Performance Evaluation 170 (2025) 102530

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

swPredictor: A data-driven performance model for distributed data parallelism training on large-scale HPC clusters

Xianyu Zhu [a], Ruohan Wu [a], Junshi Chen [a,b,*], Hong An [a,b]

[a] University of Science and Technology of China, Hefei, Anhui, China
[b] Laoshan Laboratory, Qingdao, China

ARTICLE INFO

Keywords:
High-performance computing
Performance modeling
Deep learning
Distributed training

ABSTRACT

Given the complexity of heterogeneous architectures and multi-node collaboration, large-scale HPC (High-Performance Computing) clusters pose challenges in resource utilization and performance optimization during distributed data parallelism (DDP) training. Performance modeling aims to identify application bottlenecks and guide algorithm design, but existing performance models rarely consider the impact of system architecture on communication performance or provide a systematic analysis of distributed training. To address these issues, this paper proposes swPredictor, a data-driven performance model devised for accurately predicting the performance of DDP training. First, an original performance dataset is developed based on various communication patterns at runtime to avoid systematic errors. Subsequently, a novel multi-branch module FNO-Inception is proposed, combining FNO (Fourier Neural Operator) layer with Inception structure to simultaneously utilize various frequency features. Finally, by introducing the FNO-Inception module, a novel regression model FI-Net is constructed to fit complex nonlinear relationships. The experimental results demonstrate that FI-Net can accurately predict the performance of DDP training on the Sunway OceanLight supercomputer with an overall MAPE of 0.93%, which outperforms the other baseline models.

1. Introduction

The integration of high-performance computing (HPC) and artificial intelligence (AI) enables large-scale cluster resources to accelerate AI applications in scientific research, such as weather forecasting [1], bioinformatics [2], materials modeling [3] and quantum mechanics [4]. Distributed data parallelism (DDP) is the most commonly used approach to accelerate AI training, which relies on collaborative computation across distributed nodes to improve the training efficiency. However, the complex system architecture and diverse communication patterns of large-scale HPC clusters bring challenges in algorithm performance improvement and resource management [5]. In order to improve the performance of DDP training, it is necessary to identify and optimize the application bottlenecks of computation and communication [6]. Meanwhile, reasonable resource scheduling requires the accurate estimation of the algorithm performance, which can improve resource allocation and overall system efficiency in large-scale clusters. Moreover, many AI model design approaches [7–9] combine the model performance metrics, such as latency or FLOPS, to construct a more efficient network. Performance modeling aims to analyze and predict an algorithm's performance by examining its key governing parameters. This methodology leads to more efficient resource utilization, more insightful algorithm design, and lower development costs.

* Corresponding author.
E-mail addresses: zhuxy@mail.ustc.edu.cn (X. Zhu), ruohanwu@mail.ustc.edu.cn (R. Wu), cjuns@ustc.edu.cn (J. Chen), han@ustc.edu.cn (H. An).

https://doi.org/10.1016/j.peva.2025.102530
Received 28 June 2025; Received in revised form 19 September 2025; Accepted 7 November 2025
Available online 11 November 2025

Existing performance modeling approaches can generally be classified into analytical modeling and data-driven modeling [10,11]. The analytical modeling method evaluates the algorithm performance through the theoretical analysis models, such as queueing theory [12] and stochastic model [13]. When facing the complex system and diverse algorithms, it requires substantial costs in constructing effective mathematical models. Since the model is designed specifically to describe a particular system or algorithm, its applicability is rather limited. With the advantages of wider applicability and lower costs, the data-driven modeling method has become popular. It utilizes collected extensive data to train AI models that can automatically fit the complex nonlinear relationships between algorithm parameters and performance [14].

Many data-driven modeling methods [15,16] focus on devising more complex models to fit the relationships between algorithm parameters and performance within the dataset. However, they often overlook the impact of runtime system topology on the algorithm performance. For example, though the number of allocated computational nodes is the same, different scheduling methods and resource usage conditions will make the physical nodes different, leading to different communication bandwidth and latency [17]. Moreover, many performance models are only built for model inference [18,19], lacking analysis of backward propagation, parameter update and distributed communication. To aid in the resource management and AI model design on large-scale HPC clusters, there is a lack of performance models which support the systematic analysis and prediction of DDP training.

To address these issues, the paper proposes a data-driven performance model **swPredictor** for accurately predicting the performance of DDP training on large-scale HPC clusters. First, we develop a performance dataset, which includes the operator performance dataset and communication performance dataset. The operator performance dataset encompasses fourteen commonly used operators from thirteen types of models. The communication performance dataset is built according to the multiple communication patterns at runtime, which can avoid systematic error and adapt to different network topologies. To avoid extensive experiments to select the optimal FNO (Fourier Neural Operator) mode, we construct a new multi-branch module FNO-Inception, which can simultaneously use multiple modes and capture various frequency features. Furthermore, by introducing the FNO-Inception module, the regression prediction model FI-Net is constructed to fit the complex nonlinear relationships between algorithm parameters and performance. The experimental results on predicting the operator execution time, demonstrate that FI-Net outperforms the other models in the prediction of forward propagation, backward propagation, and parameter update, with MAPE of 1.21%, 0.95% and 0.77% respectively. On the other hand, in the experiment predicting the communication time with different model parameters, FI-Net presents strong performance with an average MAPE of 4.04%. Finally, the overall experiment results present that FI-Net can accurately predict the time of DDP training on the Sunway OceanLight supercomputer with an overall MAPE of 0.93%. Besides, the ablation experiment results show that the proposed multi-branch module namely FNO-Inception can improve the accuracy of the model compared to using a single FNO mode.

The main contributions of the paper are as follows:

- Construct an original performance dataset, eliminating the systematic predicted error and adapting to different network topologies on large-scale HPC clusters.
- Propose a novel multi-branch module namely FNO-Inception to avoid experimental selection of optimal FNO mode and capture various frequency features.
- Build a new regression model FI-Net, fitting the complex nonlinear relationships between algorithm parameters and performance within the heterogeneous system.

The paper is organized as follows: Section 2 introduces the related work and the Sunway Accelerate Computing Architecture; Section 3 describes the details about swPredictor; The results and analysis of the experiment are discussed in Section 4; Section 5 summarizes the paper and proposes the future work.

2. Background

The section introduces the recent advances in analytical modeling and data-driven modeling. Section 2.1 reviews existing work in computation and communication, the two primary areas addressed by the performance model presented in this paper. Besides, Section 2.2 subsequently presents the system architecture on Sunway OceanLight supercomputer.

2.1. Performance modeling

2.1.1. Analytical model

Analytical modeling combines theoretical analysis with mathematical modeling to analyze the computational complexity and memory access patterns of the program, helping identify potential performance bottlenecks and optimization opportunities. In order to make better utilization of hardware resources, a microbenchmark-based performance model is developed to predict the performance of NVIDIA GeForce 200-series GPUs, with an error of only 5~15% [20]. Sabela et al. [21] provide a typical performance model on CPU to assess the utilization of memory and help automatically optimize the OpenMP and MPI applications. Chen et al. [22] combine the calculation pattern of operators and the architecture of the Tianhe-3 supercomputer, analyzing the Roofline model and bottlenecks of convolutional neural networks. To design a more efficient deep learning library, Fang et al. [23] derive a multi-level performance model and apply a series of optimization schemes by analyzing the convolutional neural network and the Sunway processor architecture.

By analyzing the communication algorithm, the analytical modeling can construct mathematical formulas to calculate the performance of communication. The Hockney model [24] and the $LogP$ model [25] are classical analytical models that predict

communication latency by describing the message-passing startup time, transmission rate and parallelism. Emin et al. [26] construct an analytical model for `MPI_Bcast` routines and improve the accuracy significantly, which can help to select the optimal algorithm for one-process-per-core applications. Leveraging an analytical model to predict communication performance under varying parameters, Michael et al. [27] design novel collective algorithms for exascale supercomputers that deliver a 4.5× acceleration compared to the baseline. According to the network topology of the Sunway OceanLight, Chen et al. [28] propose a hybrid approach to optimize point-to-point and collective communication through in-network computation. Meanwhile, a cost model is established to assess the performance of communication.

