PHM-Vibench: A Unified and Factory-Style Vibration Benchmarking Framework for the Foundation Model Era

Qi Li[1,2], Bojian Chen[3], Xuan Li[4], Qitong Chen[5], Liang Chen[5], Changqing Shen[5], Lu Lu[2*], Zhaoye Qin[1*], and Fulei Chu[1]

[1] *Department of Mechanical Engineering, Tsinghua University, Beijing, 100084, PR China*
liq22@tsinghua.org.cn, qinzy@mail.tsinghua.edu.cn, chufl@mail.tsinghua.edu.cn

[2] *Department of Statistics and Data Science, Yale University, New Haven, CT, USA*
lu.lu@yale.edu

[3] *State Key Laboratory of Industrial Control Technology,*
College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, PR China
bjchen@zju.edu.cn

[4] *Belt and Road Joint Laboratory on Measurement and Control Technology,*
School of Artificial Intelligence and Automation,
Huazhong University of Science and Technology, Wuhan 430074, PR China
xuanli423@hust.edu.cn

[5] *Soochow University, Suzhou, PR China*
20234029004@stu.suda.edu.cn, ChenL@suda.edu.cn, cqshen@suda.edu.cn

ABSTRACT

The Prognostics and Health Management (PHM) field faces significant challenges due to fragmented benchmarks, inconsistent evaluation protocols, and limited accessibility to comprehensive frameworks, particularly in the era of large-scale data and foundation models. To address these critical limitations, we introduce PHM-Vibench, a unified, extensible, and modular benchmarking platform for vibration-based PHM research. PHM-Vibench features a novel architecture that decouples the PHM pipeline into distinct data, model, task, and trainer factories, enabling flexible instantiation and customization of specific PHM workflows. The platform integrates comprehensive 20+ datasets with standardized protocols. It supports diverse PHM tasks including fault diagnosis, remaining useful life prediction, and anomaly detection. The framework addresses complex scenarios such as domain generalization, cross-system transfer, few-shot learning. Grounded in the Unified PHM Problem (UPHMP) framework with seven fundamental spaces: domain knowledge space ($\mathbb{P}$), data space ($\mathbb{D}$), task space ($\mathbb{T}$), model space ($\mathbb{M}$), loss function space ($\mathbb{L}$), protocol space (Π), and evaluation metric space ($\mathbb{E}$), PHM-Vibench enables systematic problem formalization and reproducible experimentation. The platform accommodates both traditional machine learning models and foundation models, with extensive experimental validation demonstrating superior cross-domain performance. PHM-Vibench addresses the standardization challenges in PHM research and provides a comprehensive solution for benchmarking and advancing the field. The platform is openly available at https://github.com/PHMbench/PHM-Vibench.

1. INTRODUCTION

Prognostics and Health Management (PHM) is a critical engineering discipline focused on predicting the failure of systems or components, thereby enhancing the reliability, safety, and efficiency of industrial operations (Zio, 2022). The field has experienced rapid evolution from traditional model-based approaches to sophisticated data-driven methodologies, with machine learning and deep learning models being applied to core tasks such as fault diagnosis (FD), remaining useful life (RUL) prediction, and anomaly detection (AD) (Lei et al., 2018; Y. Wu, Sicard, & Gadsden, 2024). Advanced research now explores complex scenarios including domain generalization, meta-learning, and few-shot learning to address the fundamental challenge of model robustness across diverse op-

erational conditions and system configurations (C. Zhao, Zio, & Shen, 2024a; Feng et al., 2022).

However, PHM research currently lacks the systematic theoretical foundations that have enabled rapid progress in computer vision and natural language processing. Unlike these established fields, PHM suffers from inconsistent problem formulations, ad hoc evaluation protocols, and fragmented datasets with incompatible formats. This theoretical gap becomes particularly problematic as the field embraces deep learning models and foundation model paradigms, which require rigorous frameworks for systematic evaluation and comparison.

Existing benchmarks, while valuable for specific applications such as deep learning-based fault diagnosis (Z. Zhao et al., 2020, 2021), represent isolated solutions rather than unified theoretical frameworks. The PHM community faces practical challenges: bespoke data processing pipelines that prevent cross-study validation, inconsistent evaluation metrics that preclude meaningful performance comparisons, and domain-specific solutions that cannot be systematically generalized. The emergence of foundation models in PHM amplifies these challenges, demanding theoretically grounded frameworks for systematic research advancement.

