcal{G}$ *a Borel measurable mapping on* $C(D_u)$ *with* $\mathcal{G} \in L^2(\mu)$. *Then for every* $\varepsilon > 0$, *there is a DeepONet* $\mathcal{G}_\theta$ *such that* $\|\mathcal{G} - \mathcal{G}_\theta\|_{L^2(\mu)} < \varepsilon$.

Some error estimates are also available

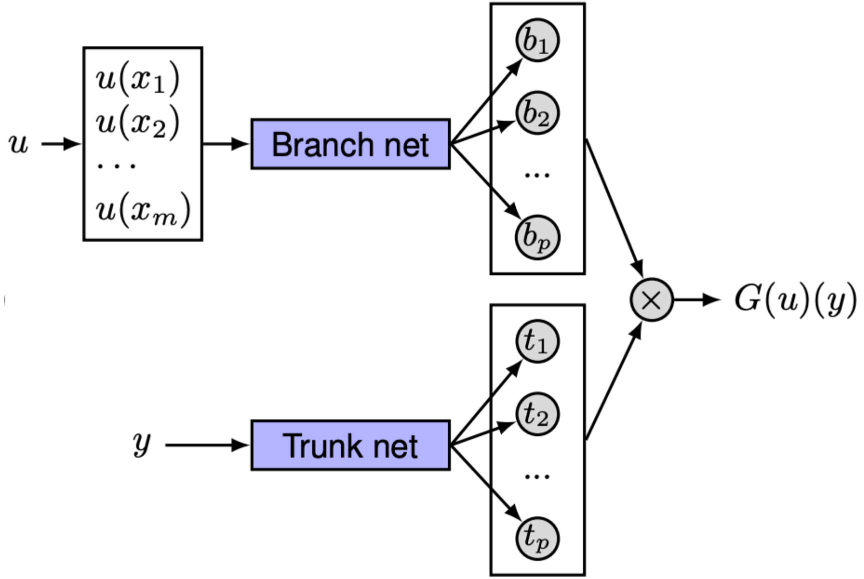
DeepONet Code

```
class BranchNet(nn.Module):
    def __init__(self):
        super(BranchNet, self).__init__()
        . . .

class TrunkNet(nn.Module):
    def __init__(self):
        super(TrunkNet, self).__init__()
        . . .

class DeepONet(nn.Module):
    def __init__(self, branch_net, trunk_net):
        super(DeepONet, self).__init__()
        self.branch_net = branch_net
        self.trunk_net = trunk_net

    def forward(self, u, y):
        branch_out = self.branch_net(u)
        trunk_out = self.trunk_net(y)
        return torch.sum(branch_out * trunk_out, dim=1)
```