However, when predicting DDP performance, the large variety of deep learning operators and computational patterns makes analytical modeling highly labor-intensive. Meanwhile, the wide variability of algorithm parameters imposes inherent limitations on the accuracy of mathematical models.

2.1.2. Data-driven model

Instead of relying on detailed analysis of the algorithm, data-driven modeling takes advantage of machine learning models to predict the algorithm performance. Kaufman et al. [29] develop a learned performance model for tensor programs with high precision, which helps the autotuner discover program settings faster. To aid in the design of high-performance machine learning architecture on specific computing platforms, a data-driven model ResPerfNet [30] is proposed to predict the performance of DNN, achieving an MAPE of 8.4%. For accurately evaluating the computational cost of deep neural networks, a lightweight prediction approach, DNNAbacus [31] is proposed to predict the time and memory cost for 29 types of classic models. The mean relative error of DNNAbacus is only 0.9% for time and 2.8% for memory, making it more accurate than state-of-the-art methods. By combining kernel detection and adaptive sampling, nn-Meter [16] is built to predict the inference latency of DNNs on edge devices, significantly outperforming the other methods. In order to discover the bottleneck of HPC application on the Sunway supercomputer and solve the low accuracy of existing performance models, Zhang et al. [32] propose a new performance modeling method based on the Fourier neural operator, whose prediction error is less than 10%.

Based on substantial data, the data-driven modeling can adapt to complex network conditions and communication algorithms, providing higher accuracy than traditional analytical models. The classification models including the decision tree and random forest, are utilized to predict the optimal algorithm for MPI collective communication [33]. This prediction can be represented as "$f(c, m, n, ppn) \rightarrow alg\_id$" and the experimental results show the approach outperforms the default configuration of the standard MPI library. To predict the collective communication performance in the MPI library and find the best function parameters, Hunold et al. [34] develop an auto-tuning approach using several machine learning models. The experiment shows that the approach can accurately predict the best possible algorithm.

Generally, data-driven modeling methods have gained popularity in large-scale heterogeneous environments, owing to their broader applicability and higher accuracy. However, although existing data-driven models have achieved satisfactory accuracy, many approaches lack a systematic feature analysis when constructing performance datasets. By primarily focusing on algorithm parameters while neglecting the crucial impact of dynamic runtime system features, these models can be systematically biased. This bias means that even high fitting accuracy on an existing dataset is misleading and does not guarantee the model's generalizability. Therefore, in addition to constructing a more efficient data-driven model, establishing a high-quality dataset that accurately captures the relationship between influencing factors and performance is equally crucial.

2.2. Sunway Accelerate Computing Architecture

The Sunway OceanLight supercomputer introduces a unified programming architecture known as the Sunway Accelerate Computing Architecture (SACA), which serves as the platform for building the performance dataset in the paper.

2.2.1. Sunway OceanLight system

Sunway OceanLight supercomputer adopts an improved fat-tree topology, which forms a multi-level network including leaf switch network and central switch network. Sunway OceanLight includes many racks and each rack is composed of four supernodes. Each supernode contains 256 nodes, each equipped with a SW26010 Pro processor, all of which are connected through the leaf switch [35]. The leaf switch features 304 ports in total, with 256 ports dedicated to the processors and the remaining 48 ports connected to the central switch. If the bandwidth is uniform across all ports, each supernode has a theoretical uplink bandwidth of 2.7 TB/s with each leaf switch port offering a bandwidth of 56.25 GB/s. Actually, bandwidth limitations require different nodes to share the same resources, inevitably leading to competition.

Sunway series supercomputers adopt the LSF (Load Sharing Facility) strategy for job scheduling [36], resulting in non-contiguous physical node distribution for a large-scale computational task. For the reason that the communication bandwidth of nodes within a supernode is different from the nodes within different supernodes or racks, the communication patterns may be different depending on the allocation of physical nodes [17]. These different patterns result in significantly different communication latency and bandwidth, severely impacting the performance of communication-intensive applications like DDP training. Therefore, to accurately predict DDP performance, it is crucial to distinguish these communication patterns when constructing the dataset.

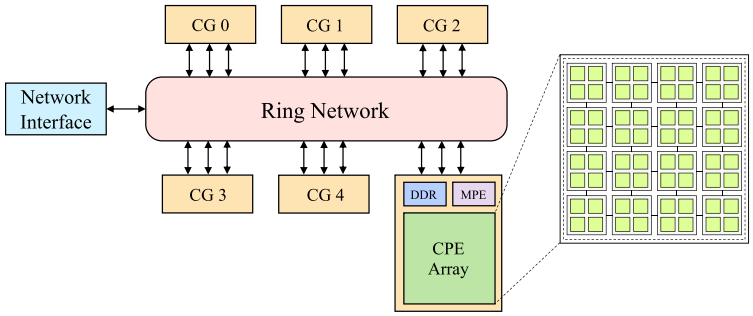


Fig. 1. The architecture of SW26010 Pro, including six CGs connected by ring network.

2.2.2. New generation processor: SW26010 Pro

Instead of using ×86 or ARM architecture processors, Sunway OceanLight employs the SW26010 Pro processor, which features a master–slave heterogeneous architecture as illustrated in Fig. 1. The processor comprises six core groups (CGs) and each core group contains an MPE (Management Processing Element), a CPE (Computing Processing Element) array with 64 cores, and 16 GB of DDR memory. The MPE operates at around 2.1 GHz, while the CPE operates at approximately 2.25 GHz and supports 512-bit vector operations. Each core in the CPE has its own LDM (Local Data Memory), which offers high memory access speed.

With the development of SACA, programs can utilize two computing patterns: single-CG mode and all-shared mode. In the single-CG mode, six core groups run six separate processes with inter-group data exchange facilitated by MPI interfaces. Each core group can only access its own local 16 GB of memory. Computational tasks are typically assigned to the core group on each processor through `MPI_rank`. By contrast, in the all-shared mode, a single process runs on one processor with all six core groups sharing 96 GB of memory. Data exchange between core groups relies on shared memory, which provides the expansive memory capacity required for training larger models. Computational tasks are assigned to each processor via `MPI_rank`, after which the MPE partitions the tasks and distributes them to each core group. In the paper, all-shared mode is used for all analyses and experiments.

2.2.3. swPyTorch: Sunway deep learning library

swPyTorch is an efficient deep learning library on Sunway OceanLight supercomputer, aiming at sufficiently utilizing the CPE array on SW26010 Pro to accelerate the operator execution [37]. Although many deep learning operators have been optimized for the SACA, most operators still only use MPE for computation. To enhance the existing operator library and facilitate performance modeling, the previously unoptimized ReLU6 operator has been optimized to leverage CPE for computation. By utilizing multiple CPE arrays instead of relying solely on the MPE, it achieves a 1370× speed-up in forward propagation and a 15× speed-up in backward propagation with a tensor shape of $1 \times 3 \times 224 \times 224$. Additionally, all operators in swOpset have been verified to use the CPE array for acceleration and their computational patterns can be broadly categorized into three categories: matrix operation, single-CG computation, and multi-CG computation.