To address these gaps, we introduce the Unified PHM Problem (UPHMP) framework along with its practical implementation through PHM-Vibench. The UPHMP framework establishes theoretical foundations through seven fundamental spaces that enable systematic problem representation and formal optimization protocols. PHM-Vibench provides a unified benchmarking platform that bridges theoretical rigor with reproducible experimental validation. Our key contributions include:

- **Unified Theoretical Framework:** We develop the UPHMP framework with seven fundamental spaces $(\mathbb{P}, \mathbb{D}, \mathbb{T}, \mathbb{M}, \mathbb{L}, \Pi, \mathbb{E})$ that provides the theoretical foundation for PHM research standardization and cross-study comparisons.
- **Practical Benchmarking Platform:** PHM-Vibench implements this framework through a modular architecture that decouples PHM pipelines into data, model, task, and trainer components, enabling reproducible evaluation across diverse scenarios.
- **Comprehensive Validation:** We integrate public PHM datasets with standardized preprocessing and evaluation protocols, providing examples with domain generalization, cross-system transfer, few-shot learning and foundation model integration.

This paper is structured as follows: Section 2 reviews existing PHM benchmarks and related work. Section 3 details the design and implementation of the PHM-Vibench platform. Section 4 presents preliminary experimental results and discussions, including demonstrations of domain generalization and cross-dataset evaluation. Finally, Section 5 concludes the paper and outlines future research directions, including the development of foundation models for PHM.

2. RELATED WORK

Standardized benchmarks are foundational to progress in data-driven research, and PHM is no exception. Several open initiatives already support PHM experimentation. pyPHM provides Python utilities for accessing and preparing a small set of canonical datasets (e.g., IMS bearings, U.C. Berkeley milling, Airbus helicopter accelerometer), which facilitates reproducible workflows but remains limited in scope (von Hahn & Mechefske, 2023). ProgPy (NASA) offers a mature prognostics library with model- and algorithm-level abstractions and includes dataset loaders for C-MAPSS turbofan and NASA battery data, providing a practical foundation for RUL-centric studies (Teubert, Jarvis, Corbetta, Kulkarni, & Daigle, 2023; Saxena & Goebel, 2008). More recently, PHMD curates a substantially larger catalog ($\approx$ 59 datasets at publication) with unified search/load interfaces (Solís-Martín, Galán-Páez, & Borrego-Díaz, 2025).

More specialized benchmarks also exist. For example, Zhao et al. have developed influential benchmarks for bearing fault diagnosis, initially focusing on deep learning and transfer learning applications (Z. Zhao et al., 2020, 2021), and more recently extending their work to address the specific challenges of domain generalization (C. Zhao, Zio, & Shen, 2024b).

The functionality of existing tools and platforms is summarized in Table 1. In summary, the features are as follows:

- **pyPHM** focuses on accessing and preprocessing common PHM datasets, with a class hierarchy for datasets and simple preprocessing (e.g., windowing) that supports standardized workflows.
- **ProgPy** provides physics-based and data-driven prognostic models, state estimation, prognostics, evaluation, visualization, and selected dataset downloaders.
- **PHMD** supports 59 datasets with unified search/load interfaces, standardized formats, metadata, and task-specific experimental settings; includes cross-validation and seeds for reproducibility.
- **Zhao'20** releases an open-source benchmark comparing deep learning models across seven datasets with unified code and evaluation settings.
- **Zhao'21 (UDTL)** provides a taxonomy, comparative study, and a released test framework for unsupervised deep transfer learning in intelligent fault diagnosis.
- **Zhao'24 (DGFD)** conducts a domain generalization benchmark on eight open-source and two self-collected datasets and releases code; the paper frames domain generalization for fault diagnosis applications and protocols.

PHM-Vibench extends existing approaches by providing com-

prehensive support for advanced scenarios including domain generalization, cross-system transfer, few-shot learning, and foundation model integration within a unified theoretical framework.

Notwithstanding these contributions, current tools typically face critical constraints:

- Limited scope and narrow dataset coverage or specialization by equipment type;
- Weak or ad hoc support for users to provide **private and consortial datasets**, which are common in industrial collaborations;
- Inadequate extensibility for new models, tasks, or datasets.

PHM-Vibench addresses these gaps through three key innovations:

- **Comprehensive Dataset Integration**: Broadens access by integrating extensive public PHM datasets (building on catalogs such as PHMD and the PHM Society) while providing metadata standards in HuggingFace or ModelScope that are automatically downloaded during code execution.
- **Modular Architecture Design**: Features a modular architecture that enables seamless integration of new algorithms (including state-of-the-art time-series models), preprocessing pipelines, and tasks.
- **Standardized Evaluation Framework**: Emphasizes comprehensive, standardized evaluation, combining general machine learning metrics with PHM-specific measures to ensure rigorous assessment across diverse scenarios.