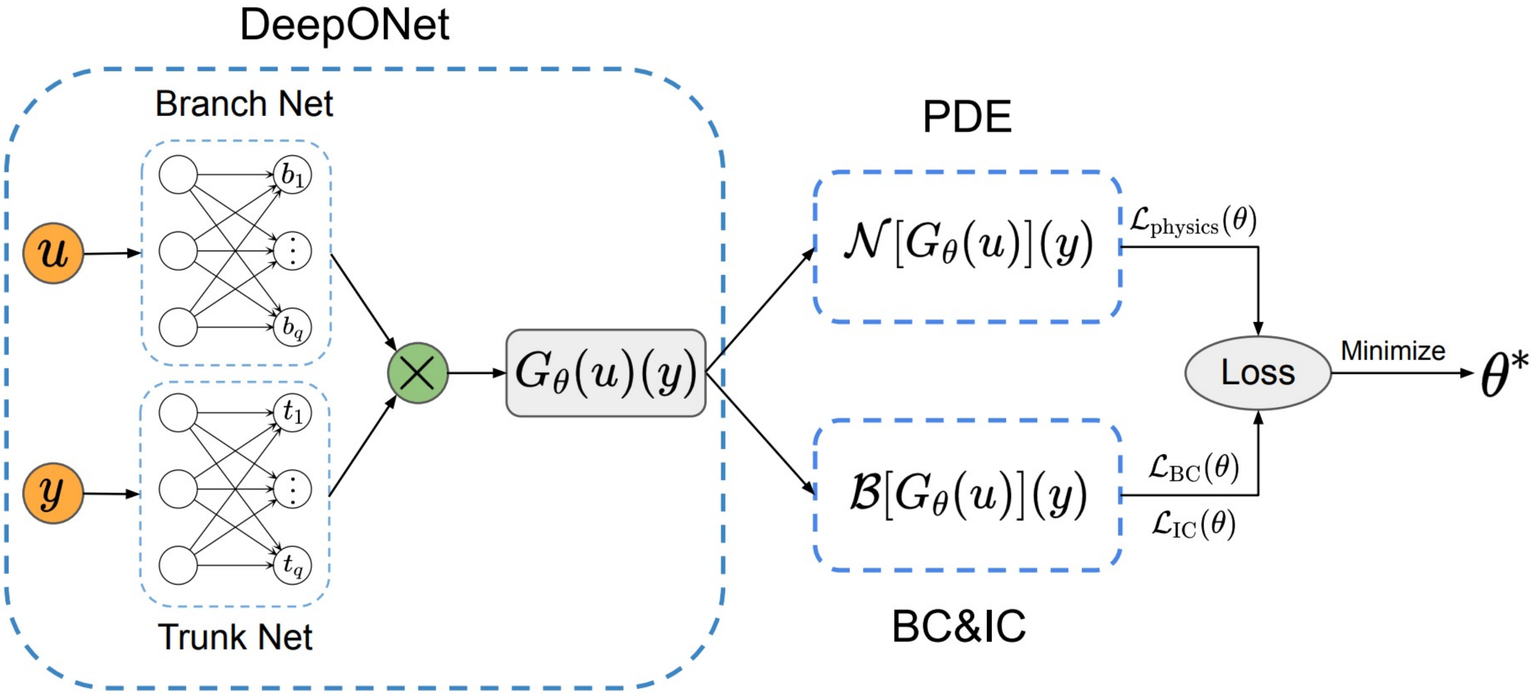
DeepONet extensions and applications

DeepONet is very simple and general

In practice, probably want to tailor it more specifically to the application
- Many extensions
- Different architectures as the branch and trunk networks
- Different ways of combining the branch and trunk outputs
- Multiple inputs, multiple outputs, embeddings, …

https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=deeponet&btnG=

Physics-Informed DeepONets



Super-resolution example

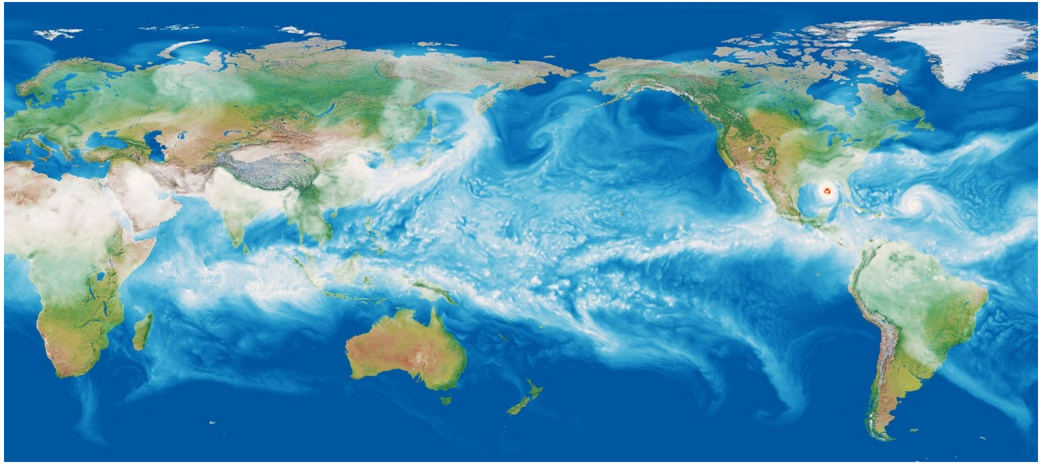
Many existing Computer Vision approaches for image and video SR

Many of these approaches have been adapted for spatiotemporal scientific data, in particular by adding physics losses to the objective functions

But most of these approaches are based on function learning
→ same limitations as PINNs

Super-resolution example

High-resolution simulations are often required to capture faithfully essential dynamics which occur at small spatiotemporal scales



Numerical integrators become too expensive as the resolution becomes large

Super-resolution example

SR can be framed as an operator learning problem:

