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

Junya Onishi
RIKEN Center for Computational Science
RIKEN
Kobe, Japan
junya.onishi@riken.jp

Tokiya Tanaka
Graduate School of System Informatics
Kobe University
Kobe, Japan

Ayato Takii
Graduate School of System Informatics
Kobe University
Kobe, Japan

Makoto Tsubokura
Graduate School of System Informatics
Kobe University
Kobe, Japan

I. 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.

Operator-learning methods, including Deep Operator Networks (DeepONets) [1], have attracted increasing attention because they learn mappings from input functions to solution fields. However, training data for these models are commonly generated through a large number of numerical simulations, resulting in substantial computational cost. Physics-informed approaches [2], [3] reduce their dependence on simulation data by incorporating governing equations into training loss. Their training, however, requires higher-order spatial derivatives at a large number of function–coordinate pairs, resulting in substantial computational and memory costs.

In this work, we reorganize the PI-DeepONet training and investigate the effects of data representation, differentiation strategy, and tensor contraction on training performance and memory consumption.

II. Methodology

A. Data Layout

A PI-DeepONet takes an input function f and a coordinate $\boldsymbol{x}$ as inputs. The Branch network encodes f, represented by its values at m sensors, into a feature vector $\boldsymbol{b}(f) \in \mathbb{R}^p$, while the Trunk network maps the d-dimensional coordinate $\boldsymbol{x}$ to a feature vector $\boldsymbol{t}(\boldsymbol{x}) \in \mathbb{R}^p$. The solution value is represented through their contraction, $u(f, \boldsymbol{x}) = \sum_{k=1}^{p} b_k(f) t_k(\boldsymbol{x})$, where p denotes the number of latent features.

In the training of N_x coordinates and N_f input functions, the original PI-DeepONet implementation [2] flattens their Cartesian product into $N_x N_f$ function–coordinate pairs. This layout repeats each Branch input at N_x coordinates and each Trunk coordinate for N_f input functions. It also constructs the automatic-differentiation graph on the duplicated pairs.

We instead retain the Branch inputs and Trunk coordinates separately as $V \in \mathbb{R}^{N_f \times m}$ and $X \in \mathbb{R}^{N_x \times d}$, respectively. The Branch and Trunk networks are evaluated only at their unique inputs, producing the feature matrices $B \in \mathbb{R}^{N_f \times p}$ and $T \in \mathbb{R}^{N_x \times p}$, respectively.

B. GEMM-Based Contraction

In the flattened layout, a Branch–Trunk inner product is evaluated separately for each of the $N_x N_f$ pairs. The compact layout instead assembles all predictions with a single GEMM as $U = TB^{\mathsf{T}} \in \mathbb{R}^{N_x \times N_f}$. Moreover, because B is independent of the coordinates, derivatives have the same matrix form as $U_x = T_x B^{\mathsf{T}}$ and $U_{xx} = T_{xx} B^{\mathsf{T}}$.

C. Differentiation

Physics-informed training requires spatial derivatives to evaluate PDE residuals. In the compact layout, the Trunk features and their spatial derivatives are evaluated only at the unique coordinates and shared among all the input functions through GEMM-based contraction.

The Trunk network maps d coordinate variables to p latent features, where $d \ll p$. Direct reverse-mode differentiation is therefore unfavorable because the required sweeps depend on the number of outputs p. We compare three approaches:

- **Forward-mode AD**: generic forward-mode differentiation provided by PyTorch, requiring sweeps over the d input directions;
- **ZCS**: reformulation of coordinate derivatives using zero coordinate shifts, allowing reverse-mode AD without p-dependent sweeps [4]; and
- **Explicit propagation**: layerwise propagation of first and second derivatives through the Trunk MLP with smooth activations.

Explicit propagation is mathematically equivalent to Forward-mode AD for the supported MLP but avoids the