3. PHM-VIBENCH DESIGN AND IMPLEMENTATION

3.1. Overview

PHM-Vibench targets three primary goals: standardization, extensibility, and reproducibility for vibration-based PHM research. The platform provides a seamless workflow from data to evaluation through five core components. These include dataset management for both public and private data with standardized formats, data preprocessing with common techniques and custom extensions, and baseline model implementations with integration frameworks. Additionally, the platform supports task definitions for fault diagnosis, anomaly detection, and RUL prediction, along with evaluation engines featuring standardized metrics and protocols.

The platform follows a three-step workflow. First, researchers specify experimental configurations by selecting elements from the seven UPHMP spaces, creating a complete problem formalization. Second, PHM-Vibench instantiates appropriate factory components based on this specification, handling data preprocessing, model initialization, and evaluation protocols automatically. Finally, results are generated following standardized evaluation protocols with performance metrics, confidence intervals, and cross-domain analysis.

This approach transforms the UPHMP theoretical framework into practical research capabilities while maintaining experimental reproducibility.

3.2. Unified PHM Problem (UPHMP) Framework

We introduce the UPHMP framework based on seven fundamental spaces that represent any PHM problem. Each space contains all possible choices for that component. For instance, the data space $\mathbb{D}$ contains every possible PHM dataset, while the model space $\mathbb{M}$ contains every possible algorithm. A specific PHM experiment selects one element from each space, creating a complete experimental specification.

The UPHMP framework is theoretically represented as:

$$\Omega = (\mathbb{P}, \mathbb{D}, \mathbb{T}, \mathbb{M}, \mathbb{L}, \Pi, \mathbb{E}) \tag{1}$$

where each space contains all possible instances of its respective component:

- $\mathbb{P}$ denotes the domain knowledge space, which represents physical constraints, engineering expertise, and regularization terms
- $\mathbb{D}$ denotes the data space, which represents PHM datasets with sensor measurements, labels, and metadata
- $\mathbb{T}$ denotes the task space, which represents fault diagnosis, RUL prediction, anomaly detection, and forecasting tasks
- $\mathbb{M}$ denotes the model space, which represents traditional ML models and foundation models with embedding-backbone-head architecture
- $\mathbb{L}$ denotes the loss function space, which represents optimization objectives from standard ML to PHM-specific losses
- Π denotes the protocol space, which represents evaluation protocols ensuring reproducibility and preventing data leakage
- $\mathbb{E}$ denotes the evaluation metric space, which represents performance measures from general ML metrics to PHM-specific assessments

Any specific PHM problem is formalized as a **meta-setting** $\omega = (P, D, T, M, L, \pi, E)$, defined as a complete specification that selects exactly one element from each of the seven UPHMP spaces. A meta-setting provides a unique experimental configuration that fully determines the problem domain knowledge, dataset, task formulation, model architecture, loss function, evaluation protocol, and performance metrics. For example, a bearing fault diagnosis experiment might be repre-

Table 1. Related Work comparisons

Dimension	pyPHM	ProgPy	PHMD	DLFD	UDTL	DGFD	PHM-Vibench
Dataset scale	Few	Few	59 datasets	Medium	Medium	$\approx$ 10 datasets	Target $>$ 20
Input/Output abstraction & task readiness	✓	∼	✓✓	✓	✓	✓	✓✓
Model/algorithm layer	×	✓✓	×	✓	✓	✓	✓✓
Task coverage (AD, FD, and RUL)	∼, ✓, ∼	✓, ×, ✓	∼, ✓, ✓	✓, ✓, ×	✓, ✓, ×	✓, ✓, ×	✓, ✓✓, ✓✓
Evaluation (general metrics)	✓	✓	✓	✓	✓	✓	✓✓
Advanced scenarios	×	∼	×	×	✓	✓✓	✓✓
Reproducibility (splits, seeds, scripts)	∼	∼	✓✓	✓	✓	✓	✓✓

Legend: ✓✓ = Strong and Native support, ✓ = Supported, ∼ = Partial and Limited support, × = Not provided

