Data Layout and Differentiation for Efficient Training of Physics-Informed Deep Operator Networks

Junya Onishi[1], **Tokiya Tanaka**[2], **Ayato Takii**[2], **Makoto Tsubokura**[1,2]

[1]RIKEN Center for Computational Science, [2]Kobe University

Introduction

- Engineering applications such as indoor ventilation design require surrogate models that can rapidly predict solutions over a wide range of boundary and operating conditions. Deep Operator Networks (DeepONets) [1] provide such models by learning mappings from input functions to solution fields.
- Physics-informed DeepONets (PI-DeepONets) [2,3] reduce the need for simulation-generated training data by incorporating governing equations into the loss. However, their training requires higher-order spatial derivatives at many function-coordinate pairs, resulting in substantial computational and memory costs.
- This work investigates how data layout and spatial differentiation affect the training efficiency of PI-DeepONets. To address the computational and memory bottlenecks, we develop a compact training formulation that exploits the tensor-product structure of PI-DeepONets.

PI-DeepONet

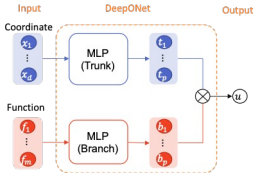
- A PI-DeepONet maps an input function $f(y)$ and a coordinate $\boldsymbol{x}$ to the solution $u(f,\boldsymbol{x})$ through the contraction of Branch and Trunk latent features. The Branch network encodes $f(y)$ into Branch features while the Trunk network encodes $\boldsymbol{x}$ into Trunk features.

Ex) Input function: Inlet velocity distribution

Solution: Corresponding flow velocity and pressure

$$u(f,\boldsymbol{x}) = \sum_{k=1}^{p} b_k(f)\, t_k(\boldsymbol{x})$$

b_k : Branch feature
t_k : Trunk feature
d : Number of spatial dimensions
m : Number of sensors
p : Number of features
N_x : Number of spatial coordinate points
N_y : Number of input functions



Data Layout

- In the training of N_x coordinates and N_f input functions, the original PI-DeepONet formulation [3] represents training data as $N_x N_f$ flattened function-coordinate pairs. This layout duplicates each Branch input N_x times and each Trunk coordinate N_f times.
- To address this issue, we introduce a compact layout that retains the Branch inputs and Trunk coordinates separately. As a result, the Branch and Trunk networks are evaluated only at their unique inputs, and all $N_x N_f$ predictions can be computed with a single GEMM.

Differentiation

- Physics-informed training additionally requires spatial derivatives of the predicted solution with respect to the coordinate inputs to evaluate PDE residuals.
- In the flattened layout these spatial derivatives are evaluated over $N_x N_f$ duplicated function-coordinate pairs. In the compact layout the Trunk features are differentiated only at the N_x unique coordinates, and the resulting derivatives are reused across all N_f input functions.
- Because $d \ll p$, computing the spatial derivatives of all Trunk features by direct reverse-mode AD requires p sweeps and can be less efficient than forward-mode AD, which requires d sweeps.
- To address this issue, we evaluate the following methods:
  - Forward-mode AD: d input-direction sweeps
  - ZCS: Reverse-mode AD with scalar coordinate shifts [4]
  - Explicit propagation: Layer-wise propagation of spatial derivatives using analytical MLP formulas (bypass generic AD engine overhead and reduces CUDA kernel launches)
- To compute gradients with respect to network parameters, reverse-mode AD is used in all the above methods.

Performance evaluation

- We benchmark physics-only PI-DeepONet training for two-dimensional steady incompressible Navier–Stokes flows with varying inlet boundary conditions [5].
- Four implementations are compared to evaluate the effects of data layout and spatial differentiation in the following conditions:
  - Systematic sweep over N_x, N_f, p (60 configurations)
  - FP32 on NVIDIA A100

Implementation	Data layout	Contraction	Spatial differentiation
Flattened Base	$N_x N_f$ duplicated pairs	Pairwise inner products	Reverse-mode AD
Forward-mode AD	Compact	GEMM	Forward-mode AD (Generic)
ZCS	Compact	GEMM	Reverse-mode AD with Zero coordinate shift
Explicit propagation	Compact	GEMM	Layerwise propagation of spatial derivatives

- Flattened Base failed in the largest cases because of GPU out-of-memory (OOM) errors, whereas all the compact implementations completed every configuration with substantially lower GPU memory consumption.
- Explicit propagation achieved a 144× memory reduction and a 4.87× speedup, compared to Flattened Base.
  - $N_x = 2500,\ N_f = 64,\ p = 32$

3D Demonstration

- Ventilation optimization in a cubic domain
  - Restriction: Fixed exhaust opening (red outline)
  - Design variable: Velocity distribution at the inlet
  - Objective: Minimize the age of air
- The proposed memory-efficient formulation enabled practical 3D PI-DeepONet training.
  - After a single training run, the PI-DeepONet produced flow fields that agreed well with OpenFOAM reference solutions for multiple inlet velocity distributions.
  - The resulting surrogate enabled dense evaluation of the age of air across the inlet-distribution space.

Conclusion

- The compact PI-DeepONet training substantially reduces training time and GPU memory consumption while preserving numerical equivalence, making practical three-dimensional training possible. Two elements were essential:
  - Eliminating duplicated function–coordinate data through compact data layout and GEMM-based contraction, and
  - Efficiently computing and reusing Trunk spatial derivatives using an appropriate differentiation method.
- ZCS provided a fast and general differentiation method, while Explicit derivative propagation achieved the best overall performance by exploiting the known MLP architecture of the Trunk network considered in this study.
- The same formulation may be extended to other neural operators with separable solution representations, enabling larger training problems and more complex three-dimensional applications.

References

[1] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. 2021. "Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators." Nature Machine Intelligence 3 (3): 218–29.

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis. 2019. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." Journal of Computational Physics 378 (February): 686–707.

[3] S. Wang, H. Wang, and P. Perdikaris. 2021. "Learning the Solution Operator of Parametric Partial Differential Equations with Physics-Informed DeepONets." Science Advances 7 (40): eabi8605.

[4] K. Leng, M. Shankar, and J. Thiyagalingam. 2024. "Zero Coordinate Shift: Whetted Automatic Differentiation for Physics-Informed Operator Learning." Journal of Computational Physics 505 (112904): 112904.

[5] J. Onishi, H. Kitagawa, R. Puri, M. Rüttgers, R. Sarma, A. Lintermann, and M. Tsubokura. 2026. "Surrogate Modeling of Fluid Flow under Different Conditions Using Physics-Informed Deep Operator Networks." Computers & Fluids 316 (107154): 107154.

Acknowledgement

This work was partly supported by JSPS KAKENHI Grant Number JP25K15154.