The following provides a detailed introduction to three computation patterns:

1. Matrix operation: In the swPyTorch library, the convolution operation and FC (Fully Connected) layer are implemented by calling the matrix operation function in the swBLAS [38] library. Specifically, the matrix function uses only the MPE for small matrix dimensions. For large dimensions, it leverages the CPE array for acceleration.
   Included Operator: Conv, DepthwiseConv, FC Layer
2. Single-CG operation: Operators that use single core group for acceleration are part of the swDNN [23] library. These operators assign corresponding computational tasks to different CPEs based on their $local\_id$. In the all-shared mode, only CG_0 performs the computation, while the other five CGs wait until CG_0 completes its computation before executing the subsequent code.
   Included Operator: MaxPool, AvgPool, BatchNorm
3. Multi-CG operation: Operators that utilize multiple core groups for acceleration are supported by the swTensor [39] library. Each CPE has its local id and also obtains a global id by combining the $group\_id$. The calculation formula is $Global\_id = group\_id \times 64 + local\_id$. Computational tasks are assigned based on the $Global\_id$ and each CPE performs computations in parallel.
   Included Operator: ReLU, ReLU6, Mean, Add, Concat, Mul, Transpose, Split

Each operator exhibits distinct computational characteristics, and constructing tailored analytical performance models for each of them is both time-consuming and labor-intensive. Data-driven models could automatically fit the complex non-linear relationship without the analysis of the underlying algorithm workflow. Therefore, we utilize the data-driven models instead of the analytical models to predict the performance of deep learning operators. Meanwhile, the parameter optimizer, namely AdamW has been enhanced to firstly support the process of parameter update using MPE and CPE arrays for acceleration. The paper models the performance of the default parameter update method in swPyTorch.

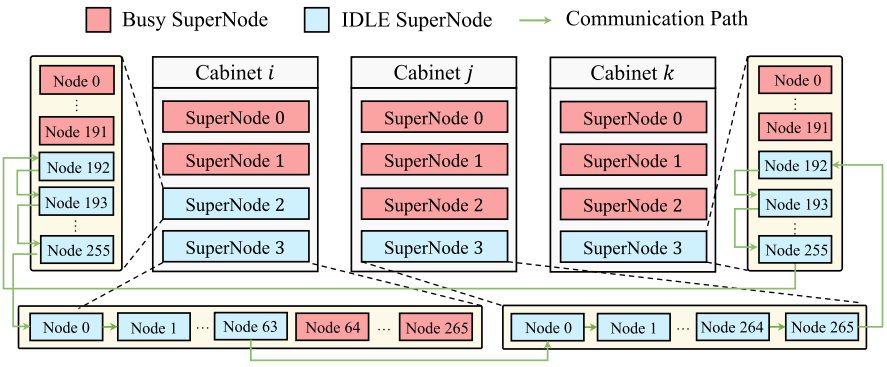


Fig. 2. The communication path of Ring AllReduce algorithm on the Sunway OceanLight supercomputer.

2.2.4. *Distributed data parallelism training algorithm*

Sunway OceanLight supercomputer leverages large-scale node collaboration to improve DDP training efficiency and the communication algorithm is based on the Ring AllReduce algorithm, which is widely used in large-scale distributed training [40,41]. The details are shown in Algorithm 1. First, different computation tasks are distributed to different nodes, and then each node performs forward and backward propagation independently to compute the local gradient. After local computation, for the reduction of global gradient, the Ring AllReduce algorithm is employed as the underlying communication algorithm. Since the small messages have low bandwidth utilization efficiency [42], the tensor fusion method is used to improve the communication performance. To utilize the CPE arrays to accelerate computation, the Ring AllReduce algorithm is implemented by MPI communication primitives `MPI_Sendrecv`, and operators in the swTensor are also utilized for acceleration. The algorithm consists of two phases: reduce-scatter phase and allgather phase. During the reduce-scatter phase, each node communicates with its adjacent nodes to obtain a part of global gradient. And during the allgather phase, the global gradient segments are sent to all nodes, allowing each node to obtain the complete global gradient. Finally, each node can use the global gradient to update the model parameters locally and start a new iteration.

Algorithm 1 DDP Training by Ring AllReduce Algorithm

Require: Training data $\mathcal{D}$, model $\mathcal{M}$, number of nodes N
1: Allocate tasks to $\{node_1, node_2, \ldots, node_N\}$ and split $\mathcal{D}$ into $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$
2: Initialize the parameters of model $\mathcal{M}$ and send parameters to each node, node rank is i
3: **for** each item j in $\mathcal{D}_i$ **do**
4: **Perform forward propagation on $\mathcal{M}$ with item j**
5: **Perform backward propagation and compute local gradient ∇L_i**
6: // **Use Ring AllReduce algorithm to communicate and compute the global gradient**
7: // Stage 1: Reduce-Scatter phase
8: **for** $k = 0$ to $N - 1$ **do**
9: Use `MPI_Sendrecv` to send segment gradient to $Node_{(rank+1)\%N}$ and receive segment gradient from $Node_{(rank+N-1)\%N}$
10: `torch.add`($gradient_{received}, gradient_{local}$), utilizing operators in the swTensor library for accelerating reduction
11: **end for**
12: // Stage 2: AllGather phase
13: **for** $k = 0$ to $N - 1$ **do**
14: Use `MPI_Sendrecv` to send segment gradient to $Node_{(rank+1)\%N}$ and receive segment gradient from $Node_{(rank+N-1)\%N}$
15: **end for**
16: **Update the parameters of model $\mathcal{M}$ using the global gradient ∇L**
17: **end for**

By analyzing the Ring AllReduce algorithm and architecture of Sunway OceanLight, Fig. 2 illustrates a potential communication path where nodes are distributed across different supernodes and racks. Although MPI ranks are contiguous, the physical nodes may be located in different locations. Therefore, the theoretical communication performance bottleneck is determined by the slowest communication bandwidth between two nodes [43]. As a result, the latency and bandwidth limitations of the supernodes and racks can constrain overall performance. In practice, it is challenging to apply for all nodes within a single supernode or rack, so the job scheduler has to allocate distributed nodes. In the paper, swPredictor must account for the impact of different communication patterns, such as intra-supernode, inter-supernode, and inter-rack communication, on the performance to avoid systematic errors.

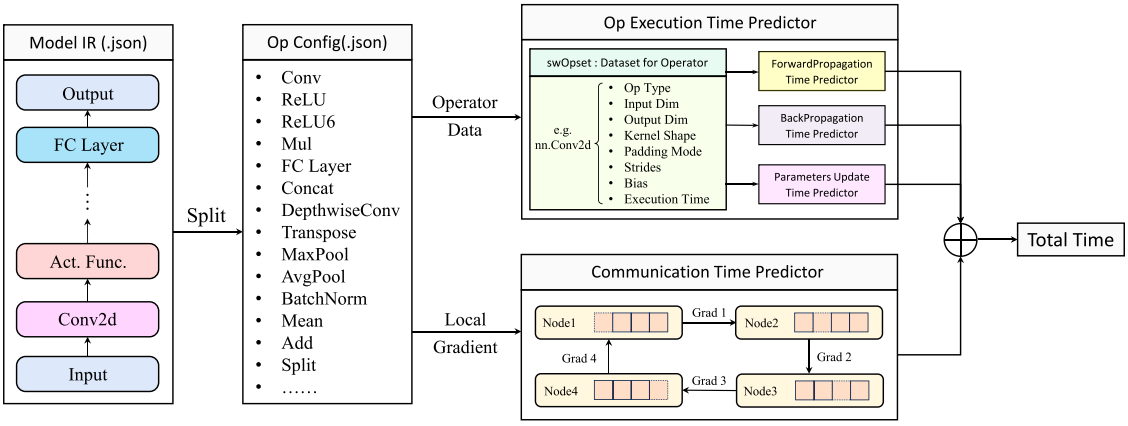


Fig. 3. The overall framework of swPredictor for predicting the DDP training time, which includes operators execution time and communication time.

3. swPredictor design

This section presents a detailed introduction to the performance model swPredictor, showcasing the overall framework of swPredictor, performance dataset construction, and a novel prediction model FI-Net.

3.1. Overall framework

The overall framework of swPredictor is illustrated in Fig. 3. First, a deep learning model is transformed into an Intermediate Representation (IR) and model IR can be decomposed into operator IR. By analyzing the DDP training algorithm, the total time is calculated by summing the operator execution time and communication time. Specifically, the communication volume, namely gradient size, can be directly derived from the model IR. Therefore, only the model IR is required to realize the end-to-end prediction of DDP training time.