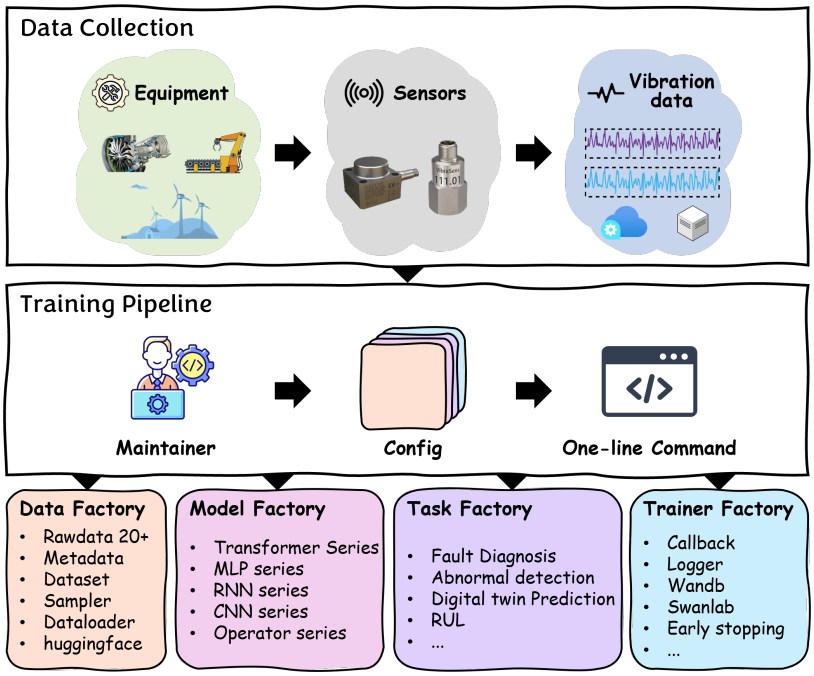

Figure 1. PHM-Vibench Architecture Overview: The unified factory-style benchmarking framework demonstrating the systematic integration of data, model, task, and trainer factories with comprehensive dataset support and standardized evaluation protocols.

sented as:

$$\omega = (P_{phys}, D_{CWRU}, T_{cls}, M_{CNN}, L_{CE}, \pi_{split}, E_{F1}) \quad (2)$$

using physics-based constraints, the CWRU bearing dataset, classification task, CNN model, cross-entropy loss, 80-20 train-test split, and F1-score evaluation. This formalization enables comparison across different datasets, models, and evaluation protocols while ensuring reproducible experiments through protocol isolation guarantees.

The framework supports both traditional machine learning and foundation models. The model space $\mathbb{M}$ encompasses transformer architectures and pre-trained models, while the protocol space Π accommodates specialized evaluation scenarios. The task space $\mathbb{T}$ enables assessment across zero-shot, few-shot, and full fine-tuning configurations.

The framework addresses complex PHM scenarios through specific meta-setting configurations. For example, domain Generalization uses leave-one-domain-out protocols:

$$\omega_{DG} = (P_{domain}, D_{multi}, T_{cls}, M, L, \pi_{LODO}, E) \quad (3)$$

where ω_{DG} represents the domain generalization meta-setting, P_{domain} denotes domain-specific prior knowledge, D_{multi} represents multi-domain datasets, T_{cls} indicates classification task specification, M denotes the model space, L represents the loss space, π_{LODO} specifies the leave-one-domain-out protocol, and E represents the evaluation space, with domain isolation constraint $d_{train} \cap d_{test} = \emptyset$. Cross-System Transfer employs leave-one-system-out evaluation:

$$\omega_{CSG} = (P_{physics}, D_{systems}, T, M, L, \pi_{LOSO}, E) \quad (4)$$

where ω_{CSG} represents the cross-system generalization meta-setting, $P_{physics}$ denotes physics-based prior knowledge, $D_{systems}$ represents multi-system datasets, T indicates task specification, M denotes the model space, L represents the loss space, π_{LOSO} specifies the leave-one-system-out protocol,

and E represents the evaluation space, with system isolation constraint $\sigma_{train} \cap \sigma_{test} = \emptyset$. Few-Shot Learning uses episodic training with class isolation:

$$\omega_{FSL} = (D_{episodes}, T, M_{meta}, L, P, \pi_{\text{N-way-K-shot}}, E) \quad (5)$$

where ω_{FSL} represents the few-shot learning meta-setting, $D_{episodes}$ denotes episodic datasets for meta-learning, T indicates task specification, M_{meta} represents meta-learning model architectures, L denotes the loss space, P represents prior knowledge, $\pi_{\text{N-way-K-shot}}$ specifies the N-way K-shot learning protocol, and E represents the evaluation space.

Foundation Model Integration leverages pre-trained models with adaptation strategies. This formalization enables PHM-Vibench to provide unified evaluation while maintaining experimental rigor. And all of the meta-settings ω can be instantiated through the factory components.

3.3. PHM-Vibench Implementation: Factory-Style Framework Instantiation

PHM-Vibench is a theoretically grounded PHM benchmark platform that demonstrates the practical instantiation of the UPHMP framework. The platform uses a single configuration file specifying elements from all seven UPHMP spaces. The platform instantiates core factory components: Data Factory (F_D), Task Factory (F_T), Model Factory (F_M), and Trainer Factory (F_π). This architecture enables complete meta-setting instantiation $F(\omega)$ for end-to-end PHM experiments.

3.3.1. Data Factory

The data space encompasses datasets containing input features, labels, and associated metadata. Formally, this is represented as:

$$\mathbb{D} = \{D | D = (X, Y, \Xi), X \in \mathscr{X}, Y \in \mathscr{Y}, \Xi \in \mathscr{M}\} \quad (6)$$

where X represents input features, Y denotes labels, and Ξ contains metadata. This structure preserves domain, system, and temporal organization essential for PHM applications.

