

基于神经网络的代理模型

黄政宇

北京大学北京国际数学研究中心

北京大学国际机器学习研究中心



本堂课大纲

➤ 卷积神经网络

➤ 神经算子

- PCA-Net

- 傅里叶神经算子

- 深度神经算子(DeepONet)

- Transformer



偏微分方程问题

➤ 偏微分方程问题

$$\begin{aligned} \mathcal{L}(x, u, a) &= 0 & x \in \Omega \\ \mathcal{B}(x, u) &= 0 & x \in \partial\Omega \end{aligned}$$

➤ 代理模型 (surrogate model)

两个实Banach空间 $\mathcal{A} = \{a: D \mapsto \mathbb{R}\}$ $\mathcal{U} = \{u: D \mapsto \mathbb{R}\}$,

求解偏微分方程给出映射 $\mathcal{G}^\dagger: \mathcal{A} \mapsto \mathcal{U}$

降阶模型、高斯回归模型 $\mathcal{G}_\theta \approx \mathcal{G}^\dagger$ 是一个这样的映射的近似

基于神经网络的代理模型 $\mathcal{G}_\theta \approx \mathcal{G}^\dagger$

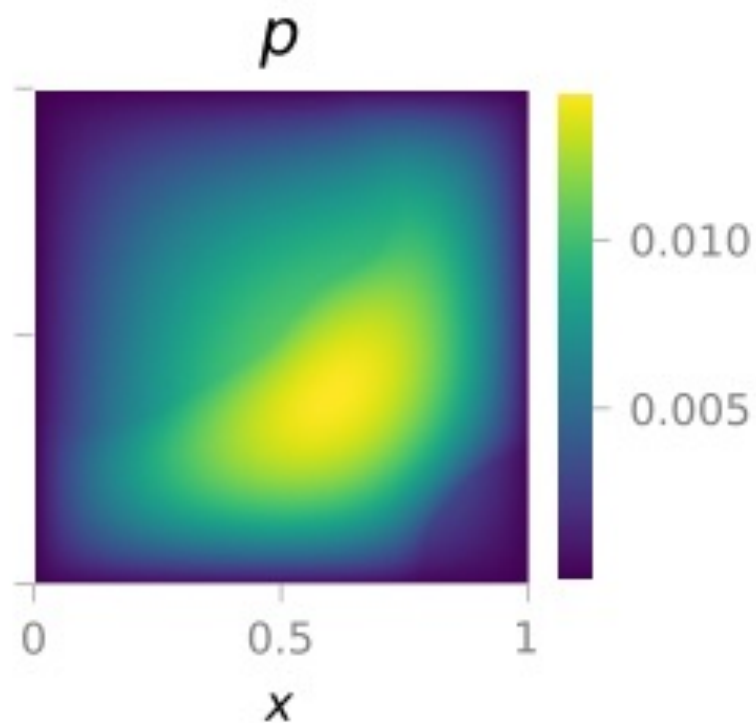
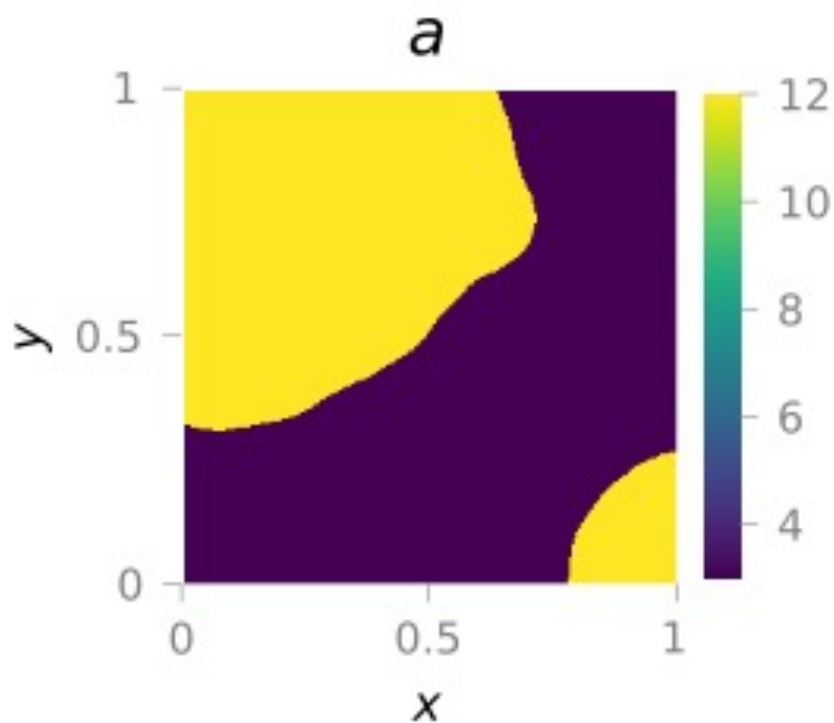


卷积神经网络

➤ Darcy 方程

$$\begin{aligned} -\nabla \cdot (a(x)\nabla p) &= f(x), & x \in D \\ p(x) &= 0 & x \in \partial D \end{aligned}$$

$$\mathcal{G}^\dagger : a \mapsto p$$





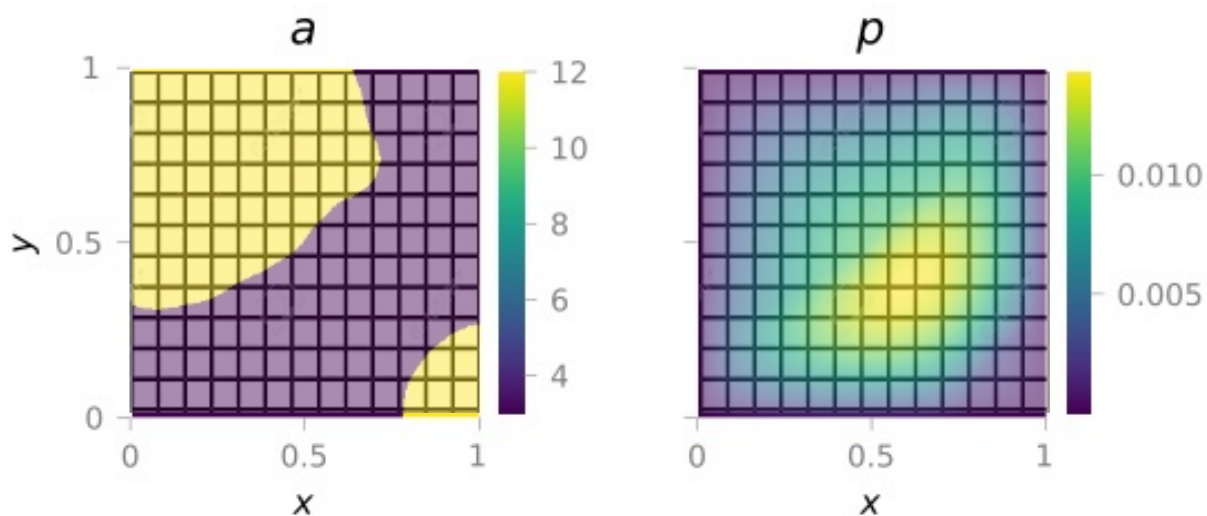
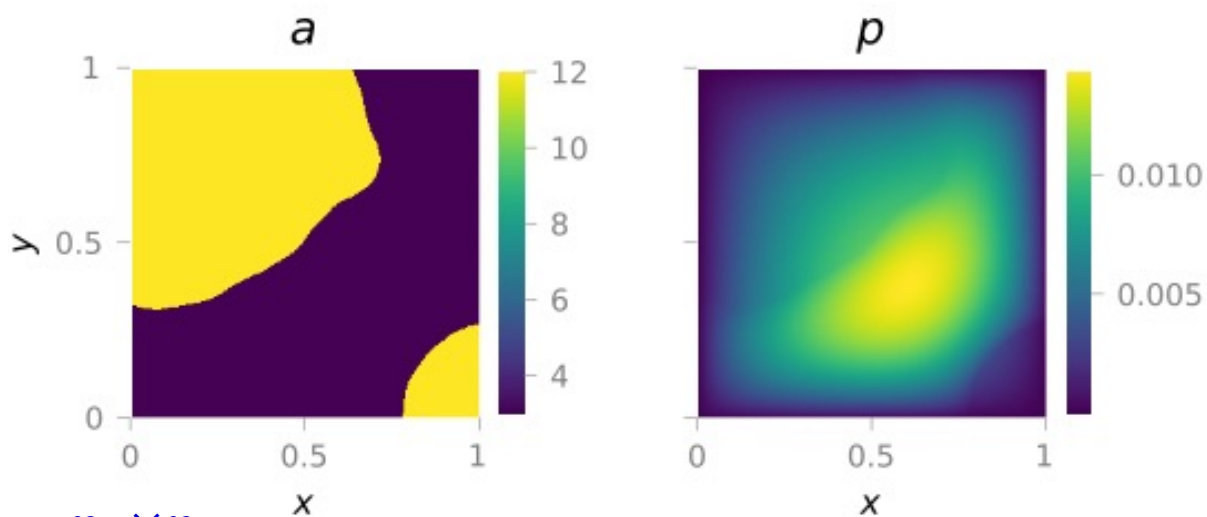
卷积神经网络