The operator execution time includes forward propagation time, backward propagation time and parameter update time. Every operator performs forward propagation and backward propagation, but only a few operators such as Conv, FC layer, BatchNorm and DepthConv, have trainable parameters, which will update their parameters at each step. Therefore, the data-driven model predicts the time of both forward and backward propagation simultaneously, as well as the parameter update time separately. In the PyTorch library, Eager mode is the default execution mode, making model definition and execution more straightforward and concise. Each operation dynamically constructs the computation graph, making it easier to debug and iterate the model. Meanwhile, the deep learning ecosystem on the Sunway system is still under development, and operator fusion is not yet supported. Hence, based on the Sunway OceanLight supercomputer, the total operator execution time is the sum of the execution times of individual operators under Eager mode.

Communication time refers to the time spent on exchanging and reducing gradient during DDP training. The process of communication is implemented by the Ring AllReduce algorithm as discussed in Section 2.2.4. Therefore, the communication time predictor is employed to estimate the total communication time at each step.

3.2. Performance dataset construction

In order to include most of the commonly used operators, the performance dataset uses operator configurations from the open-source dataset provided by nn-Meter [16]. Meanwhile, to evaluate the ability of data-driven performance models to fit the complex nonlinear relationships, the dataset is built based on the Sunway OceanLight supercomputer and includes two parts: operator performance dataset and communication performance dataset.

In many existing studies, the dataset is partitioned according to the model type, which is not applicable to this work. For example, in the nn-Meter's end-to-end experiment, five types of models AlexNet, VGG, MobileNetv1, MobileNetv2 and NASBench are selected, with AlexNet used as the test set and the remaining four types of models serving as the training set. However, the approach presents several significant issues: (1) Different models contain varying numbers of operators, leading to dataset imbalances that severely compromise the prediction accuracy. (2) In the study, all thirteen models are used for experimental analysis, which complicates achieving comprehensive coverage of operator types in both training and test datasets. To address these issues, we initially frame the problem as a minimum set covering problem, defining the operators set as U and the models set as $\{S_1, S_2, \ldots, S_n\}$. And the goal is to determine the smallest number of subsets that collectively include all elements of set U. But certain operators present challenges, such as Split and Transpose, which only appear during the channel shuffle in ShuffleNet.

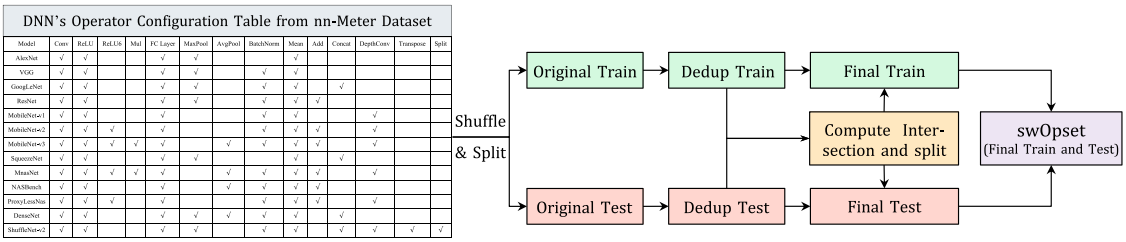


Fig. 4. Data processing workflow for the construction of the operator performance dataset swOpset.

Table 1
The details of operators for $batch = n$ in swOpset, including fourteen types of operators.

Set	Conv2D	ReLU	ReLU6	Mul	FC	MaxPool	AvgPool	BN[a]	Mean	Add	Concat	DWConv[b]	Transpose	Split
OriginTrain	862 791	699 260	179 135	112 708	27 242	43 336	45 960	913 781	20 798	193 416	144 546	155 237	25 008	20 319
OriginTest	219 237	178 574	43 865	29 292	6752	10 658	11 850	232 257	5200	49 867	37 454	38 763	6992	5681
DedupTrain	83 333	2508	1042	620	1087	2301	680	2513	449	573	30 719	6584	24	24
DedupTest	46 786	2405	1031	620	895	1624	587	2415	346	571	15 630	5567	24	24
Intersection	40 586	2389	1031	620	883	1538	572	2401	334	567	11 731	5528	24	24
FinalTrain	75 215	2030	835	496	910	1993	565	2032	384	459	28 372	5478	19	19
FinalTest	14 318	494	207	124	189	394	130	495	82	118	6246	1145	5	5

[a] BN: Batch normalization layer.
[b] DWConv: DepthwiseConv2d, including depthwise convolution and pointwise convolution.

In light of the aforementioned issues, this study adopts a convenient dataset partitioning method to help mitigate data imbalance problems. The specific partitioning process is illustrated in Fig. 4. Initially, the original dataset is randomly divided into $Original\_Train$ and $Original\_Test$ in a $4:1$ ratio. Subsequently, both $Original\_Train$ and $Original\_Test$ are deduplicated to form $Dedup\_Train$ and $Dedup\_Test$. This deduplication relies on an operator's key parameters, such as input/output size, kernel size, stride, and bias. Since operator configurations may be identical across different types of networks, the intersection of the training and test sets, denoted as I, is calculated to ensure that the training set and test set do not overlap. Subsequently, the intersection set I is randomly split in a ratio of $4:1$ and assigned to the training and test sets. Thus the $Final\_Train$ is computed as $Final\_Train = (Dedup\_Train \backslash I) \cup I_{0.8}$. In the same way, the $Final\_Test$ is calculated as $Final\_Test = (Dedup\_Test \backslash I) \cup I_{0.2}$. Finally, the swOpset is composed of $Final\_Train$ and $Final\_Test$.

The details of operators in swOpset for $batch = n$ are presented in Table 1. It is evident that the ratio of the training set to test set is approximately $4:1$, which helps in more effective model training and evaluation of generalization ability. Additionally, since the swPyTorch library is transplanted and optimized from the general-purpose computing platform to the Sunway system, it is supported by underlying libraries such as swBLAS, swDNN and swTensor. Specifically, when the Split and Transpose operators are invoked, no actual data movement occurs. Instead, the tensor format changes are recorded and the actual redistribution of data takes place only during the computation. Therefore, after calling these operators, it is necessary to call the function `tensor.contiguous` to ensure that the data movement occurs. Besides, to reduce the impact of random errors and enhance the quality of the dataset, the time values in the dataset are obtained by running sixteen times and then averaging the results. Specifically, the conditions with batch sizes of 1, 2 and 4 are all tested and utilized to train and test data-driven models in the experiment.

In addition, a communication performance dataset has been constructed based on the Sunway OceanLight supercomputer, which includes an improved fat-tree network. As the detailed analysis of the communication algorithm and system topology in Section 2.2.4, the distributed communication patterns include three types: single-supernode communication, cross-supernode communication and cross-rack communication. Therefore, the communication dataset features different gradient communication performance under multiple communication patterns to adapt to different network topologies and avoid systematic errors. Specifically, an overlap in parameters data is identified between the training and test sets. Given that the ratio of models in training set to test set is $4:1$, the overlapping part is removed from the training set to form a new training set. Similarly, the communication time is averaged over sixteen runs to enhance the quality of dataset.

3.3. A novel regression model with FNO-Inception

In contrast to CV or NLP tasks that focus on extracting semantic representations from images or text, performance modeling is concerned with accurately capturing and predicting operators' execution and communication time. To address this challenge, we propose a novel regression model, FI-Net, whose overall architecture and parameter configuration are illustrated in Fig. 5. The fully connected layer is mainly employed to adjust feature dimensionality. In particular, when the input dimension n is small, an additional FC layer is inserted after the input layer to expand the feature dimension. The core component of the network is the FNO-Inception module, which integrates the strengths of the Fourier Neural Operator [44] and the Inception module [45].