The Data Factory centralizes dataset registration, reading, preprocessing, and splitting to produce reproducible data pipelines with UPHMP compliance. Splits satisfy rigorous *protocol isolation* (no leakage) as required by $\pi \in \Pi$:

$$I_{\text{train}} \cap I_{\text{val}} = I_{\text{val}} \cap I_{\text{test}} = I_{\text{train}} \cap I_{\text{test}} = \varnothing \quad (7)$$

where I_{train}, I_{val}, and I_{test} represent the index sets for training, validation, and testing data respectively, and $\varnothing$ denotes the empty set, ensuring complete data isolation across all splits.

Additional optional constraints include $d(I_{\text{train}} \cup I_{\text{val}}) \cap d(I_{\text{test}}) = \varnothing$ (cross-domain), $\sigma(I_{\text{train}} \cup I_{\text{val}}) \cap \sigma(I_{\text{test}}) = \varnothing$ (cross-system), and temporal hold-out $I_{\text{train}} \subset \{\tau < \tau_0\}$, $I_{\text{test}} \subset \{\tau \geq \tau_0\}$, where sliding windows *must not* cross τ_0.

The DataFactory provides the following key capabilities:

- Register and read public and private datasets; standardize storage (`.npy` and HDF5; shape (B, L, C)).
- Preprocess (normalization, resampling, windowing) and inject metadata (domain and site, system and source, timestamp).
- Splitting and sampling: class, domain, and source stratification; construct few-shot episodes (N-way, K-shot, Q-query).
- Generate reproducible data streams (fixed seeds, batch sizes, distributed sampling methods, consistent batch organization).

3.3.2. Model Factory

The model space encompasses all possible architectures and their parametric representations. This space is defined as:

$$\mathbb{M} = \{M_\theta | \theta \in \Theta, M_\theta : \mathscr{X} \to \mathscr{Y}\} \quad (8)$$

where M_θ represents a model with parameters θ from parameter space Θ, mapping input space $\mathscr{X}$ to output space $\mathscr{Y}$. This formulation maintains unified architecture abstractions while preserving parameter space topology.

The Model Factory instantiates models under the *Embedding–Backbone–Head* decomposition

$$f_\theta = f_h \circ f_b \circ f_e \quad (9)$$

where f_θ represents the complete model function, f_h denotes the head component for task-specific output processing, f_b represents the backbone component for feature extraction, f_e denotes the embedding component for input processing, and $\circ$ indicates function composition. The factory declares trainable parameter subsets $\Delta\Theta \subseteq \Theta$.

Advantages of Embedding-Backbone-Head Architecture: The Embedding-Backbone-Head decomposition offers several advantages for vibration signal processing in PHM applications:

- The embedding component f_e enables hierarchical feature learning by processing raw vibration signals into meaningful representations, capturing frequency-domain characteristics and temporal patterns specific to rotating machinery.
- The backbone f_b supports multi-scale pattern recognition by providing flexible feature extraction capabilities, enabling the learning of multi-scale representations essential for diagnosing faults at different severity levels.
- The head component f_h facilitates task-specific adaptation, allowing seamless integration with diverse PHM tasks (classification, regression, anomaly detection) without modifying the core feature extraction layers.

- The decomposition allows for parameter-efficient fine-tuning by enabling selective parameter updates through $\Delta\Theta \subseteq \Theta$, which facilitates transfer learning across different equipment types and operating conditions while preserving learned knowledge.

This architecture aligns with the nature of vibration signals, which exhibit multi-frequency characteristics and require both local feature extraction and global pattern recognition for effective fault diagnosis and prognosis.
The ModelFactory provides the following essential capabilities:

- Modular implementations (CNN, Transformer, and advanced models), with unified forward shapes and numerical stability.
- Pretraining and model initialization; support parameter adaptation strategies (e.g., head fine-tuning, adapters, LoRA, full fine-tuning).
- Device and precision policies (supporting GPU acceleration and mixed precision), and reporting of parameter counts and complexity.

3.3.3. Task Factory

The task space defines all possible PHM problem formulations through input-output specifications and constraints. This is formalized as:

$$\mathbb{T} = \{T | T = (\mathscr{X}_T, \mathscr{Y}_T, \text{constraints}_T)\} \tag{10}$$

where $\mathscr{X}_T$ and $\mathscr{Y}_T$ represent task-specific input and output spaces, while constraints_T define task-specific requirements. This ensures standardized input-output contracts and evaluation protocol consistency.

The Task Factory binds task contracts, losses from $\mathbb{L}$, and metrics from $\mathbb{E}$ to ensure theoretically consistent evaluation.