➤ 图像到图像的映射 (Zhu 2018)

$$g^\dagger : a \mapsto p$$

数值离散：

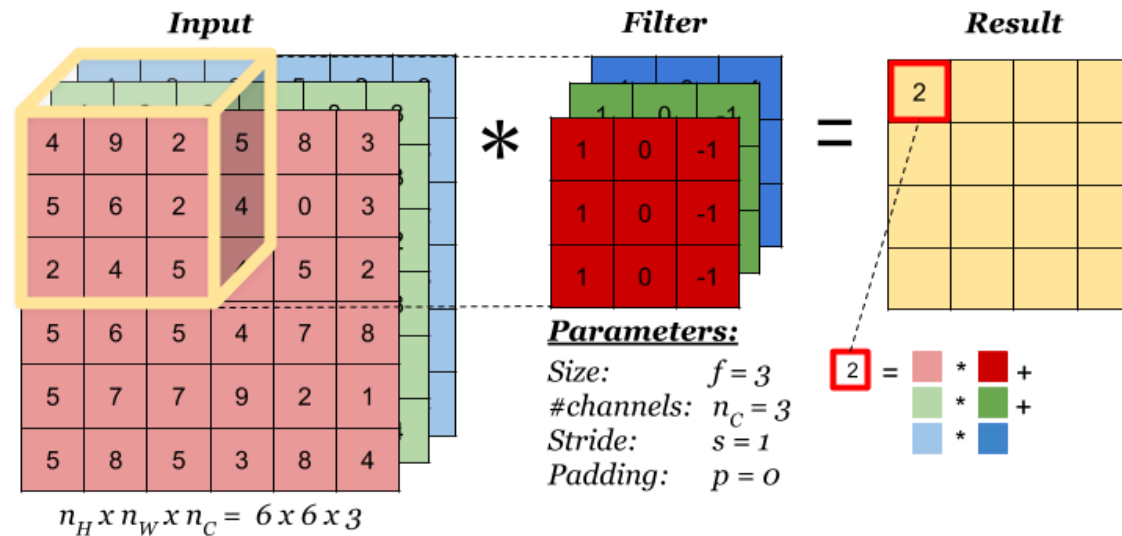
$$f^\dagger : \tilde{a} \in R^{n_x \times n_y} \mapsto \tilde{p} \in R^{n_x \times n_y}$$



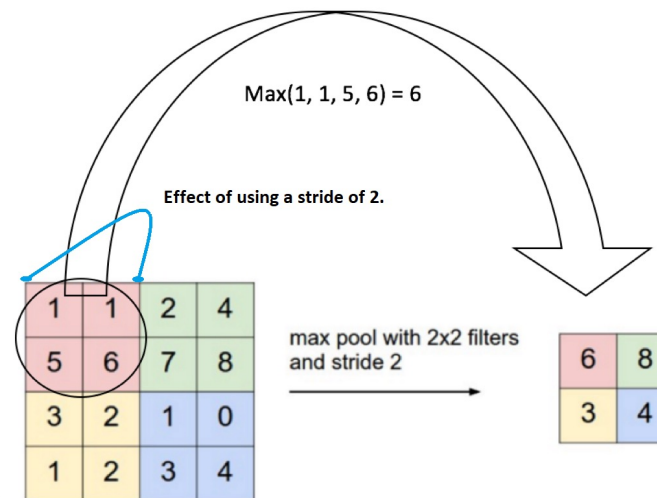


卷积神经网络

卷积模块



下采样层 (Pooling layer)