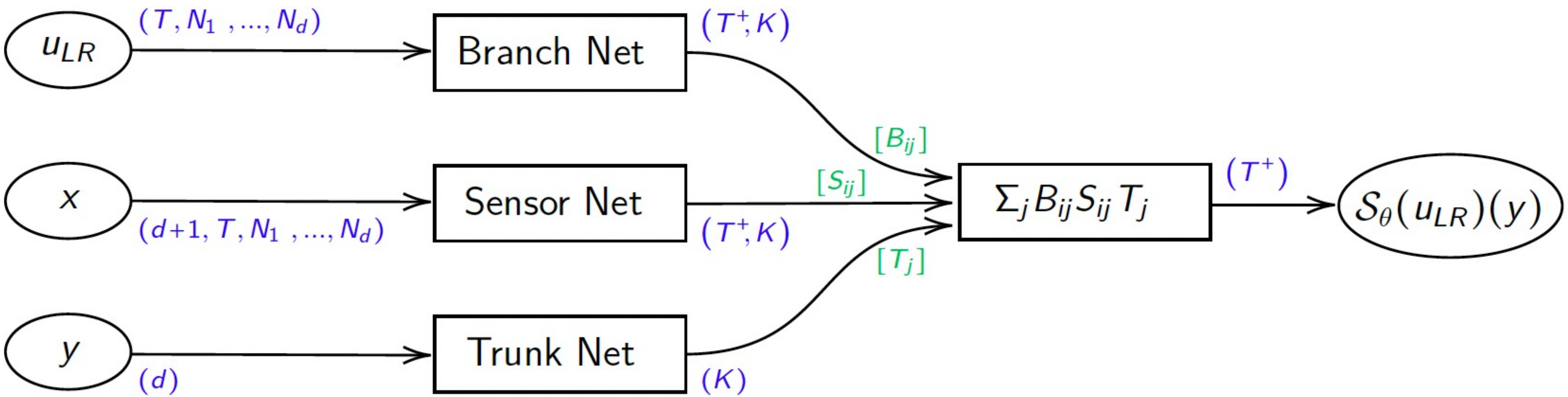
Learn a representation $\mathcal{S}_\theta$ for the operator $\mathcal{S}$ which
- takes a low-resolution simulation u_{LR} as input
- returns the solution u of a parametric PDE

Given a low-resolution simulation u_{LR}, we then obtain a continuous-time representation $u_\theta = \mathcal{S}_\theta(u_{LR})$ of the PDE solution u

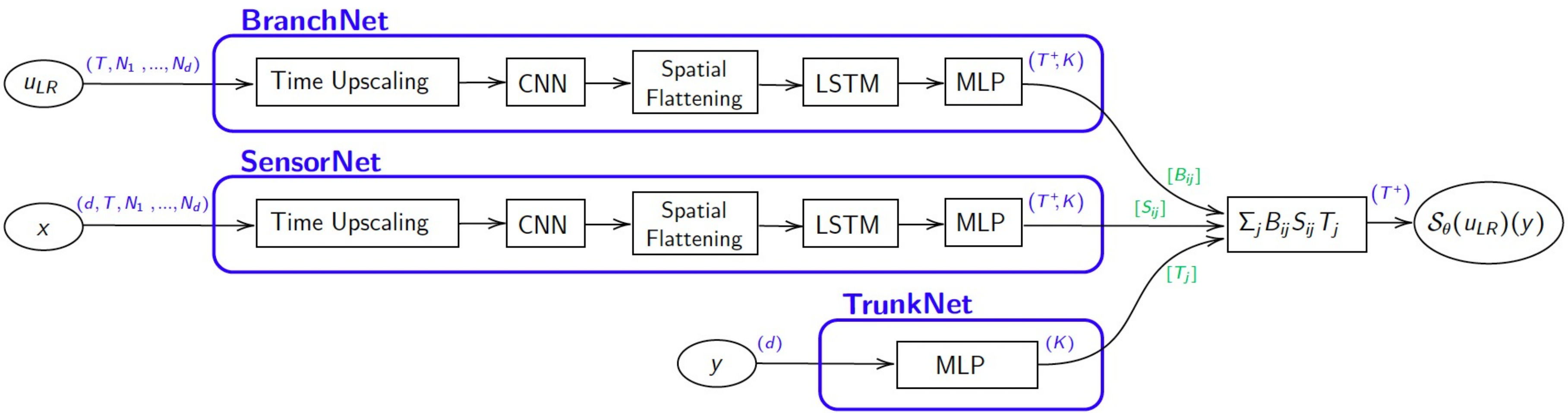
The continuous-time representation u_θ can then be evaluated on a higher-resolution grid to obtain a high-resolution simulation u_{HR}

Super-resolution example

Modified DeepONet-like sequence-to-sequence architecture



Super-resolution example



Super-resolution example

Consider the 2D Navier–Stokes equations in vorticity form for a viscous incompressible fluid

$$\partial_t \omega(x,t) + u(x,t)\cdot\nabla\omega(x,t) = \frac{1}{Re}\Delta\omega(x,t) + f(x) \qquad x \in (0,1)^2,\; t \in (0,T],$$