The factory design provides unified access to Loss Space $\mathbb{L}$ and Evaluation Space $\mathbb{E}$ elements through task specification, ensuring consistency across task instantiation $\phi_T(T)$, loss instantiation $\phi_L(L)$, and evaluation instantiation $\phi_E(E)$. The TaskFactory delivers the following specialized services:

- Task definitions: classification (fault diagnosis), anomaly detection (score-based and binary), RUL and regression, forecasting and reconstruction.
- Losses and metrics: Cross Entropy (CE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Pinball Loss, etc.; F1 Score (macro/micro), Area Under Receiver Operating Characteristic Curve (AUROC), Precision-Recall Area Under Curve (PR-AUC), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Prognostics and Health Management (PHM) scores.
- Evaluation protocol: calibrate thresholds and temperatures on `validation` data; aggregate results by overall, per-domain and per-system, and confidence intervals (CI).
- Few-shot learning: episodic evaluation with adaptation methods (prototypical networks, linear probing, lightweight fine-tuning).

3.3.4. Trainer Factory

The Trainer Factory implements the rigorous two-phase UPHMP optimization protocol with reproducible execution.
Phase 1 Implementation (Training): Model optimization with Domain Knowledge $P \in \mathbb{P}$ regularization from protocol $\pi \in \Pi$:

$$\theta^* = \arg\min_{\Theta} \underbrace{\frac{1}{|I_{\text{train}}|} \sum_{i \in I_{\text{train}}} \ell_T(f_\Theta(x_i), y_i)}_{L \in \mathbb{L}} \tag{11}$$

$$+ \underbrace{\lambda \mathscr{R}(\Theta, P)}_{P \in \mathbb{P}} \tag{12}$$

through standard optimization procedures.

$$\theta_{\text{best}} = \arg\max_{\theta^*} \underbrace{e_{\text{val}}(\theta^*, D_{\text{val}})}_{E \in \mathbb{E}} \tag{13}$$

where θ_{best} represents the optimal model parameters selected based on validation performance, θ^* denotes the candidate optimal parameters from training, e_{val} is the validation evaluation function, D_{val} represents the validation dataset, and $E \in \mathbb{E}$ indicates the evaluation space element used for model selection.

Phase 2 Implementation (Testing): Evaluation on protocol-isolated test set with zero mutual information guarantee $I(D_{\text{train}}; D_{\text{test}}) = 0$. The TrainerFactory orchestrates the following critical operations:

- Two-phase training orchestration with protocol isolation enforcement.
- Model selection via validation, final testing via protocol-defined metrics $E \in \mathbb{E}$.
- Reproducible execution: fixed seeds, configuration versioning, systematic record-keeping.

3.3.5. Foundation Model Architecture and Integration

PHM-Vibench incorporates foundation models as a critical component of the model space $\mathbb{M}$, representing a paradigm shift towards pre-trained, generalizable architectures for PHM tasks. Foundation models in PHM-Vibench follow the embedding-backbone-head decomposition $f_\theta = f_h \circ f_b \circ f_e$, enabling flexible adaptation across diverse PHM scenarios.

Model Architecture. Our implementation encompasses diverse foundation model architectures, including CNN-based

models like TimesNet (H. Wu et al., 2022), the Neural Operator models like Fourier Neural Operator (FNO) (Li et al., 2021), Transformer-based models like PatchTST (Nie, Nguyen, Sinthong, & Kalagnanam, 2023), and MLP-based models like DLinear (Zeng, Chen, Zhang, & Xu, 2022). Each architecture captures distinct aspects of temporal dependencies in vibration signals, while unified interface integration preserves the theoretical properties of the model space $\mathbb{M}$ across various PHM applications.

Multi-Task Learning Integration. Simultaneous learning across fault diagnosis, anomaly detection, and RUL prediction tasks is facilitated through shared backbone representations paired with task-specific heads. This architecture enables effective knowledge transfer between PHM objectives while preserving task isolation when necessary for experimental rigor.

Domain Adaptation Capabilities. Flexible adaptation strategies encompass full fine-tuning, parameter-efficient methods such as LoRA and adapters, and few-shot learning protocols. Such versatility allows models to accommodate new systems, operating conditions, and fault types by building upon pretrained representations from related PHM domains.

3.4. Pipeline Implementation: UPHMP Meta-Setting Execution

The core pipeline of PHM-Vibench provides systematic meta-setting execution $\phi(\omega)$ for any valid UPHMP meta-setting $\omega = (P, D, T, M, L, \pi, E) \in \Omega_{valid}$, where Ω_{valid} represents the set of all theoretically consistent meta-settings. The platform implements this execution through a modular architecture that follows the theoretically motivated sequence:

The pipeline begins with configuration parsing, transforming YAML specifications into valid meta-settings $\omega \in \Omega_{valid}$ through theoretical consistency checking across all seven spaces. Theoretical validation then verifies meta-setting consistency, including space-element relationships, protocol isolation requirements, and property preservation.