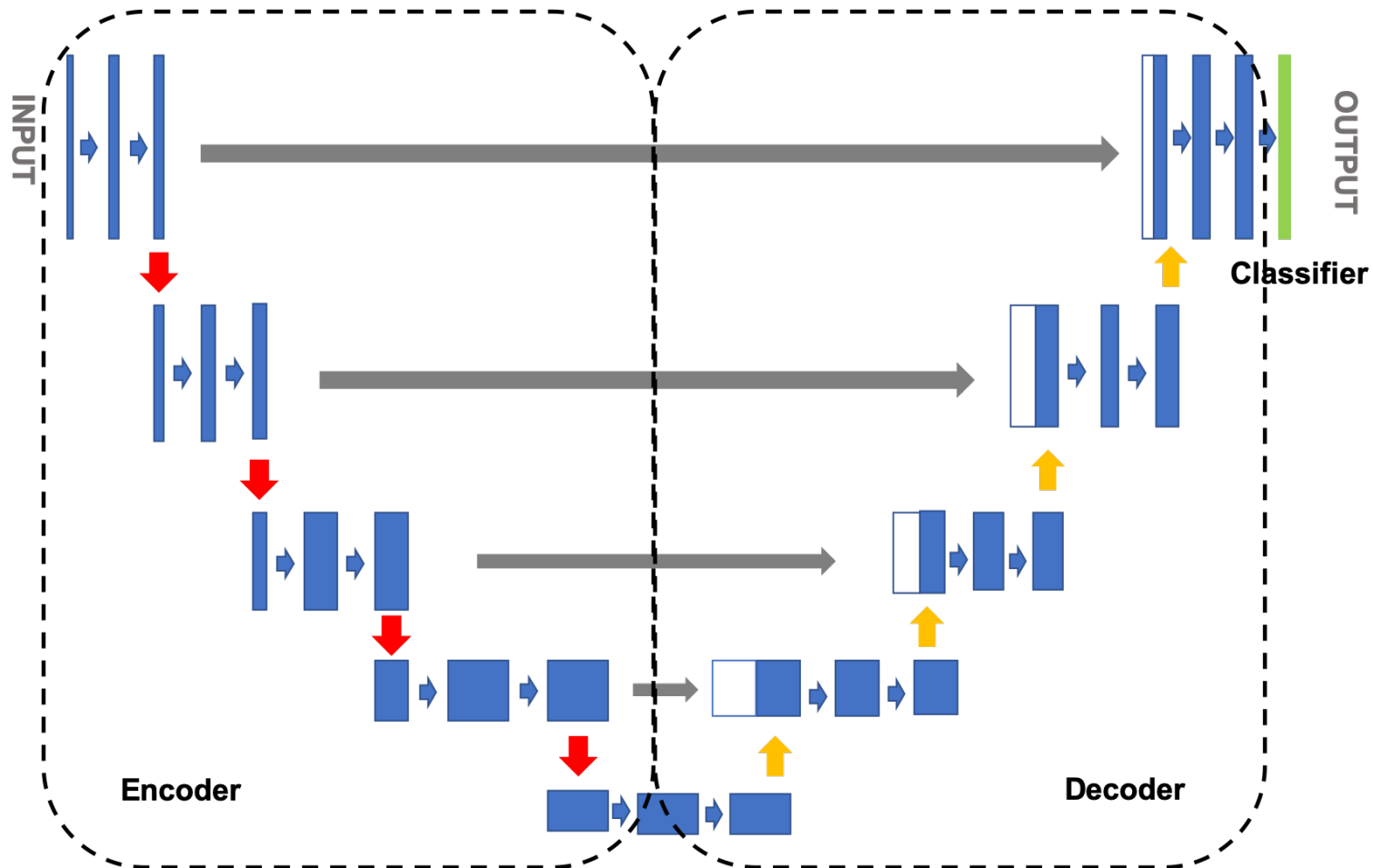


We do the pooling with 2x2 filters, so we will divide an input image on 2x2 regions, and we will use a stride of 2. Because we are using a stride of 2, these regions don't overlap.



卷积神经网络

➤ U-Net (Ronneberger 2015)





卷积神经网络

➤ U-Net (Ronneberger 2015)

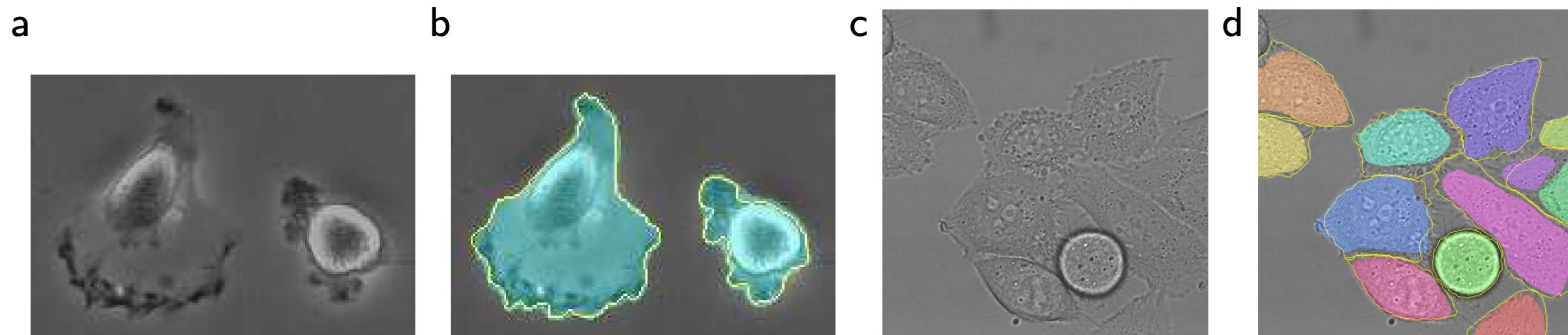


Fig. 4. Result on the ISBI cell tracking challenge. (a) part of an input image of the “PhC-U373” data set. (b) Segmentation result (cyan mask) with manual ground truth (yellow border) (c) input image of the “DIC-HeLa” data set. (d) Segmentation result (random colored masks) with manual ground truth (yellow border).



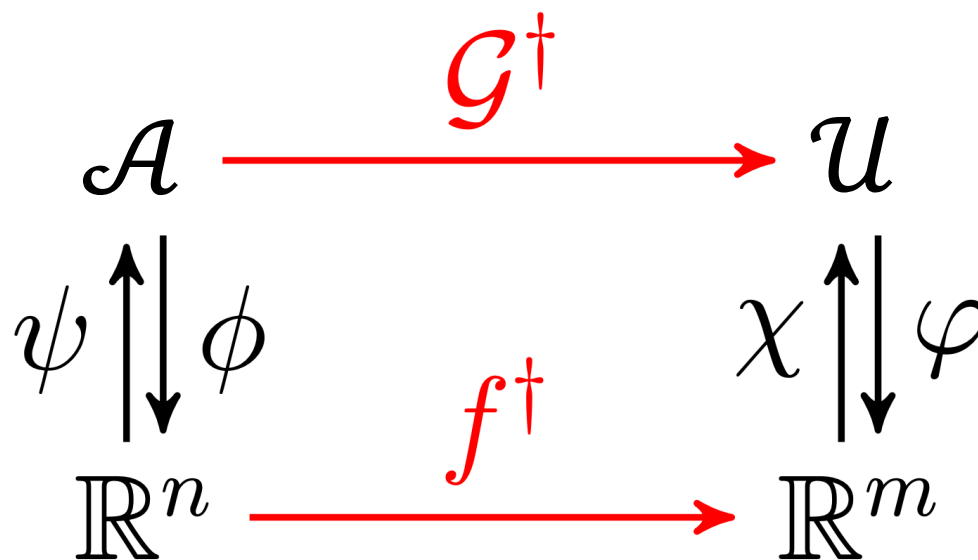
卷积神经网络

➤ Darcy 方程

$$\begin{aligned} -\nabla \cdot (a(x)\nabla p) &= f(x), & x \in D \\ p(x) &= 0 & x \in \partial D \end{aligned}$$

$$\mathcal{G}^\dagger : a \mapsto p$$

$$f^\dagger : \tilde{a} \in \mathbb{R}^{n_x \times n_y} \mapsto \tilde{p} \in \mathbb{R}^{n_x \times n_y}$$