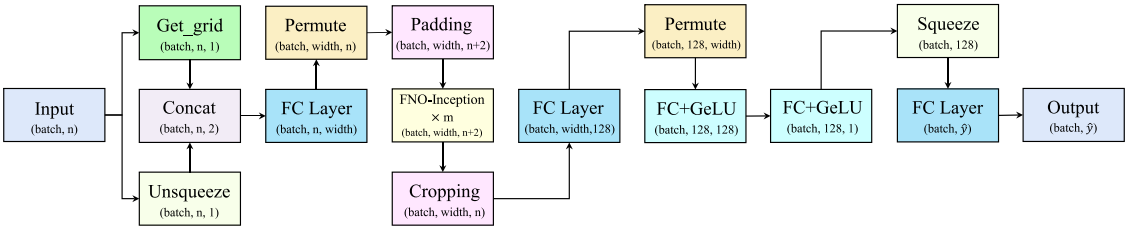


Fig. 5. The architecture of FI-Net, introducing the FNO-Inception module to fit the complex nonlinear relationships. In the paper, the hyperparameter m, denoting the number of FNO-Inception modules, is set to 6.

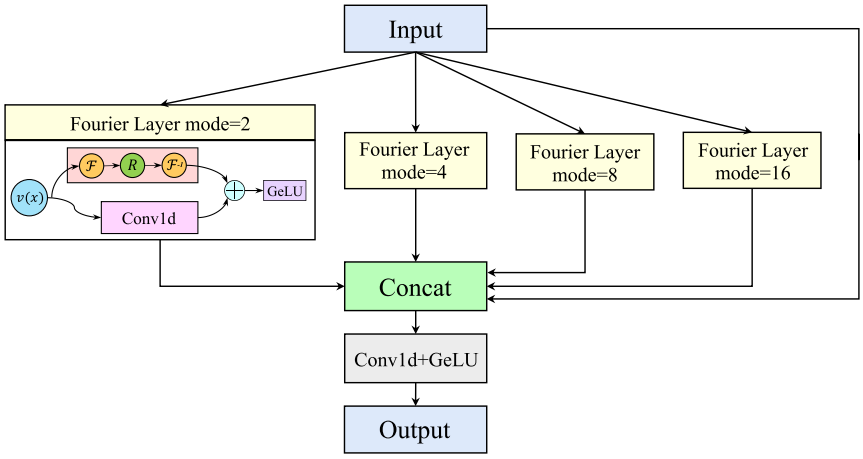


Fig. 6. The structure of FNO-Inception combines the Fourier neural operator and Inception module, which can simultaneously utilize various frequency features.

FNO layer is a neural network module designed to solve partial differential equations (PDEs). Unlike traditional neural networks, the FNO layer utilizes the Fourier transform to capture complex patterns and long-range dependencies, effectively solving the infinite-dimensional mappings and improving the model's efficiency and accuracy. According to the structure of the FNO layer, the number of modes is a critical hyperparameter that determines the quantity of Fourier coefficients used in the frequency domain, directly affecting both accuracy and efficiency. A higher mode typically retains more frequency information, potentially enhancing the model's performance, but it may lead to overfitting. To determine the appropriate mode for specific applications, existing work often relies on hyperparameter tuning methods. However, this approach is time-consuming and labor-intensive for selecting the optimal mode.

To avoid numerous experiments to select an appropriate mode and capture the multiple frequency features, the FNO-Inception is developed by introducing the Inception module. It contains multiple FNO layers and a residual connection [46] to utilize different modes and assist in model training, respectively, as presented in Fig. 6. According to the Nyquist–Shannon theorem [47], the number of modes should be less than or equal to the half frequency components of the original data. Because excessive mode will not provide additional useful information. Since the length of operator configurations is 32 in the performance dataset, the maximum number of mode is 16. In the paper, the FNO-Inception uses different Fourier modes ($mode = 2, 4, 8, 16$) to effectively capture both low-frequency and high-frequency features.

4. The experiment

This section describes the experiments conducted to evaluate the performance of FI-Net. First, it introduces the experiment platform and evaluation metrics. Following that, detailed experiments are presented to verify the accuracy of data-driven models predicting the DDP training time on the Sunway OceanLight supercomputer.

4.1. Experimental setting

The experiments for training and testing data-driven models is completed on the HPC server, whose specific configurations are shown in Table 2. It should be specifically noted that the classical regression models supporting multi-output from the auto-sklearn [48] library, such as AutoRegressor (AR), Decision Tree (DT), Extra Tree (ET), Gaussian Process (GP) and Random Forest

Table 2
The configuration of HPC server.

Name		Version
Software	OS	Ubuntu 22.04
	Auto-sklearn	0.15.0
	PyTorch	2.2.2
	Python	3.10.14
Hardware	CPU	AMD EPYC 7543
	Memory	2003 G
	GPU	NVIDIA A800
	CUDA	12.4

Table 3
Settings for training data-driven models.

Parameter		Value
Training setting	Epoch	500
	Width	256
	Optimizer	Adam
	Loss function	LpLoss (L = 2)
	Learning rate	1.0×10^{-4}
Auto-sklearn	Memory limit	16 GB
	Time limit	8 h
	Ensemble	True

(RF), are chosen as baseline models, which can automatically choose the best model to fit the training set. Apart from classical machine learning models, neural networks such as FNO-1d [32] and ResPerfNet [30], are also used as comparative models, which have demonstrated strong predictive performance on HPC applications or DNNs.

Table 3 summarizes the settings and hyperparameters used for training the data-driven models. For neural networks, the number of training epochs is set to 500. In the FI-Net architecture, a width of 256 was chosen to strike a balance between expressive capacity and computational cost. For classical machine learning models in the auto-sklearn library, the training time is limited to eight hours, and an ensemble strategy is employed to automatically combine candidate models, thereby improving generalization.

The evaluation metrics employed are MSE (Mean Squared Error) and MAPE (Mean Absolute Percentage Error), defined in Eqs. (1) and (2). MSE quantifies the discrepancy between the predicted values and actual values. However, MAPE is generally considered a more critical metric for assessing the accuracy of predictions, as it provides a normalized measure of deviation.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{1}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100\% \tag{2}$$

4.2. DDP training performance prediction on Sunway OceanLight

To validate the models' ability to fit the complex relationships within a large-scale heterogeneous environment, the subsection presents the experiment of data-driven models predicting the performance of DDP training on the Sunway OceanLight supercomputer.

4.2.1. Prediction of operator execution time

For testing the performance of models predicting operator execution time, data-driven models are trained and tested on the swOpset. And the results are shown in Table 4. Most of the machine learning models achieve relatively commendable performance with an MAPE of less than 10%, except for the GP model. It is attributed to their ability to search for the optimal combination of models and hyperparameters within the search space. In addition, the neural network model ResPerfNet demonstrates poor performance, while the FNO-1d achieves good performance, indicating that the FNO layer is well-suited to fit complex nonlinear systems. Furthermore, the proposed model FI-Net surpasses all other baseline models, achieving a more accurate prediction.

For analyzing the FI-Net in more detail, Fig. 7 shows the 90% confidence interval of MAPE for forward and backward propagation prediction of each operator. Specifically, the dot represents the average MAPE. It is found that the FC layer has large prediction errors compared to the convolution operator and depthwise convolution operator, probably due to the small sample size in swOpset. Similarly, the errors of Split and Transpose operators are greater than Concat, though they all rely on the Copy operator in the swTensor. In general, the errors of operators are within acceptable limits for accurate prediction.

Similarly, Fig. 8 shows the 90% confidence interval of MAPE for parameter update based on the FI-Net. Evidently, only the error of the convolution operator is less than the average error, while the errors of other operators are greater than the average error. Overall, the average error in predicting parameter update is only 0.77%.

Table 4
The experimental results of data-driven models predicting operator execution time.

Model	Forward propagation		Backward propagation		Parameter update	
	↓MSE (10^{-5})	↓MAPE (%)	↓MSE (10^{-5})	↓MAPE (%)	↓MSE (10^{-10})	↓MAPE (%)
AutoRegressor[a]	1.28	5.94	4.21	7.65	4.63	0.93
Decision tree	0.63	2.82	1.61	2.51	4.84	0.97
Extra tree	0.74	2.41	2.36	2.64	4.87	0.95
Gaussian process	170.97	965.78	687.00	964.94	827.43	11.04
Random forest	0.68	2.86	1.81	3.74	4.75	0.968
ResPerfNet	175.90	24.72	710.38	29.41	404.22	2.32
FNO-1d	0.78	2.00	1.69	1.79	3.90	0.78
FI-Net	0.17	**1.21**	0.24	**0.95**	3.79	**0.77**

[a] AutoRegressor: default regression model in the auto-sklearn library.