Following validation, space instantiation proceeds systematically across all UPHMP components. Data space instantiation executes $\phi_D(D)$ to create DataFactory instances with protocol-aware splitting that respects isolation constraints $\pi \in \Pi$. Task space instantiation executes $\phi_T(T)$, integrating loss space $\phi_L(L)$ and evaluation space $\phi_E(E)$ elements with type-safe contracts. Model space instantiation executes $\phi_M(M)$ to create parameter space Θ with embedding-backbone-head decomposition preserving theoretical topology. Protocol instantiation executes $\phi_\pi(\pi)$ with domain knowledge integration $\phi_P(P)$ for two-phase optimization.

Optimization execution implements the training phase through:

$$\theta^* = \arg\min_{\Theta} \underbrace{L(\phi_M(M)_\theta, \phi_D(D)_{\text{train}}, \phi_T(T))}_{\text{Loss Space Implementation}} \tag{14}$$

$$+ \quad \underbrace{\lambda \mathscr{R}(\theta, \phi_P(P))}_{\text{Domain Knowledge Integration}} \tag{15}$$

where θ^* represents the optimal model parameters, Θ denotes the parameter space, $L(\cdot)$ is the task-specific loss function from loss space implementation, λ is the regularization weight, and $\mathscr{R}(\theta, \phi_P(P))$ represents domain knowledge regularization terms, with standard optimization monitoring.

After training completion, protocol-isolated evaluation ensures rigorous testing through verified isolation. Isolation verification confirms zero mutual information $I(\phi_D(D)_{\text{train}}; \phi_D(D)_{\text{test}}) = 0$ between training and testing data before evaluation. Final evaluation executes testing phase computation $P_{final} = \phi_E(E)(\theta_{\text{best}}, \phi_D(D)_{\text{test}})$ with confidence interval computation. Meta-setting results generation produces theoretically grounded outcomes with reproducibility metrics, property validation, and UPHMP compliance verification.

Throughout this process, the pipeline implementation preserves all UPHMP theoretical properties through systematic enforcement. Optimization monitoring provides practical tracking of training progress and validation performance throughout the learning process. Isolation guarantees ensure zero mutual information between training and testing phases, maintaining experimental rigor. Reproducibility assurance through fixed seeds and configuration versioning enables deterministic meta-setting execution. Consistency guarantees ensure all factory interactions preserve space-element relationships from the theoretical framework.

Meta-Setting Catalog and Extensibility: PHM-Vibench implements a comprehensive catalog of validated meta-settings $\{\omega_{DG}, \omega_{CSG}, \omega_{FSL}, \omega_{RUL}, \omega_{AD}, \ldots\} \subset \mathbb{M}_{valid}$, where $\mathscr{M}_{catalog}$ represents the curated collection of validated meta-settings including domain generalization (ω_{DG}), cross-system generalization (ω_{CSG}), few-shot learning (ω_{FSL}), remaining useful life (ω_{RUL}), and anomaly detection (ω_{AD}), each with proven theoretical-practical equivalence. The modular design of the pipeline enables systematic extension to new PHM scenarios while preserving UPHMP theoretical properties and implementation guarantees.

4. UPHMP FRAMEWORK VALIDATION THROUGH EXPERIMENTAL DEMONSTRATIONS

This section presents comprehensive experimental validation demonstrating how the UPHMP theoretical framework translates to practical implementation through PHM-Vibench. Each experiment validates specific theoretical properties: meta-setting consistency, protocol isolation enforcement, and implementation-theory equivalence. The demonstrations focus

on three paradigmatic scenarios that showcase the framework's ability to unify diverse PHM challenges under a systematic theoretical approach while maintaining practical applicability.

The experimental validation is structured into classification-focused demonstrations (Examples 1-3) that showcase domain generalization, cross-system transfer, and few-shot learning capabilities, followed by foundation model integration analysis covering multiple PHM task types including fault diagnosis, anomaly detection, and remaining useful life prediction.