直接近似 f^\dagger



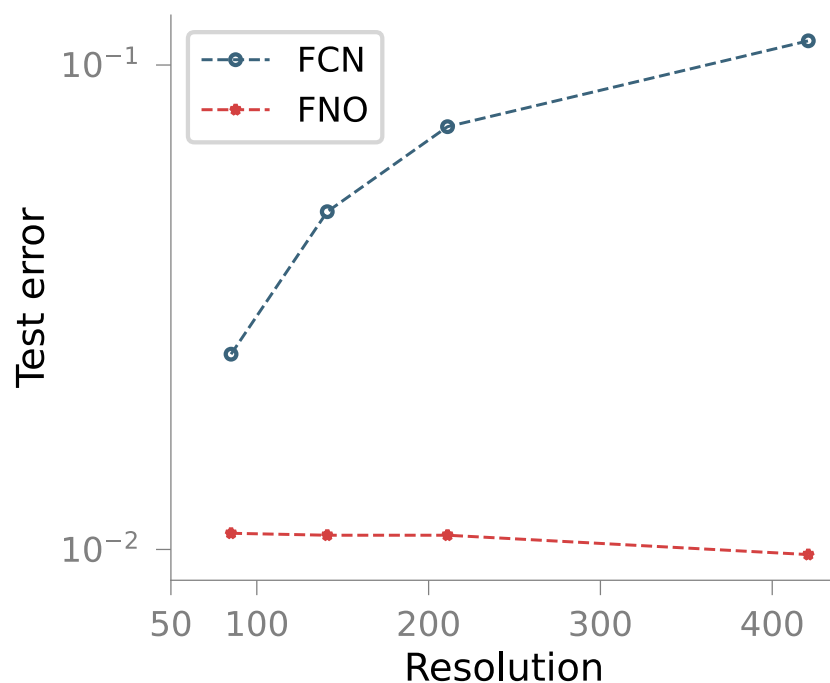
卷积神经网络

➤ Darcy 方程

$$\begin{aligned} -\nabla \cdot (a(x)\nabla p) &= f(x), & x \in D \\ p(x) &= 0 & x \in \partial D \end{aligned}$$

$$\mathcal{G}^\dagger : a \mapsto p$$

$$f^\dagger : \tilde{a} \in R^{n_x \times n_y} \mapsto \tilde{p} \in R^{n_x \times n_y}$$



诸如 Fully Convolutional Neural Network (FCN) 或者 U-Net 这样先离散再近似的神经网络方法，对于不同的离散方式，如果不改变网络结构（比如卷积核尺寸）加密网格会导致测试误差将变差。



本堂课大纲

- 卷积神经网络
- 神经算子
 - PCA-Net
 - 傅里叶神经算子
 - 深度神经算子(DeepONet)
 - Transformer



神经算子

➤ 神经网络

函数(function) : $\mathbb{R}^n \mapsto \mathbb{R}^m$

线性函数 : $x \mapsto Ax + b$

逐点激活函数 : $x \mapsto \sigma(x)$

➤ 神经算子

算子(operator) : $\mathcal{A} \mapsto \mathcal{U}$

$$\mathcal{A} = \{a: D \mapsto \mathbb{R}\}$$

$$\mathcal{U} = \{u: D \mapsto \mathbb{R}\}$$

线性算子 :

$$a(x) \mapsto \int \kappa(x, y) a(y) dy + b(x)$$

逐点激活函数 : $a(x) \mapsto \sigma(a(x))$



线性算子

➤ 核函数

正交基底： $\Psi = [\psi_1 \ \psi_2 \ \cdots \ \psi_l]$

矩阵： $A \in R^{l \times l}$

$$\kappa(x, y) = \Psi(x)A\Psi(y)^T$$

➤ 线性算子

$$a(x) \mapsto \int \kappa(x, y)a(y)dy + b(x)$$

$$= \int \Psi(x)A\Psi(y)^T a(y)dy + b(x)$$

计算系数

$$= \Psi(x)A \int \Psi(y)^T a(y)dy + b(x)$$

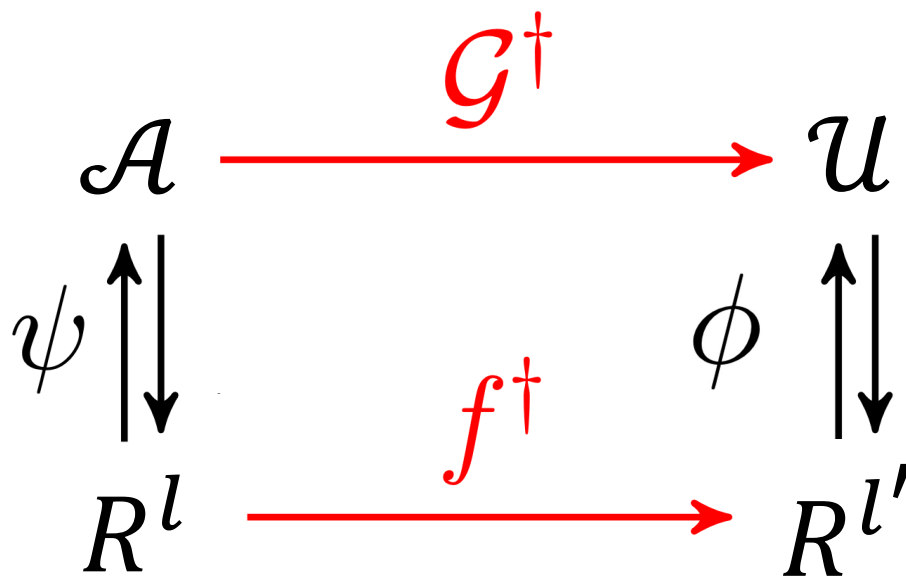


PCA神经网络

➤ PCA-Net(Hesthaven 2018, Bhattachary 2021)

$$a(x) = \sum \hat{a}_i \psi_i(x) \quad u(x) = \sum \hat{u}_i \phi_i(x)$$

$\{\psi_i\}_{i=1}^{l'}$ 和 $\{\phi_j\}_{j=1}^{l'}$ 分别是输入和输出函数的基底，
采用本征正交分解从输入输出训练集中得到。





PCA神经网络

万能逼近定理(Bhattacharya 2021)

设 \mathcal{A} 和 \mathcal{U} 为实的、可分的 Hilbert 空间，设 μ 是在 \mathcal{A} 上的概率测度，且满足 $\mathbb{E}_{a \sim \mu} \|a\|^4 < \infty$ 。假设映射 $G^\dagger: \mathcal{A} \rightarrow \mathcal{U}$ 是 μ -可测的，并且全局 Lipschitz 连续。给定 $d_{\mathcal{A}}$ 和 $d_{\mathcal{U}}$ ，训练数据量 n ，误差参数 $\epsilon > 0$ 那么存在PCA神经网络 G_θ 满足：

$$\begin{aligned} & \mathbb{E}_{\{a_i\} \sim \mu} \mathbb{E}_{a \sim \mu} \|G^\dagger(a) - G_\theta(a)\|^2 \\ & \leq C(\epsilon^2 + \sqrt{\frac{d_{\mathcal{A}}}{n}} + R^\mu(V_{d_{\mathcal{A}}}) + \sqrt{\frac{d_{\mathcal{U}}}{n}} + R^{G^\dagger \# \mu}(V_{d_{\mathcal{U}}})) \end{aligned}$$

有限维空间的神经网络万能逼近定理

本征正交分解 \mathcal{A} 和 \mathcal{U} 空间中的投影误差



傅里叶神经算子

➤ 平移不变核函数

$$\kappa(x, y) = \kappa(x - y)$$

$$\kappa(r) = \sum \hat{\kappa}_k e^{2\pi i k \cdot r} \quad (\psi_k(x) = [e^{2\pi i k \cdot x}] \quad A = \text{diag}\{\hat{\kappa}_k\})$$

➤ 傅里叶线性算子

$$\begin{aligned} a(x) &\mapsto \int \kappa(x, y) a(y) dy + b(x) \\ &= \int \kappa(x - y) a(y) dy + b(x) \\ &= \sum \hat{\kappa}_k \hat{a}_k e^{2\pi i k x} + b(x) \end{aligned}$$

