论文阅读报告:
DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators

刘昭阳 25215133
中山大学数学学院 (珠海)

2025 年 12 月 22 日

摘要

本文是对论文《Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators》的深度阅读报告。该论文由 Lu Lu, Pengzhan Jin 和 George Em Karniadakis 等人撰写，提出了一种名为 DeepONet 的深度神经网络架构，旨在学习连续非线性算子。不同于传统的神经网络通常用于逼近有限维空间之间的函数映射，DeepONet 基于算子通用近似定理，能够学习从一个无限维函数空间到另一个无限维函数空间的映射。本文将详细阐述 DeepONet 的理论基础、网络架构、实验结果，并结合个人理解探讨其带来的启发以及未来的衍生想法。

目录

1 引言

在科学计算和工程领域，理解和模拟复杂系统的行为至关重要。许多物理系统，如流体力学、电磁学和量子力学系统，都由偏微分方程（PDEs）或积分方程描述。传统的数值方法（如有限元法、有限差分法）虽然精确，但在处理高维问题或需要实时预测的场景下，计算成本往往过高。

近年来，深度学习在函数逼近方面取得了巨大成功。然而，大多数现有的神经网络模型（如 CNN, RNN）主要设计用于学习有限维空间之间的映射（例如，图像分类中的像素到标签）。在物理建模中，我们往往需要学习的是一个“算子”（Operator），即从一个函数（如初始条件、边界条件或源项）到另一个函数（如系统解）的映射。

DeepONet 的提出旨在应对这一挑战。它不仅仅是针对某一个特定的方程解进行拟合，而是试图学习算子本身。一旦训练完成，DeepONet 可以针对任意新的输入函数快速给出系统的响应，而无需重新求解复杂的微分方程。这种“一次训练，多次使用”的特性，使其在代理模型（Surrogate Modeling）和实时控制中具有巨大的潜力。

2 相关工作

在 DeepONet 提出之前，已有许多尝试利用神经网络求解微分方程或学习算子的工作。其中最具代表性的是 Raissi 等人提出的 Physics-Informed Neural Networks (PINNs)。PINNs 通过将 PDE 残差加入损失函数，成功求解了正向和反向问题。然而，PINNs 主要针对单一特定的方程实例，一旦边界条件或初始条件改变，网络通常需要重新训练，这限制了其在实时应用中的效率。

除了 DeepONet，近年来还出现了 Fourier Neural Operator (FNO) 等神经算子方法。FNO 在频域内进行卷积操作，具有很好的分辨率无关性。与 FNO 相比，DeepONet 的理论基础更加直接地源于算子通用近似定理，且架构更加灵活，易于处理复杂几何形状。此外，DeepONet 也可以被视为一种非线性的降阶模型（Reduced Order Modeling, ROM）。传统的 ROM（如 POD, DMD）通过线性基函数投影来降低计算复杂度，而 DeepONet 的 Branch Net 学习的是非线性特征系数，Trunk Net 学习的是非线性基函数，从而能够捕捉更复杂的系统动力学特征。

3 DeepONet 方法论

3.1 理论基础：算子通用近似定理

DeepONet 的核心理论依据是 Chen & Chen 在 1995 年提出的算子通用近似定理（Universal Approximation Theorem for Operators）。该定理指出，对于任意非线性连续算子 G，都可以用一个单隐层的神经网络以任意精度逼近。

定理 1 (算子通用近似定理): 假设 σ 是一个连续非多项式函数，X 是一个 Banach 空间，$K_1 \subset X$ 和 $K_2 \subset \mathbb{R}^d$ 是紧集。V 是 $C(K_1)$ 中的紧集。$G : V \to C(K_2)$ 是一个连续非线性算子。那么对于任意 $\epsilon > 0$，存在正整数 n, p, m，以及常数 $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k, w_k \in \mathbb{R}$，使得对于任意 $u \in V$ 和 $y \in K_2$，有：

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

DeepONet 的架构正是基于此定理的简化形式，其中 Branch Net 逼近输入函数 u 的泛函部分，Trunk Net 逼近输出域 y 的函数部分。

3.2 网络架构

基于上述定理，DeepONet 设计了独特的双塔结构，分为“未堆叠”(Unstacked) 和“堆叠”(Stacked) 两种变体。最常用的是未堆叠结构，示意如图 1 所示：