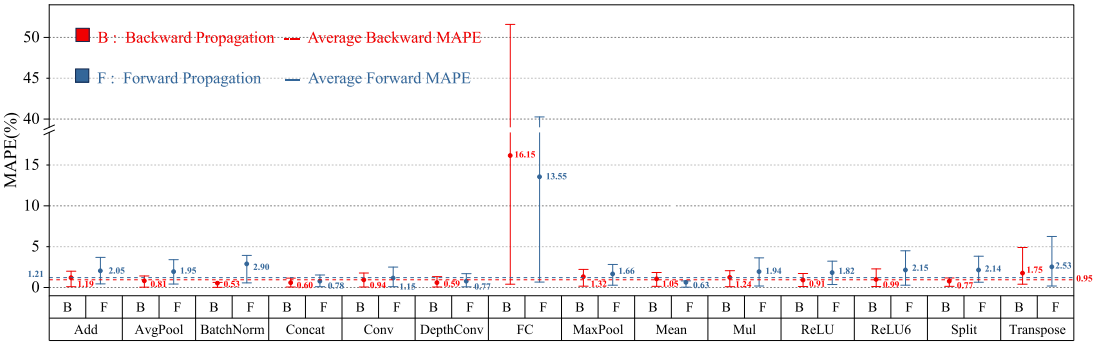


Fig. 7. The performance of FI-Net predicting the time of forward and backward propagation for each operator.

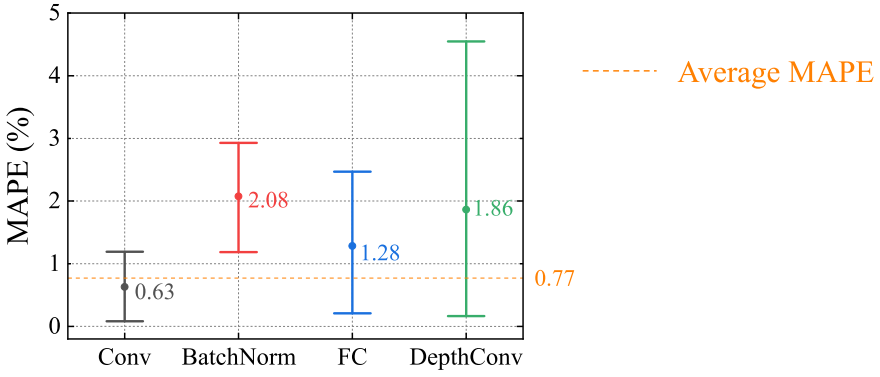


Fig. 8. The performance of FI-Net predicting the time of parameter update.

4.2.2. Prediction of communication time

For testing the data-driven models' ability to predict the communication time with different nodes under multiple communication patterns, the experiments are based on the communication performance dataset constructed on the Sunway OceanLight supercomputer.

In order to verify whether different communication patterns will affect the performance, the experiment presents the communication time of different gradient and communication patterns under 256, 512 and 1024 nodes. Fig. 9 presents the experimental data, which is obtained by sampling 32 data points from the communication dataset. It shows that even with a consistent number of nodes, different physical nodes will lead to variations in communication time. If different communication patterns are not distinguished, it will cause systematic errors within the dataset.

The models' prediction results across different communication patterns are shown in Fig. 10. In most cases, both machine learning models and neural networks obtain an accurate prediction. Specifically, similar to the prediction of operators execution, the GP model also performs poorly in predicting communication time. Besides, by analyzing the results of models predicting the communication time of 256 nodes, the data-driven models perform better when predicting communication times within a rack.

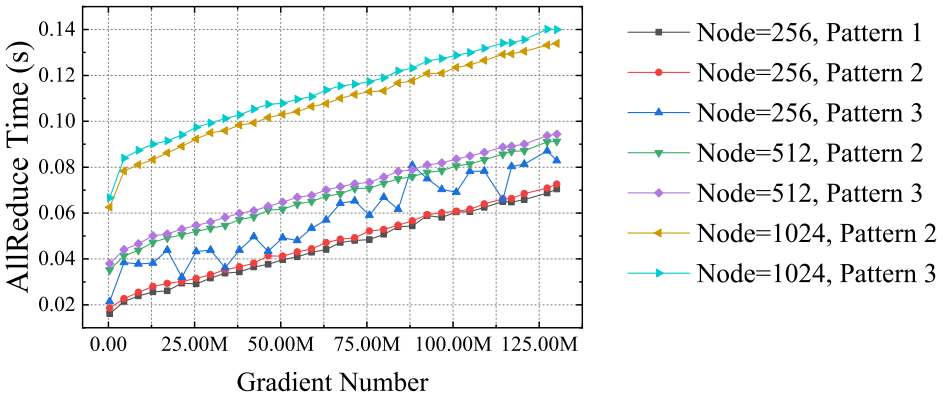


Fig. 9. Gradient reduction time under different communication patterns. *Pattern* = 1 means all nodes are within a supernode; *Pattern* = 2 means it exists cross-supernode communication; *Pattern* = 3 means it exists cross-rack communication.

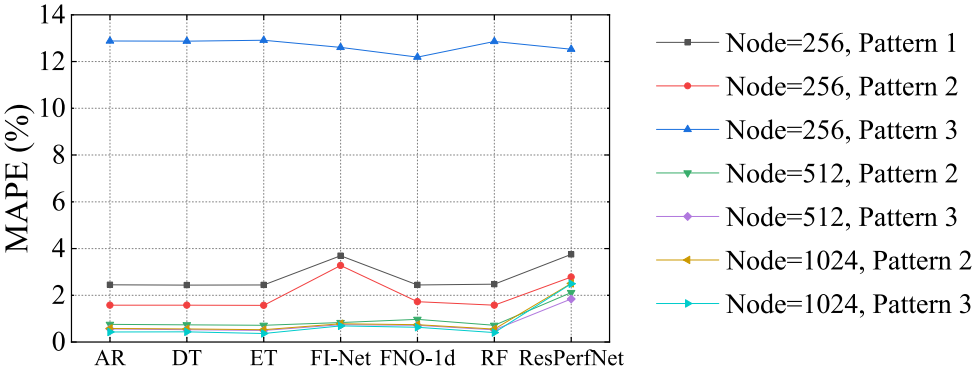


Fig. 10. The performance of models predicting communication time across different patterns. *Pattern* = 1 means all nodes are within a supernode; *Pattern* = 2 means it exists cross-supernode communication; *Pattern* = 3 means it exists cross-rack communication.

This difference is likely due to the increased data jitter in the communication performance dataset. In the process of DDP training on large-scale HPC clusters, the most commonly used approach is to distribute nodes across different racks, as it is challenging to allocate all nodes within a single rack. Therefore, the following experiment shows the results of 1024 nodes DDP training under the Pattern of 3.

4.2.3. Results of DDP training time prediction

The section presents the final end-to-end performance prediction, integrating the component-level results from Sections 4.2.1 and 4.2.2. Our evaluation focuses on a representative 1024-node, cross-rack communication scenario with a per-node batch size of 1. Such a batch size is commonly employed in contemporary large-scale model training due to memory constraints.

The results of models are shown in Fig. 11. Among the classical machine learning models, the prediction results are acceptable. However, the MAPE of GP model exceeds 300% and ResPerfNet exhibits an MAPE of 56.86%. With the FNO layer to fit the complex relationships, the prediction error of FNO-1d is lower than 2%. Moreover, FI-Net demonstrates the best performance among the baseline models, with an overall prediction error of only 0.93%. It indicates that FI-Net can accurately predict the performance of DDP training on the Sunway OceanLight supercomputer, effectively supporting resource management and model design.