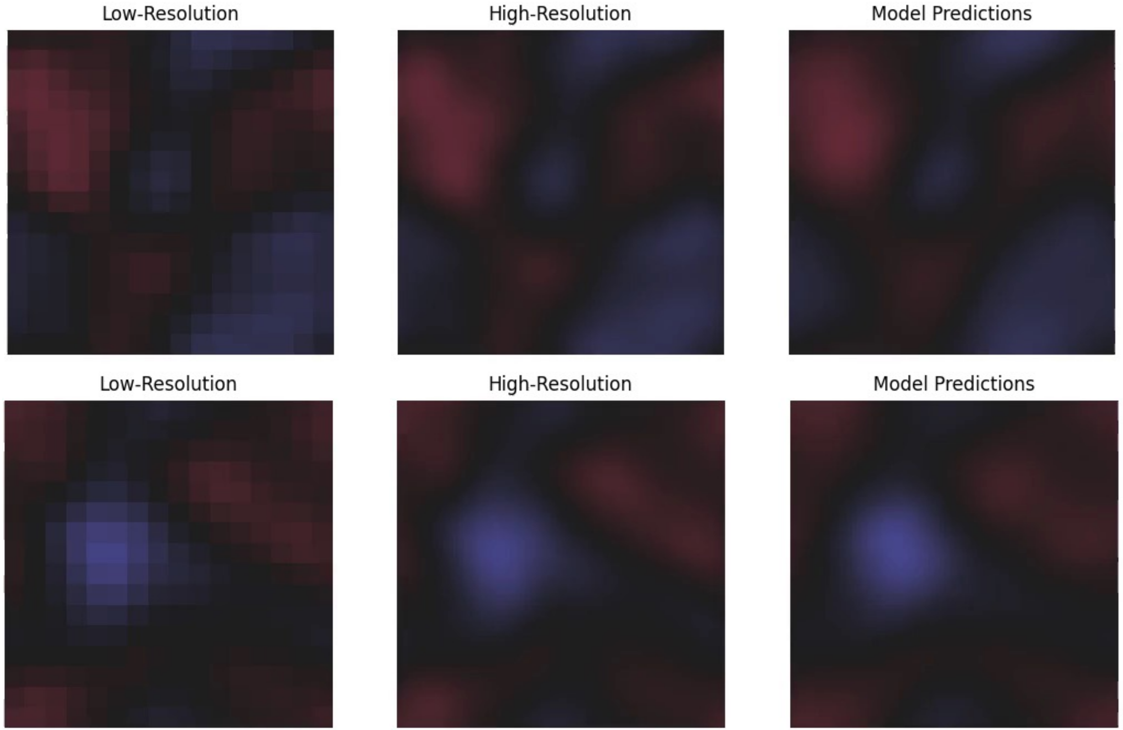
$$\nabla \cdot u(x,t) = 0 \qquad x \in (0,1)^2,\; t \in (0,T],$$

$$\omega(x,0) = \omega_0(x) \qquad x \in (0,1)^2,$$

where ω is vorticity, u represents the velocity field, Re is the Reynolds number, the initial condition $\omega_0(x)$ is sampled from a Gaussian random field, and the source term $f(x)$ is given by

$$f(x) = 0.1\left[\sin(2\pi(x_1+x_2)) + \cos(2\pi(x_1+x_2))\right]$$

Super-resolution example



References

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3:218–229, 03 2021. doi: 10.1038/s42256-021-00302-5.

Goswami, S., Bora, A., Yu, Y., Karniadakis, G.E. (2023). Physics-Informed Deep Neural Operator Networks. In: Rabczuk, T., Bathe, KJ. (eds) Machine Learning in Modeling and Simulation. Computational Methods in Engineering & the Sciences. Springer, Cham.

Wang S, Wang H, Perdikaris P. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Sci Adv. 2021 Oct;7(40):eabi8605. doi: 10.1126/sciadv.abi8605. Epub 2021 Sep 29. PMID: 34586842; PMCID: PMC8480920.

V. Duruisseaux, and A. Chakraborty. An Operator Learning Framework for Spatiotemporal Super-Resolution of Scientific Simulations. 2023.

S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepONets: a deep learning framework in infinite dimensions. Transactions of Mathematics and Its Applications, 6(1), 03 2022. ISSN 2398-4945. doi: 10.1093/imatrm/tnac001.

T. Chen and H. Chen. Approximations of continuous functionals by neural networks with application to dynamic systems. IEEE Transactions on Neural Networks, 4(6):910–918, 1993. doi: 10.1109/72.286886.

T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its applications to dynamic systems. Neural Networks, IEEE Transactions on, pages 911 – 917, 08 1995. doi: 10.1109/72.392253.

T. Chen and H. Chen. Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. IEEE Transactions on Neural Networks, 6(4):904–910, 1995. doi: 10.1109/72.392252.