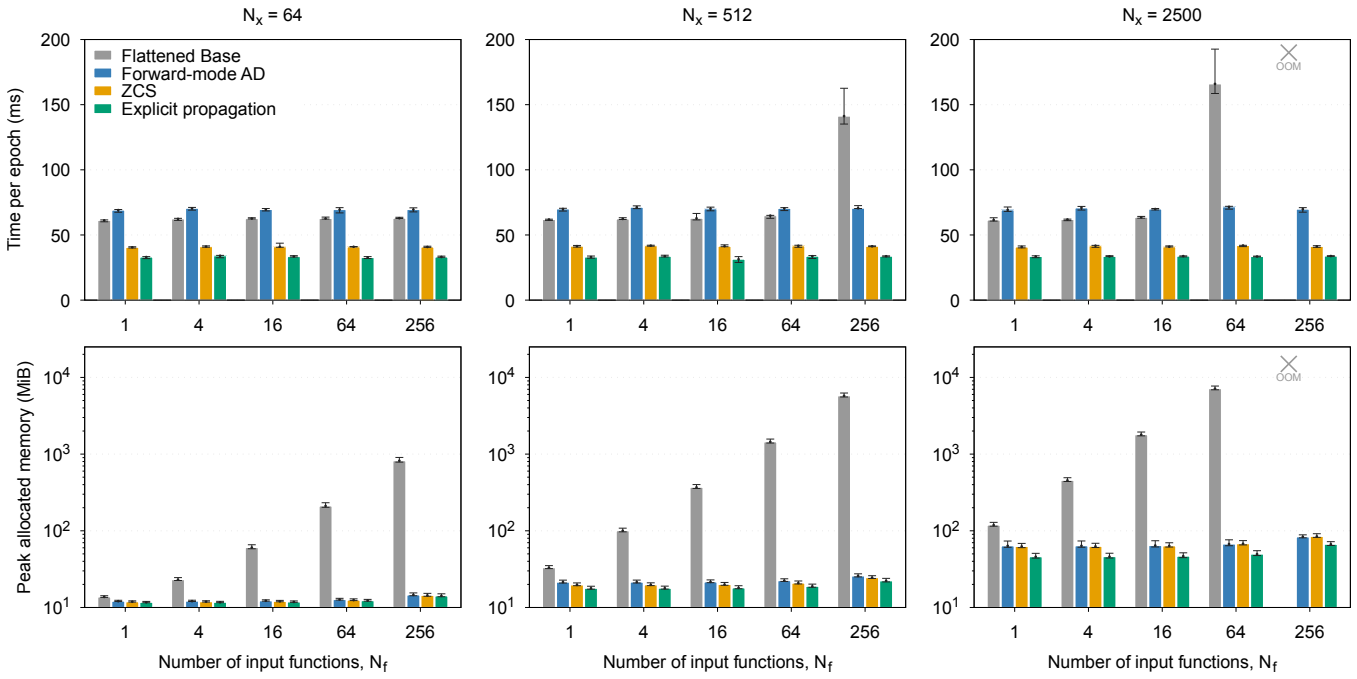


Fig. 1. Time per epoch and peak allocated GPU memory for the complete parameter sweep over N_x, N_f, and p. Bars represent the median over $p = 16, 32, 64,$ and 128, and error bars indicate the corresponding minimum–maximum range. The symbol "OOM" denotes an out-of-memory failure.

overhead of generic AD operations. All three approaches use reverse-mode AD to compute gradients with respect to the trainable network parameters.

III. Performance Evaluation

A. Problem Setting

We consider physics-only PI-DeepONet training for two-dimensional steady incompressible Navier–Stokes flows under varying inlet boundary conditions [5].

The Flattened Base is compared with three compact implementations employing Forward-mode AD, ZCS, and Explicit propagation, respectively. A systematic sweep over N_f, N_x, and p comprises 60 configurations per implementation. Performance measurements use FP32 on an NVIDIA A100 and consist of 10 warm-up epochs followed by 50 measured epochs.

B. Results

Figure 1 summarizes the performance of the four implementations. All three compact implementations completed all 60 configurations, whereas the Flattened Base failed in the four largest cases because of insufficient GPU memory.

Increasing N_f and N_x enlarged the number of function–coordinate pairs without changing the network architecture. The Flattened Base exhibited rapidly increasing memory consumption and eventually longer epoch times because its network evaluations and differentiation graphs were duplicated over the pairs. In contrast, the compact implementations greatly reduced memory consumption, enabling training for three-dimensional fluid flows. Their time per epoch also remained nearly constant as N_f and N_x increased. CUDA profiling showed that the smaller 2D workloads underutilized the GPU, leaving sufficient device capacity to absorb the increased tensor sizes within the tested range.

Increasing p moderately increased the epoch time of the Flattened Base, whereas the compact implementations showed little sensitivity to p. Notably, ZCS avoided p-dependent reverse sweeps despite using reverse-mode AD.

For the representative case $N_f = 64$, $N_x = 2500$, and $p = 32$, Explicit propagation was $4.87\times$ faster than the Flattened Base and reduced peak memory by a factor of 144. Among the compact implementations, CUDA profiling suggested that the shorter times per epoch of ZCS and Explicit propagation relative to Forward-mode AD were primarily attributable to fewer kernel launches.

IV. Conclusion

Compact data layout eliminates duplicated function–coordinate data, enables GEMM-based contraction, and shares Trunk derivatives among input functions. Explicit propagation achieved the shortest time per epoch and lowest peak memory among the tested implementations. The reduced memory footprint enabled practical three-dimensional PI-DeepONet training. The underlying strategy may also be applicable to other neural operators with separable solution representations.

Acknowledgment

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

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.