1. **Branch Net**: 接收输入函数 u 在 m 个固定传感器位置 $\{x_1, x_2, \ldots, x_m\}$ 上的观测值 $[u(x_1), u(x_2), \ldots, u(x_m)]$ 作为输入。其输出为一组系数 $[b_1, b_2, \ldots, b_p]$。

2. **Trunk Net**: 接收输出函数评估点 y 的坐标作为输入。其输出为一组基函数值 $[t_1, t_2, \ldots, t_p]$。

   最终的预测输出 $G(u)(y)$ 是这两个网络输出的点积（Dot Product）：

$$G(u)(y) \approx \sum_{k=1}^{p} b_k \cdot t_k + b_0 \tag{2}$$

这种架构巧妙地将输入函数的特征提取与输出域的空间依赖性解耦，极大地提高了模型的表达能力和泛化能力。通常会在最后加上一个偏置项 b_0 以提高拟合能力。

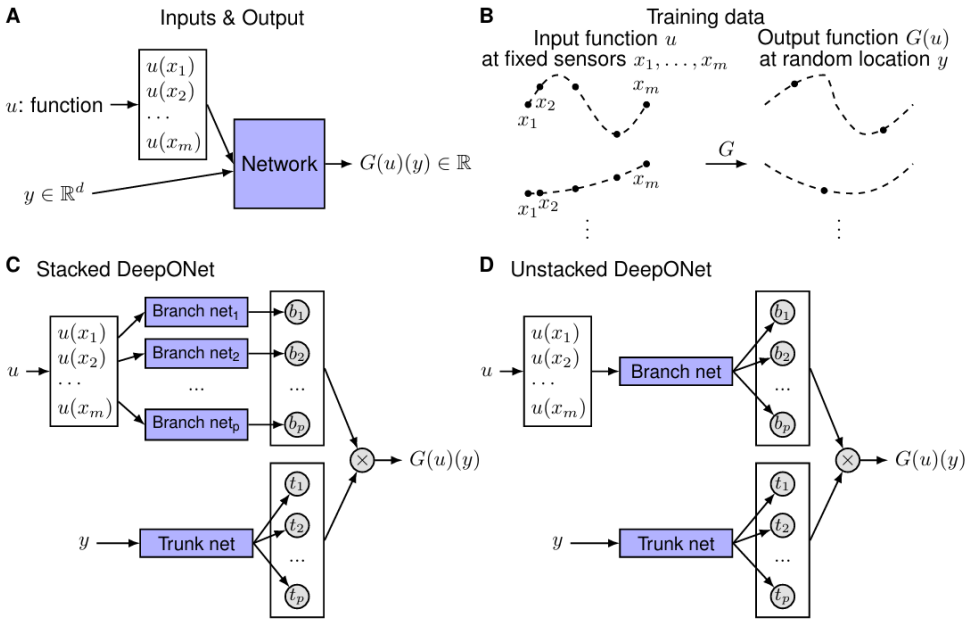

图 1: DeepONet 架构示意图，包含输入/输出、训练数据生成流程，以及未堆叠与堆叠两种实现。

3.3 数据生成与训练

为了训练 DeepONet，需要生成大量的 (输入函数, 输出函数) 对。

- **输入函数空间采样**: 通常使用高斯随机场（Gaussian Random Fields, GRF）来生成多样化的输入函数 $u(x)$，以覆盖感兴趣的函数空间。GRF 的协方差核函数通常定义为径向基函数（RBF）：

$$k(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||^2}{2l^2}\right) \tag{3}$$

  其中 l 是相关长度参数，控制生成函数的平滑度。

- **标签生成**: 对于每一个生成的 $u(x)$，使用高精度的传统数值求解器（如 FEM, Spectral Methods）求解对应的微分方程，得到真实的解 $G(u)(y)$ 作为标签。

- **损失函数**: 通常使用均方误差（MSE）作为损失函数，最小化预测值与真实值在训练点上的差异。

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{P}|G(u^{(i)})(y_j) - \hat{G}(u^{(i)})(y_j)|^2 + \lambda||\Theta||^2$$