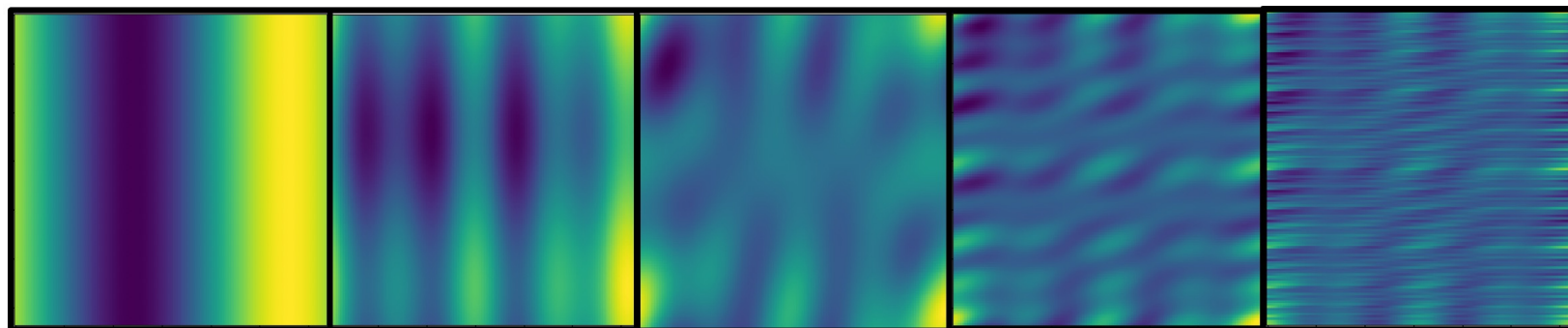


傅里叶神经算子

➤ 图像到图像的映射



Filters in CNN



Fourier Filters



傅里叶神经算子

➤ 傅里叶层

给定函数 $a(\cdot): D \rightarrow \mathbb{R}$

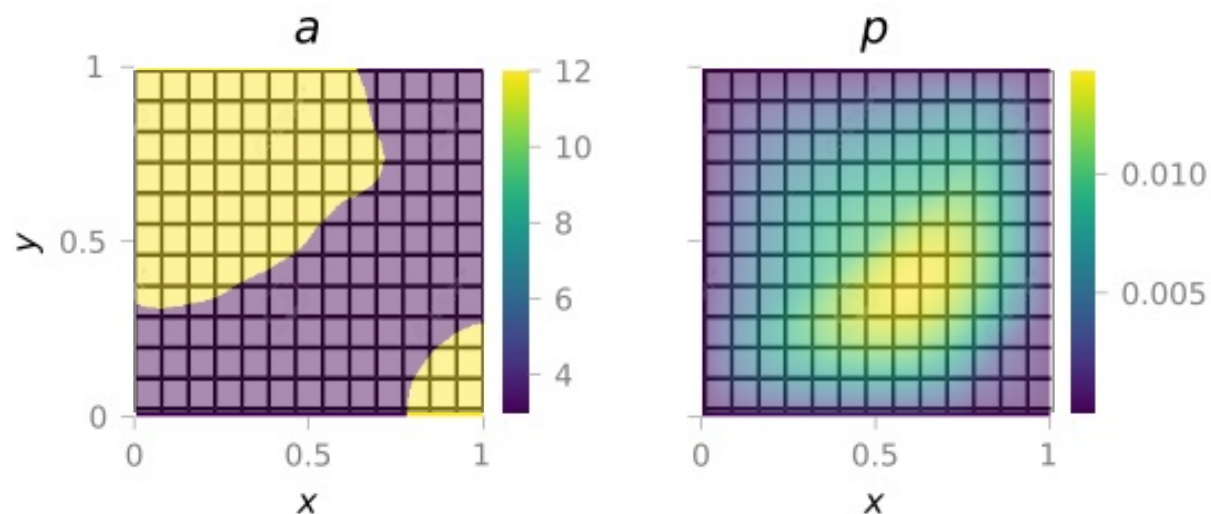
$$f(a) = h(wa + b + \mathcal{F}^{-1}A\mathcal{F}a)$$

\mathcal{F} 和 \mathcal{F}^{-1} 为傅里叶变换和傅里叶逆变换

$w, b \in \mathbb{R}$

A 为对角矩阵 $\text{diag}\{\hat{\kappa}_k\}$

h 为逐点的激活函数





傅里叶神经算子

➤ 傅里叶层

给定函数 $a(\cdot): D \rightarrow \mathbb{R}^{d_f}$

$$f(a) = h(Wa + b + \mathcal{F}^{-1}A\mathcal{F}a)$$

\mathcal{F} 和 \mathcal{F}^{-1} 为傅里叶变换和傅里叶逆变换

$$W \in \mathbb{R}^{d'_f \times d_f}, b \in \mathbb{R}^{d'_f}$$

A 为分块对角矩阵 $\text{diag}\{A_i \in \mathbb{C}^{d'_f \times d_f}\}$

h 为逐点的激活函数

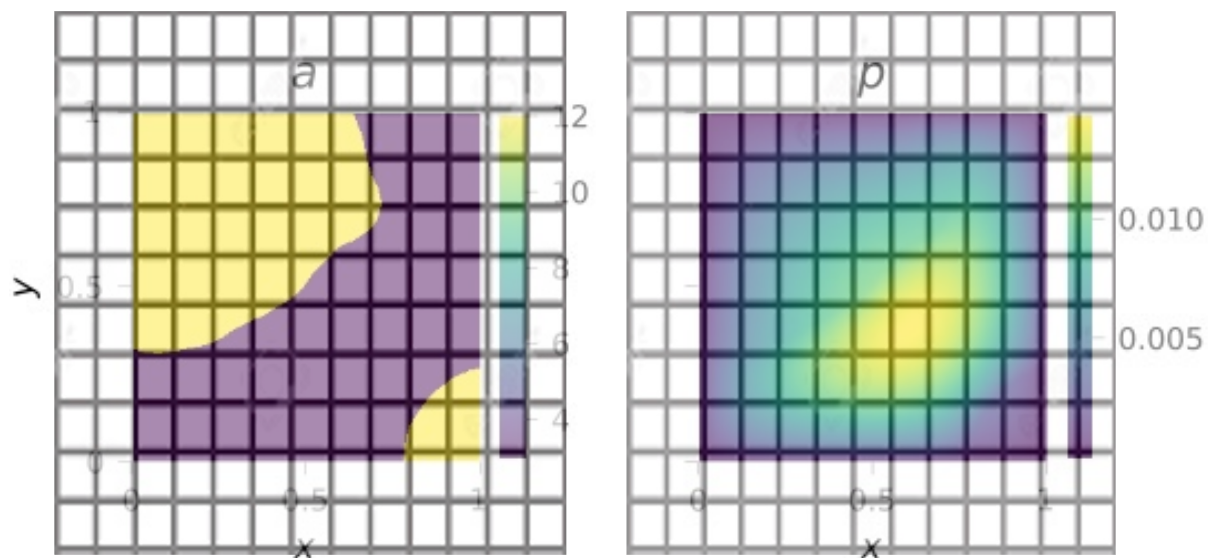
$$\begin{array}{c}
\mathcal{F}a_1 \Rightarrow \\
A\mathcal{F}(a(\xi)) = A_1 \begin{bmatrix} \hat{a}_1^{(1)} \\ \vdots \\ \hat{a}_{d_f}^{(1)} \end{bmatrix} A_2 \begin{bmatrix} \hat{a}_1^{(2)} \\ \vdots \\ \hat{a}_{d_f}^{(2)} \end{bmatrix} \dots \dots A_{k_{max}} \begin{bmatrix} \hat{a}_1^{(k_{max})} \\ \vdots \\ \hat{a}_{d_f}^{(k_{max})} \end{bmatrix}
\end{array}$$