4.2.4. Ablation experiment

To validate the effectiveness of the FNO-Inception structure, an ablation experiment is conducted. By varying the number of modes in the FNO layer, the experiment assesses the impact on the prediction accuracy of FI-Net. Table 5 presents the experimental results of MSE and MAPE for the prediction of forward propagation, backward propagation and parameter update with different modes.

By analyzing the experimental results, it is observed that more modes do not necessarily lead to higher prediction accuracy. For example, the model with $mode = 8$ outperforms the model with $mode = 16$. Besides, FI-Net with the FNO-Inception structure exhibits the best performance, meaning the multi-branch structure effectively combines the advantages of different modes,

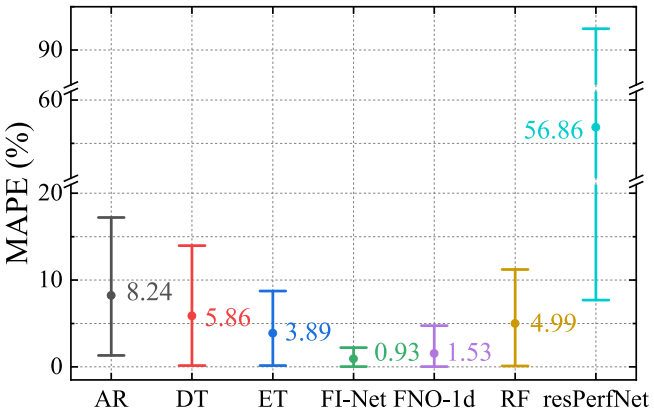


Fig. 11. The results of data-driven models predicting 1024 nodes DDP training time under cross-rack communication pattern.

Table 5
The results of ablation experiment to prove the effectiveness of FNO-Inception.

FNO mode	Forward propagation		Backward propagation		Parameter update	
	↓MSE (10^{-5})	↓MAPE (%)	↓MSE (10^{-5})	↓MAPE (%)	↓MSE (10^{-10})	↓MAPE (%)
mode = 2	0.58	1.76	1.03	1.55	4.41	0.79
mode = 4	0.44	1.58	0.73	1.35	3.83	0.78
mode = 8	0.25	1.39	0.35	1.16	4.30	0.78
mode = 16	0.39	1.49	0.48	1.30	3.81	0.79
mode = [2, 4, 8, 16][a]	0.17	**1.21**	0.24	**0.95**	3.79	**0.77**

[a] mode = [2, 4, 8, 16]: use multi-branch structure, FNO-Inception.

significantly improves the prediction accuracy and avoids multiple experiments to select an appropriate mode. Therefore, it proves the effectiveness of the FNO-Inception structure in performance modeling.

4.2.5. Summary

In this section, it has been verified that FI-Net achieves good performance in predicting operator execution time and communication time. Combining these two components, the prediction results show that FI-Net outperforms all models with an average MAPE of 0.93% in predicting the DDP training time on the Sunway OceanLight supercomputer. In addition, it has also been demonstrated that the FNO-Inception module surpasses the FNO layer with a single mode.

5. Conclusion

This paper presents a novel data-driven performance model swPredictor designed for predicting the performance of DDP training on large-scale HPC clusters. To fit the complex nonlinear relationships between algorithm parameters and performance within the heterogeneous environment, we construct a new regression prediction model namely FI-Net, which includes a novel multi-branch structure FNO-Inception. By introducing the Inception module to the FNO layer, the FNO-Inception can utilize various frequency features and improve the model's accuracy. Meanwhile, it avoids additional experiments to select the optimal FNO mode. Furthermore, in order to make the model adapt to different network topologies and eliminate systematic errors, we develop an original performance dataset by considering the impact of runtime system topology on algorithm performance. The experimental results demonstrate that the proposed model FI-Net can accurately predict the time of DDP training on Sunway OceanLight supercomputer with an overall MAPE of 0.93%.

Building on swPredictor's solid foundation, future work will extend our methodology. We aim to generalize the model to encompass other distributed parallelism strategies and communication algorithms, and to validate its predictive accuracy across diverse large-scale system architectures.

CRediT authorship contribution statement

Xianyu Zhu: Writing – review & editing, Writing – original draft, Methodology, Data curation. **Ruohan Wu:** Writing – review & editing. **Junshi Chen:** Writing – review & editing. **Hong An:** Writing – review & editing.

Declaration of competing interest

No potential conflict of interest was reported by the authors.

Acknowledgments

The paper is supported by the Chinese Academy of Sciences (Strategic Priority Research Program, China), Grant (XDB0500102). And this work is financially supported by Laoshan Laboratory (China) (LSKJ202300305). We would like to thank Ziyue You from the University of Edinburgh, Shengtao Xue from the Northwest A&F University and Jiahao Huang from the National University of Singapore, for their valuable feedback and assistance in revising the manuscript.

Data availability

The authors are unable or have chosen not to specify which data has been used.

References

[1] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, et al., Learning skillful medium-range global weather forecasting, Science 382 (6677) (2023) 1416–1421.
[2] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A.J. Ballard, J. Bambrick, et al., Accurate structure prediction of biomolecular interactions with AlphaFold 3, Nature (2024) 1–3.
[3] S. Das, B. Kanungo, V. Subramanian, G. Panigrahi, P. Motamarri, D. Rogers, P. Zimmerman, V. Gavini, Large-scale materials modeling at quantum accuracy: Ab initio simulations of quasicrystals and interacting extended defects in metallic alloys, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2023, pp. 1–12.
[4] X. Liang, M. Li, Q. Xiao, J. Chen, C. Yang, H. An, L. He, Deep learning representations for quantum many-body systems on heterogeneous hardware, Mach. Learn. Sci. Technol. 4 (1) (2023) 015035–015046.
[5] T. Patki, D. Ahn, D. Milroy, J.-S. Yeom, J. Garlick, M. Grondona, S. Herbein, T. Scogland, Fluxion: A scalable graph-based resource model for HPC scheduling challenges, in: Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, 2023, pp. 2077–2088.
[6] M. Li, H. Lin, J. Chen, J.M. Diaz, Q. Xiao, R. Lin, F. Wang, G.R. Gao, H. An, swFLOW: A large-scale distributed framework for deep learning on sunway TaihuLight supercomputer, Inform. Sci. 570 (2021) 831–847.
[7] H. Cai, L. Zhu, S. Han, ProxylessNAS: Direct neural architecture search on target task and hardware, in: International Conference on Learning Representations, 2019, pp. 1–13.
[8] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10734–10742.
[9] J. Mellor, J. Turner, A. Storkey, E.J. Crowley, Neural architecture search without training, in: International Conference on Machine Learning, PMLR, 2021, pp. 7588–7598.
[10] H. Qi, E.R. Sparks, A. Talwalkar, Paleo: A performance model for deep neural networks, in: International Conference on Learning Representations, 2017, pp. 1–10.
[11] R. Wu, M. Li, H. Li, T. Chen, X. Tian, X. Xu, B. Zhou, J. Chen, H. An, Machine learning-enabled performance model for DNN applications and ai accelerator, in: 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application, HPCC/DSS/SmartCity/DependSys, 2022, pp. 25–34.
[12] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, 2013.
[13] D. Bruneo, A stochastic model to investigate data center performance and QoS in iaas cloud computing systems, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2013) 560–569.
[14] J.J. Thiagarajan, R. Anirudh, B. Kailkhura, N. Jain, T. Islam, A. Bhatele, J.-S. Yeom, T. Gamblin, Paddle: Performance analysis using a data-driven learning environment, in: 2018 IEEE International Parallel and Distributed Processing Symposium, IEEE, 2018, pp. 784–793.
[15] J. Tan, J. Pang, C. Liu, What factors affect the performance of software after migration: A case study on sunway TaihuLight supercomputer, IEICE Trans. Inf. Syst. 105 (1) (2022) 26–30.
[16] L.L. Zhang, S. Han, J. Wei, N. Zheng, T. Cao, Y. Yang, Y. Liu, Nn-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices, in: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, 2021, pp. 81–93.
[17] Z. Ma, J. He, J. Qiu, H. Cao, Y. Wang, Z. Sun, L. Zheng, H. Wang, S. Tang, T. Zheng, et al., BaGuaLu: Targeting brain scale pretrained models with over 37 million cores, in: Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2022, pp. 192–204.
[18] S. Nair, S. Abbasi, A. Wong, M.J. Shafiee, Maple-edge: A runtime latency predictor for edge devices, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3660–3668.
[19] Z. Li, M. Paolieri, L. Golubchik, Predicting inference latency of neural architectures on mobile devices, in: Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering, 2023, pp. 99–112.
[20] Y. Zhang, J.D. Owens, A quantitative performance analysis model for GPU architectures, in: 2011 IEEE 17th International Symposium on High Performance Computer Architecture, 2011, pp. 382–393.
[21] S. Ramos, T. Hoefler, Capability models for manycore memory systems: A case-study with Xeon Phi KNL, in: 2017 IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2017, pp. 297–306.
[22] W. Chen, X. Dong, H. Chen, Q. Wang, X. Yu, X. Zhang, Performance evaluation of convolutional neural network on Tianhe-3 prototype, J. Supercomput. (2021) 1–19.
[23] J. Fang, H. Fu, W. Zhao, B. Chen, W. Zheng, G. Yang, swDNN: A library for accelerating deep learning applications on sunway TaihuLight, in: 2017 IEEE International Parallel and Distributed Processing Symposium, 2017, pp. 615–624.
[24] R.W. Hockney, C.R. Jesshope, Parallel Computers 2: Architecture, Programming and Algorithms, CRC Press, 2019.
[25] D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, T. Von Eicken, LogP: Towards a realistic model of parallel computation, in: Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 1993, pp. 1–12.
[26] E. Nuriyev, J.-A. Rico-Gallego, A. Lastovetsky, Model-based selection of optimal MPI broadcast algorithms for multi-core clusters, J. Parallel Distrib. Comput. 165 (2022) 1–16.