  其中 λ 是正则化参数，用于防止过拟合。

4 实验结果与分析

论文中进行了一系列系统性的实验，涵盖了从简单的常微分方程（ODEs）到复杂的偏微分方程（PDEs）。

4.1 动态系统与积分算子

在 1D 问题中，DeepONet 展示了极高的精度。首先是积分算子的学习，即 $G(u)(x) = \int_0^x u(t)dt$。尽管这是一个线性算子，但它验证了 DeepONet 能够准确捕捉积分的累积效应。其次，论文研究了重力摆 (Gravity Pendulum) 系统，考虑如下非线性 ODE：

$$\frac{d^2s}{dt^2} + \sin(s) = u(t), \quad s(0) = \frac{ds}{dt}(0) = 0 \tag{4}$$

其中 $u(t)$ 是外力。DeepONet 的任务是学习从外力 $u(t)$ 到位移 $s(t)$ 的映射。作者对 $u(t)$ 做密集时间采样（数量级约 10^2 的传感器点），并在大量不同外力轨迹上训练，结果表明模型不仅能拟合训练数据，还能较好泛化到未见过的外力函数。

4.2 偏微分方程 (PDEs)

在更复杂的 2D PDE 问题中，DeepONet 同样表现出色。以 Darcy Flow 为例，模型学习从渗透率场 $K(x)$ 到压力场 $P(x)$ 的映射。方程为：

$$-\nabla \cdot (K(x)\nabla P(x)) = f(x), \quad x \in (0,1)^2 \tag{5}$$

这是一个典型的非线性算子学习问题。实验显示，DeepONet 能够处理具有剧烈变化的渗透率场，并准确预测压力分布。此外，在对流扩散方程 (Advection-Diffusion) 中，DeepONet 也成功学习了不同源项下的浓度分布。方程形式为：

$$\frac{\partial s}{\partial t} + v\frac{\partial s}{\partial x} = D\frac{\partial^2 s}{\partial x^2} \tag{6}$$

结果证明 DeepONet 能够准确捕捉波的传播和扩散过程。

4.3 对比实验

论文将 DeepONet 与传统的全连接神经网络（FNN）进行了对比。在 FNN 中，输入通常是将 $u(x)$ 和 y 拼接在一起的向量。实验结果显示，在相同的参数量下，DeepONet 的测试误差比 FNN 低一个到两个数量级。这表明 DeepONet 的架构引入了适合算子学习的归纳偏置（Inductive Bias），即算子的输出可以分解为输入相关系数与空间基函数的乘积。

4.4 误差收敛性分析

论文不仅展示了实验结果，还提供了理论上的误差界。实验观察到，DeepONet 的测试误差随着训练样本数量的增加呈现多项式甚至指数级的下降。此外，误差还与传感器数量 m 有关，更多的传感器能提供更丰富的输入函数信息，从而降低近似误差。

5 个人启发

阅读这篇论文给我带来了多方面的启发，不仅限于具体的算法细节，更在于解决问题的思维方式。

5.1 架构即先验 (Architecture as Prior)

DeepONet 的成功再次证明了神经网络架构设计的重要性。通用的 MLP 虽然理论上也是通用近似器，但在实际学习算子时效率低下。DeepONet 通过显式地构造 Branch 和 Trunk 两个分支，实际上是强加了一种物理上的先验知识：即算子的输出可以看作是输入函数特征与空间基函数的线性组合。这种结构与分离变量法、谱方法等传统数学物理方法有着深刻的同构性。这启示我们在设计 AI for Science 模型时，不应盲目堆砌层数，而应深入思考问题的数学结构，将其融入网络设计中。

5.2 从“解方程”到“学算子”的范式转变

传统的科学计算关注于求解单个方程实例。而 DeepONet 代表了一种范式转变：从求解单一实例转向学习方程背后的算子规律。这种转变类似于从“死记硬背”到“掌握规律”。虽然训练过程可能昂贵（需要大量数据），但一旦学会，推理成本极低。这对于需要反复求解同一类方程的反问题优化、不确定性量化等任务具有革命性意义。

5.3 数据驱动与物理信息的互补

虽然 DeepONet 主要是一个数据驱动的方法，但它并不排斥物理信息。论文中提到的数据生成过程依赖于传统求解器，这本身就是一种物理知识的注入。此外，DeepONet 的框架很容易扩展为 Physics-Informed DeepONet，即在损失函数中加入 PDE 残差项，从而减少对标签数据的依赖。这种混合建模的思路是未来科学智能（AI4S）发展的主流方向。