傅里叶神经算子

➤ 傅里叶层

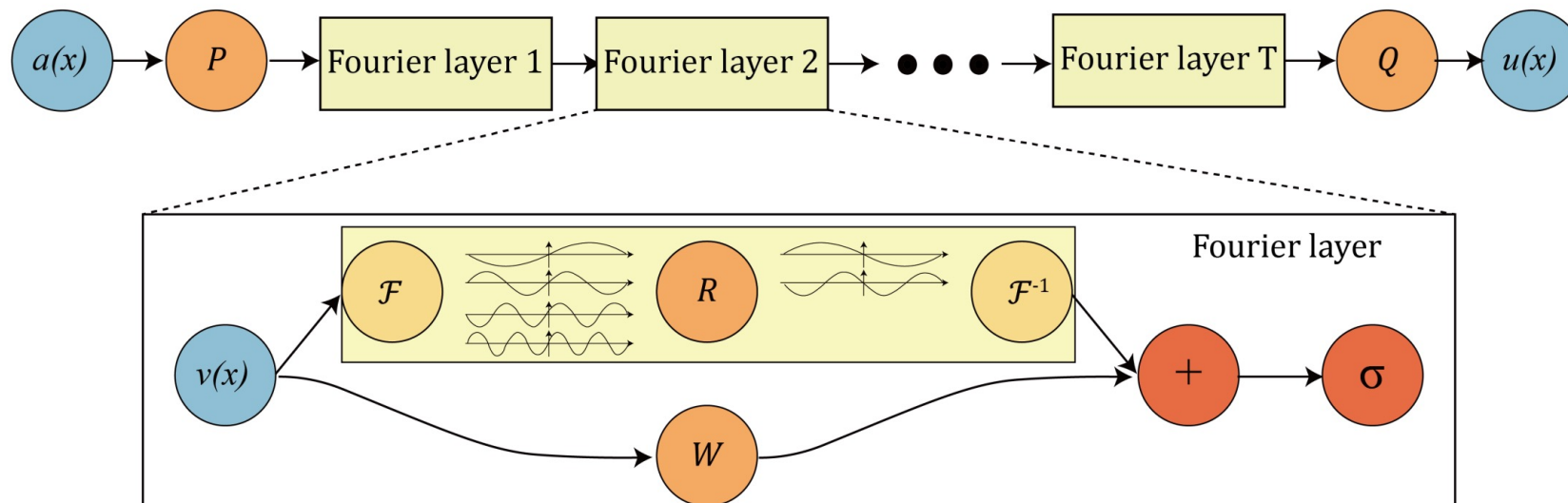
当问题非周期时，使用补零填充(padding)





傅里叶神经算子

傅里叶神经算子



升维(lifting)算子 $P: \{D \rightarrow \mathbb{R}\} \rightarrow \{D \rightarrow \mathbb{R}^{d_f}\}$
 $P: a \rightarrow Wa + b, \quad W \in \mathbb{R}^{d_f \times 1}, b \in \mathbb{R}^{d_f}$

投影(projection)算子 $Q: \{D \rightarrow \mathbb{R}^{d_f}\} \rightarrow \{D \rightarrow \mathbb{R}\}$
 $Q: a \rightarrow W^{(2)} \sigma(W^{(1)} a + b^{(1)}) + b^{(2)},$
 $W^{(1)} \in \mathbb{R}^{d_{out} \times d_f}, b^{(1)} \in \mathbb{R}^{d_{out}} \quad W^{(2)} \in \mathbb{R}^{1 \times d_{out}}, b^{(2)} \in \mathbb{R}$



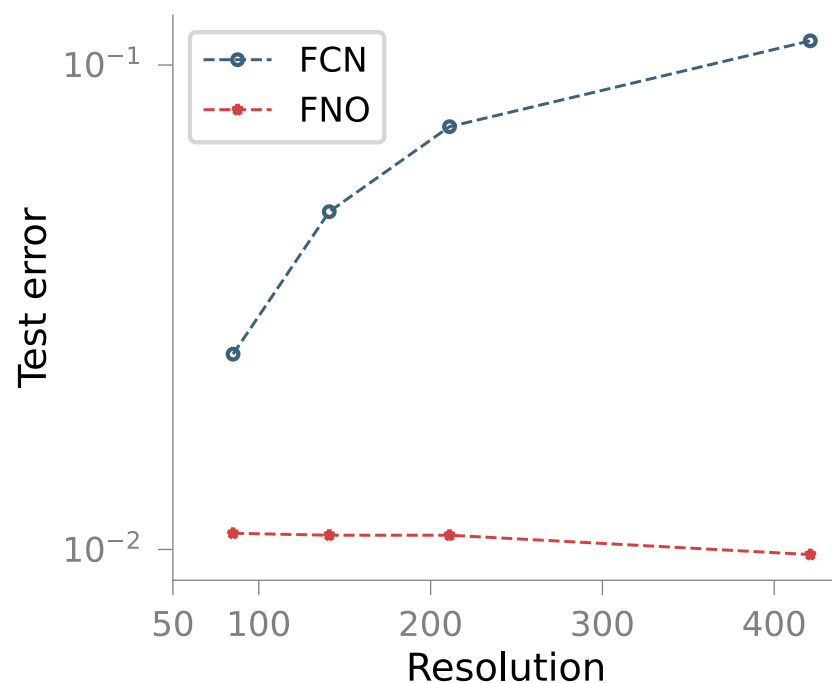
傅里叶神经算子

➤ Darcy 方程

$$\begin{aligned} -\nabla \cdot (a(x)\nabla p) &= f(x), & x \in D \\ p(x) &= 0 & x \in \partial D \end{aligned}$$

$$\mathcal{G}^\dagger : a \mapsto p$$

$$f^\dagger : \tilde{a} \in \mathbb{R}^{n_x \times n_y} \mapsto \tilde{p} \in \mathbb{R}^{n_x \times n_y}$$