[27] M. Wilkins, H. Wang, P. Liu, B. Pham, Y. Guo, R. Thakur, P. Dinda, N. Hardavellas, Generalized collective algorithms for the exascale era, in: 2023 IEEE International Conference on Cluster Computing, IEEE, 2023, pp. 60–71.

[28] S. Chen, W. He, F. Qi, Y. Zheng, K. Yu, Hybrid approach to optimize MPI collectives by in-network-computation and point-to-point messages, in: 2022 7th International Conference on Computer and Communication Systems, IEEE, 2022, pp. 773–783.

[29] S. Kaufman, P. Phothilimthana, Y. Zhou, C. Mendis, S. Roy, A. Sabne, M. Burrows, A learned performance model for tensor processing units, Proc. Mach. Learn. Syst. 3 (2021) 387–400.

[30] C.-C. Wang, Y.-C. Liao, C.-H. Tu, M.-C. Kao, W.-Y. Liang, S.-H. Hung, ResPerfNet: Deep residual learning for regressional performance modeling of deep neural networks, 2020, pp. 1–14, arXiv preprint arXiv:2012.01671.

[31] L. Bai, W. Ji, Q. Li, X. Yao, W. Xin, W. Zhu, Dnnabacus: Toward accurate computational cost prediction for deep neural networks, in: In Proceedings of ACM Conference, Conference'17, 2017, pp. 1–9.

[32] Y. Zhang, Y. Liu, P. Jiao, Y. Zhou, T. Wei, Automatic multi-parameter performance modeling of HPC applications on a new sunway supercomputer, IEEE Trans. Parallel Distrib. Syst. 34 (11) (2023) 2965–2977.

[33] S. Hunold, A. Carpen-Amarie, Algorithm selection of MPI collectives using machine learning techniques, in: 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, IEEE, 2018, pp. 45–50.

[34] S. Hunold, A. Bhatele, G. Bosilca, P. Knees, Predicting MPI collective communication performance using machine learning, in: 2020 IEEE International Conference on Cluster Computing, IEEE, 2020, pp. 259–269.

[35] R. Lin, X. Yuan, W. Xue, W. Yin, J. Yao, J. Shi, Q. Sun, C. Song, F. Wang, 5 ExaFlop/s HPL-mxp benchmark with linear scalability on the 40-million-core sunway supercomputer, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2023, pp. 1–13.

[36] Y. Dai, Y. Dong, K. Lu, R. Wang, W. Zhang, J. Chen, M. Shao, Z. Wang, Towards scalable resource management for supercomputers, in: SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2022, pp. 1–15.

[37] T. Zheng, Research on Pytorch's Optimization Method Based on Sunway Supercomputing System (Master's thesis), Shandong University, 2022.

[38] W. Zhao, H. Fu, J. Fang, W. Zheng, L. Gan, G. Yang, Optimizing convolutional neural networks on the sunway taihulight supercomputer, ACM Trans. Archit. Code Optim. 15 (1) (2018) 1–26.

[39] X. Zhong, H. Yang, Z. Luan, L. Gan, G. Yang, D. Qian, swTensor: Accelerating tensor decomposition on Sunway architecture, CCF Trans. High Perform. Comput. 1 (3) (2019) 161–176.

[40] S. Liu, J. Gao, X. Liu, Z. Huang, T. Zheng, Establishing high performance AI ecosystem on Sunway platform, CCF Trans. High Perform. Comput. 3 (3) (2021) 224–241.

[41] Z. Hu, S. Shen, T. Bonato, S. Jeaugey, C. Alexander, E. Spada, J. Dinan, J. Hammond, T. Hoefler, Demystifying NCCL: An in-depth analysis of GPU communication protocols and algorithms, 2025, pp. 1–13, arXiv preprint arXiv:2507.04786.

[42] H. Lin, X. Tang, B. Yu, Y. Zhuo, W. Chen, J. Zhai, W. Yin, W. Zheng, Scalable graph traversal on sunway taihulight with ten million cores, in: 2017 IEEE International Parallel and Distributed Processing Symposium, IEEE, 2017, pp. 635–645.

[43] Baidu, Baidu ring allreduce, 2017, September 11th, 2024, URL https://github.com/baidu-research/baidu-allreduce.

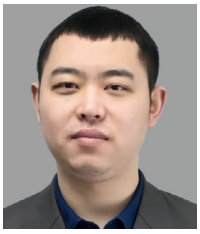
[44] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2020, pp. 1–16.

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[46] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

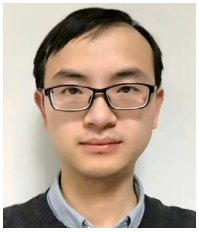
[47] C.L. Farrow, M. Shaw, H. Kim, P. Juhás, S.J. Billinge, Nyquist-Shannon sampling theorem applied to refinements of the atomic pair distribution function, Phys. Rev. B—Condens. Matter Mater. Phys. 84 (13) (2011) 134105–134114.

[48] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: Advances in Neural Information Processing Systems 28 (2015), 2015, pp. 2962–2970.



Xianyu Zhu currently majors in computer science and technology at University of Science and Technology of China. His research interests include high-performance computing and computer architecture.

Ruohan Wu currently majors in computer science and technology at University of Science and Technology of China. His research interests include high-performance computing, computer architecture and AI for science.



Junshi Chen received his Ph.D. degree in computer science from the University of Science and Technology of China (USTC), Hefei, in 2020. He is assistant professor of USTC. His research interests include high-performance computing and computer architecture.

Hong An is currently a Professor at School of Computer Science and Technology, University of Science and Technology of China (USTC). She received her Ph.D. degree in Computer Science from the USTC in 2000. She is the director of the Advanced Computer System Architecture (ACSA) Lab in the USTC. Her main research focuses on parallel computer architecture, parallel programming, operating system design, and high performance computing.