6 衍生想法

基于 DeepONet 的框架，我认为可以在以下几个方向进行深入探索和改进：

6.1 1. 架构层面的改进

首先是 Attention 机制的引入。目前的 Branch Net 通常使用 MLP 或 CNN，但对于复杂的输入函数，不同区域的重要性可能不同。引入 Self-Attention 机制（如 Transformer Encoder）作为 Branch Net，可以更好地捕捉输入函数的长程依赖和局部奇异性。例如，在流体力学中，激波位置附近的输入特征对全局流场影响巨大，Attention 机制可以自动聚焦于这些区域。

其次是 Trunk Net 的基函数优化。Trunk Net 负责学习空间基函数，可以尝试使用傅里叶特征映射（Fourier Feature Mapping）或 SIREN（Sinusoidal Representation Networks）作为 Trunk Net，以增强模型对高频细节和导数信息的拟合能力。这对于求解高波数的波动方程尤为重要。

最后是自适应传感器布局。目前的传感器位置 x_i 通常是固定的。可以设计一种机制，让网络在训练过程中自动学习最优的传感器位置，或者针对不同的输入函数动态调整采样点，从而提高采样效率。

6.2 2. 训练策略与数据效率

在训练策略方面，可以引入课程学习 (Curriculum Learning)。算子学习的难度往往与输入函数的复杂度和 PDE 的非线性程度有关。可以设计课程学习策略，先让网络学习平滑、简单的函数映射，然后逐渐增加输入函数的频率和幅度，引导网络逐步掌握复杂算子。

此外，考虑到数据生成代价高昂，可以使用主动学习 (Active Learning) 策略。通过评估模型在未标注数据上的不确定性，有针对性地生成那些模型“最困惑”的样本进行标注，从而以最少的数据量达到最大的性能提升。

6.3 3. 应用场景拓展

DeepONet 的应用场景可以进一步拓展到多物理场耦合。现实中的物理问题往往涉及多个物理场的耦合（如流固耦合、热流耦合）。可以设计 Multi-Branch DeepONet，分别编码不同的物理场输入，然后在 Trunk Net 端进行融合，实现多物理场系统的快速模拟。

在反问题求解方面，DeepONet 不仅可以用于正向预测，也可以用于反问题。例如，已知系统响应 $G(u)$，求解输入 u。可以通过冻结训练好的 DeepONet 参数，将输入 u 作为可优化变量，通过梯度下降反向求解。或者直接训练一个 Inverse DeepONet，交换 Branch 和 Trunk 的角色。

最后是几何泛化。原始 DeepONet 主要处理规则区域。对于复杂几何形状，可以将几何信息（如 SDF, Signed Distance Function）也作为 Branch Net 的输入，或者使用 Graph Neural Networks (GNN) 作为 Trunk Net 来处理非结构化网格，从而实现对不同几何形状的泛化能力。

7 局限与注意事项

尽管 DeepONet 表现出强大的算子学习能力，实践中仍需注意以下局限：

- **传感器依赖**: Branch Net 依赖固定的采样点 x_i，当测试时的输入函数具有新的局部奇异或分辨率需求时，性能可能下降。
- **分布外泛化**: 训练集通常来自有限的随机场或参数分布，超出分布的输入函数（更高频或更剧烈的非线性）会显著降低精度。
- **边界/物理约束**: 基础 DeepONet 未显式编码 PDE 约束，若标签数据稀缺，建议结合 Physics-Informed 损失以提升稳定性。
- **高维输出成本**: 对高维 y 域或长时间演化问题，Trunk Net 所需的基函数数目 p 可能迅速增大，导致内存与训练时间开销增加。

8 结论

DeepONet 作为一种基于算子通用近似定理的深度学习框架，为非线性算子的学习提供了一种高效、通用的解决方案。它不仅在理论上具有坚实的基础，在实验中也展现出了优越的性能和泛化能力。通过将输入函数编码与输出空间表示解耦，DeepONet 成功地打破了传统神经网络在函数空间映射上的局限。

本文详细回顾了 DeepONet 的原理与贡献，并结合个人思考提出了若干改进方向。随着 AI 与科学计算的深度融合，相信 DeepONet 及其衍生变体将在流体力学、材料科学、气候预测等领域发挥越来越重要的作用。它不仅是一个高效的计算工具，更是连接数据科学与物理定律的一座桥梁。