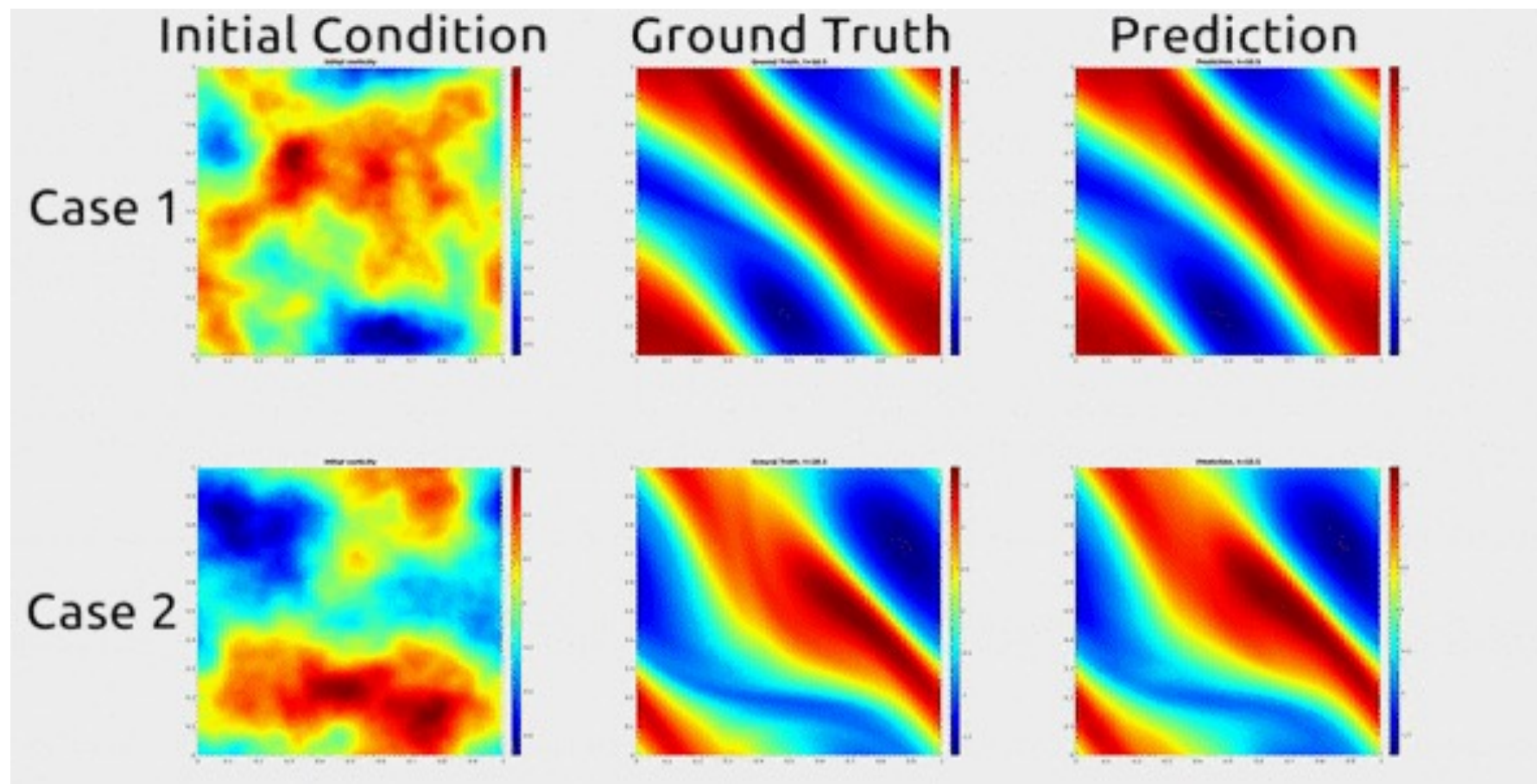


傅里叶神经算子是函数空间上的映射，与网格离散无关，因此网格加密不会导致测试误差变差。



傅里叶神经算子

➤ 2维不可压缩Navier-Stokes方程





基底选择

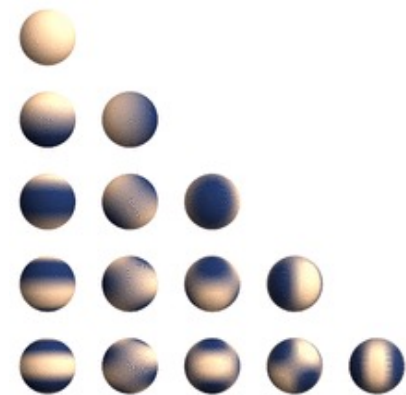
➤ 基底函数

PCA基底函数

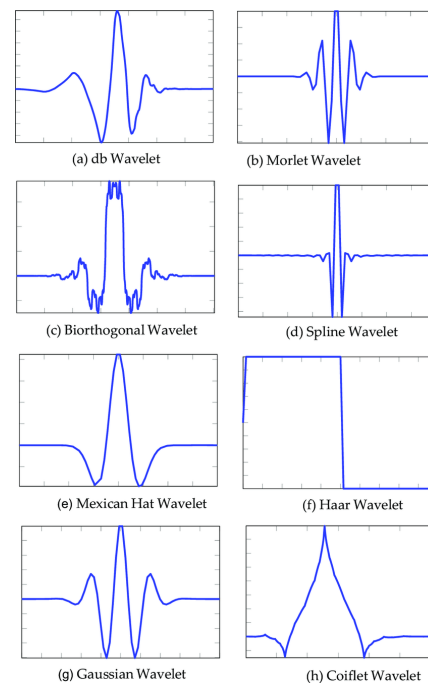
傅里叶基底函数： $e^{-2\pi ik\frac{x}{L}}$

球面上的问题：球谐函数(spherical harmonics)

多尺度问题：小波(wavelet)



从数据中学习基底函数





深度神经算子(DEEPONET)

- PCA-Net (Hesthaven 2018, Bhattachary 2021)

$$a(x) = \sum \hat{a}_i \psi_i(x) \quad u(x) = \sum \hat{u}_i^{NN}(\hat{a}) \phi_i(x)$$

$\{\psi_i\}_{i=1}^l$ 和 $\{\phi_j\}_{j=1}^{l'}$ 分别是输入和输出函数的基底，采用本征正交分解从输入输出训练集中得到。

- 深度神经算子(Deep Operator Networks)

$$a(x) = \sum \hat{a}_i \psi_i(x) \quad u(x) = \sum \hat{u}_i^{NN}(\hat{a}) \hat{\phi}_i^{NN}(x)$$

\hat{u}_i^{NN} : 分支网络(branch network)

$\hat{\phi}_i^{NN}$: 主干网络(trunk network)

同时从数据中进行学习



深度神经算子(DEEPONET)

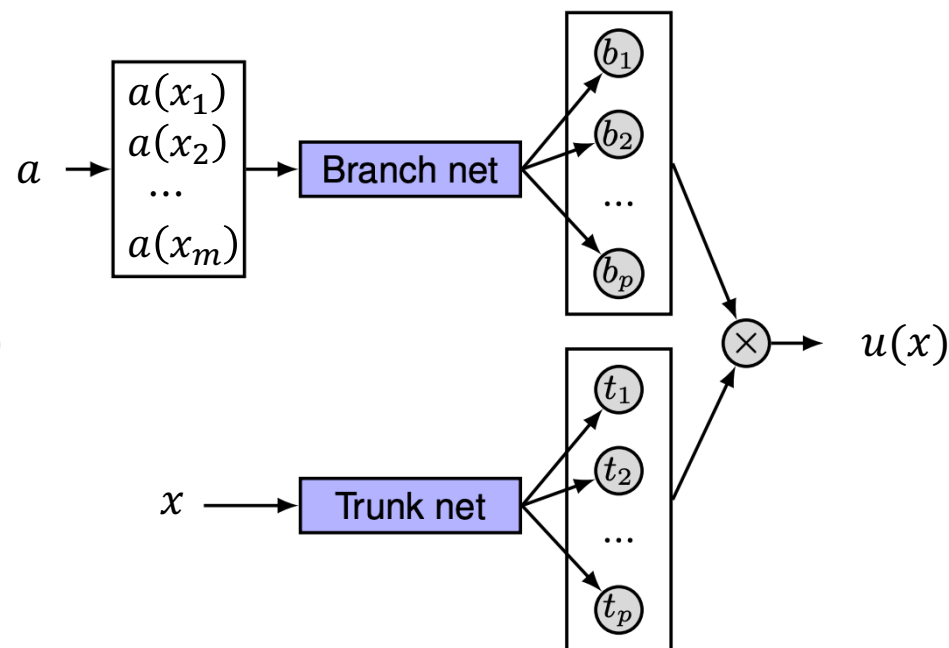
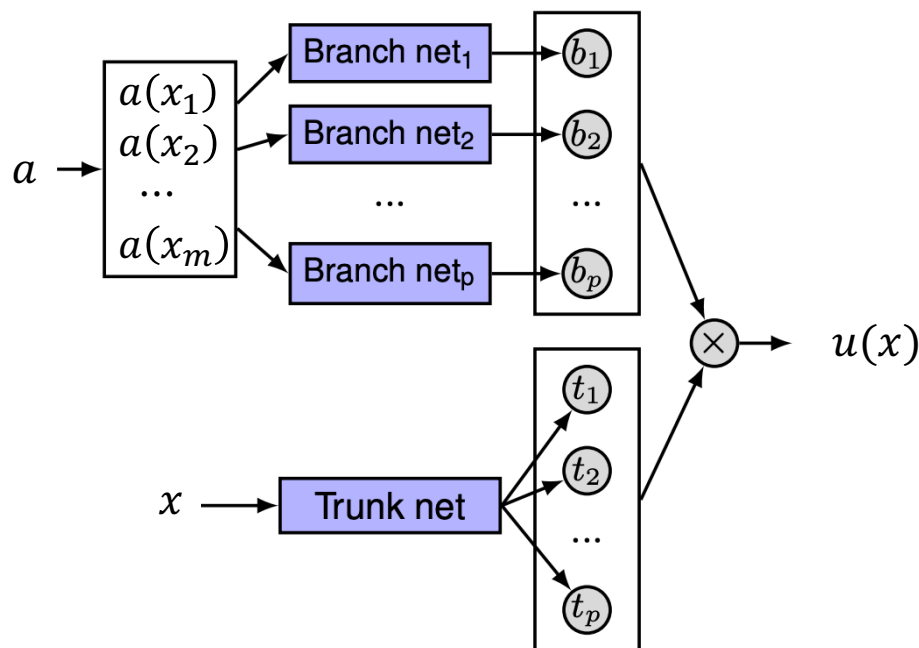
➤ 深度神经算子(Deep Operator Networks)

特征提取： $a(x) \rightarrow \hat{a} = [a(x_1), a(x_2), \dots, a(x_m)]$

DeepONet： $a(\cdot) \rightarrow u(\cdot) = \sum b_i(\hat{a})t_i(\cdot)$

b_i ：分支网络(branch network)

t_i ：主干网络(trunk network)





TRANSFORMER

➤ 注意力机制(Attention)

编码器状态 (encoder states) $x_1, x_2, \dots, x_n \in R^c$

解码器状态 (decoder states) $h \in R^o$

投影：

$$f = \langle x_1, h \rangle x_1 + \langle x_2, h \rangle x_2 + \dots + \langle x_n, h \rangle x_n$$

计算编码器状态 和解码状态的相关性：

$$\text{score}(h, x_i)$$

注意力权重：

$$a_i = \frac{\exp(\text{score}(h, x_i))}{\sum_{i=1}^n \exp(\text{score}(h, x_i))}$$

输出： $f = a_1 x_1 + a_2 x_2 + \dots + a_n x_n$



TRANSFORMER

➤ 自注意力机制(Attention)

编码器状态 (encoder states) $x_1, x_2, \dots, x_n \in R^c$

查询 (query): $q_i = W^q x_i$ ($W^q \in R^{m \times c}$)

键 (key): $k_i = W^k x_i$ ($W^k \in R^{m \times c}$)

值 (value): $v_i = W^v x_i$ ($W^v \in R^{m \times c}$)

相关性(alignment\score):

$$e_{ij} = \frac{q_i \cdot k_j}{\sqrt{d}} \quad a_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^n \exp(e_{ij'})}$$

输出 :

$$\text{Attention}(q, k, v)_i = \sum_{j=1}^n a_{ij} v_j$$



TRANSFORMER

➤ 自注意力机制(Attention)

编码器状态 (encoder states)

$$a = [a_1, a_2, \dots, a_c]: D \rightarrow R^c$$

查询 (query):

$$q = W^q a \quad (W^q \in R^{m \times c}): D \rightarrow R^m$$

键 (key):

$$k = W^k a \quad (W^k \in R^{m \times c}): D \rightarrow R^m$$

值 (value):

$$v = W^v a \quad (W^v \in R^{m \times c}): D \rightarrow R^m$$

计算每个格点之间的相关性

$$[a_1(x), a_2(x), \dots, a_c(x)]$$



TRANSFORMER

➤ 自注意力机制(Attention)

相关性(alignment\score):

$$e_{ij} = \frac{q(x_i) \cdot k(x_j)}{\sqrt{N}} \quad a_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^N \exp(e_{ij'})}$$

输出:

$$u(x_i) = \text{Attention}(q, k, v)_i = \sum_{j=1}^N a_{ij} v(x_j)$$

$$u(x) = \int \frac{\exp(q(x) \cdot k(y))}{\int \exp(q(x) \cdot k(z)) dz} v(y) dy$$

↑
系数

↑
基底



本堂课大纲

➤ 神经算子

- 对于一些问题能够高效、精确预测
- 需要大量数据训练
- 超参选取
- 预测仅限于内插（分布内数据）



参考文献

➤ 参考文献

基于卷积神经网络的代理模型：Zhu, Yinhao, and Nicholas Zabaras. "Bayesian deep convolutional encoder – decoder networks for surrogate modeling and uncertainty quantification." *Journal of Computational Physics* 366 (2018): 415-447.

提出U-net 的文章：Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *Medical image computing and computer-assisted intervention – MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III* 18, pp. 234-241. Springer International Publishing, 2015.

提出 PCA-Net 的文章：Hesthaven, Jan S., and Stefano Ubbiali. "Non-intrusive reduced order modeling of nonlinear problems using neural networks." *Journal of Computational Physics* 363 (2018): 55-78.

分析 PCA-Net 的文章：Bhattacharya, Kaushik, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. "Model reduction and neural networks for parametric PDEs." *The SMAI journal of computational mathematics* 7 (2021): 121-157.



参考文献

➤ 参考文献

提出 FNO 的文章 : Li, Zongyi, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. "Fourier neural operator for parametric partial differential equations." arXiv preprint arXiv:2010.08895 (2020).

用球谐函数做基底的文章 : Bonev, Boris, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. "Spherical fourier neural operators: Learning stable dynamics on the sphere." In International conference on machine learning, pp. 2806-2823. PMLR, 2023.

用小波做基底的文章 : Gupta, Gaurav, Xiongye Xiao, and Paul Bogdan. "Multiwavelet-based operator learning for differential equations." Advances in neural information processing systems 34 (2021): 24048-24062.

提出 DeepONet 的文章 : Lu, Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators." Nature machine intelligence 3, no. 3 (2021): 218-229.