Experimental Design Principles: All experiments follow UPHMP theoretical requirements:

1. **Meta-Setting Instantiation**: Each experiment implements a complete meta-setting $\omega = (P, D, T, M, L, \pi, E) \in \mathbb{M}_{valid}$ through factory pattern mapping $\phi(\omega)$
2. **Protocol Isolation Verification**: Rigorous enforcement of isolation constraints with validation of zero mutual information
3. **Performance Monitoring**: Systematic tracking of training progress and validation metrics during optimization
4. **Reproducibility Assurance**: Fixed seeds and deterministic execution enabling exact experimental replication
5. **Theoretical Property Validation**: Post-hoc verification that implementation preserves all UPHMP theoretical properties

4.1. Example 1: Domain Generalization

We evaluate domain generalization using leave-one-domain-out validation across five vibration datasets (D1, D5, D6, D13, D19 detailed in Table 2). The protocol ensures strict domain isolation: $d(D_{\text{train}} \cup D_{\text{val}}) \cap d(D_{\text{test}}) = \emptyset$. We compare four model architectures (DLinear (Zeng et al., 2022), PatchTST (Nie et al., 2023), FNO (Li et al., 2021), TimesNet (H. Wu et al., 2022)) using cross-entropy loss and F1-score evaluation. The experimental setup employs only basic loss functions without incorporating physical constraints or domain-specific prior knowledge regularization terms.

Results in Table 3 show TimesNet achieves superior cross-domain robustness (0.9073±0.015 on D19), demonstrating the effectiveness of the framework for domain generalization evaluation.

4.2. Example 2: Cross-System Transfer Learning

PHM-Vibench validated cross-system generalization using meta-setting ω_{CSG} with Leave-One-System-Out protocol. Testing across bearing systems from different manufacturers demonstrated the ability of the framework to systematically evaluate model transfer capabilities while maintaining protocol isolation $\sigma(D_{train}) \cap \sigma(D_{test}) = \emptyset$. The evaluation employs standard cross-entropy loss without incorporating physical constraints or system-specific prior knowledge regularization terms. Results in Table 4 demonstrated the superior cross-system performance of TimesNet (0.9635±0.0018), validating the systematic approach of the framework to transfer learning evaluation.

4.3. Example 3: Few-Shot Learning Evaluation

PHM-Vibench validated few-shot learning using episodic meta-setting ω_{FSL} with N-way K-shot protocol ensuring class isolation. The experimental setup relies solely on basic cross-entropy loss functions without incorporating physical constraints or meta-learning-specific prior knowledge regularization terms. Results in Table 5 show FNO demonstrated superior few-shot adaptation (0.6996±0.0722), confirming the systematic approach of the framework to meta-learning evaluation.

4.4. Example 4: Foundation Model Evaluation

This example demonstrates foundation model evaluation across three distinct PHM task types: fault diagnosis (classification), anomaly detection (score-based binary classification), and remaining useful life (RUL) prediction (regression). This multi-task evaluation validates the capability of the UPHMP framework to systematically assess foundation model performance across diverse PHM applications within a unified theoretical structure.

We conducted comprehensive foundation model evaluation using the UPHMP meta-setting ω_{FM} across datasets detailed in Table 2 (D1, D5, D6, D13, D19) plus additional dataset D2 i.e XJTU dataset. The evaluation employs in-distribution testing with partial train-test splits within each system to assess multi-task performance across fault diagnosis, anomaly detection, and RUL prediction tasks. The experimental approach utilizes only fundamental loss functions (cross-entropy for classification, MSE for regression) without incorporating physical constraints or foundation model-specific prior knowledge regularization terms.

Table 6 presents comprehensive performance across classification accuracy, anomaly detection, and RUL prediction tasks, demonstrating the effectiveness of foundation models with different architectures.

FNO demonstrates superior overall performance with the lowest loss (1.2750±0.0148) and highest classification accuracy (0.9627±0.0048), while DLinear achieves the best anomaly detection accuracy (0.8944±0.0467). The results validate the systematic approach of the UPHMP framework to foundation model evaluation across diverse PHM tasks.

5. CONCLUSION AND FUTURE WORK

This paper introduces the Unified PHM Problem (UPHMP) framework and its practical implementation through PHM-Vibench. The UPHMP framework provides a systematic theo-

Table 2. Dataset Specifications for example demonstrations

ID	Dataset Name	Classes	Rate (Hz)	Duration
D1	CWRU	4	12k/48k	40.3s
D5	Ottawa	3	200k	10s
D6	THU	4	20.48k	60s
D13	Ottawa19	5	42k	10s
D19	HUST	6	25.6k	10.2s

Table 3. Domain Generalization Performance Validation

	D1	D5	D6	D13	D19
TimesNet	0.765±0.0086	0.0449±0.0248	0.3887±0.01315	0.9934±0.0025	0.9073±0.015
FNO	0.7338±0.027	0.0664±0.0276	0.5295±0.0127	0.9955±0.0017	0.7802±0.0264
PatchTST	0.7922±0.0202	0.0039±0	0.6732±0.0178	0.9954±0.0021	0.948±0.0096
DLinear	0.6926±0.0247	0.9901±0.0028	0.0723±0.0098	0.9901±0.0028	0.8448±0.0243

Table 4. Cross-System Generalization Performance Validation

CDDG task	D1	D5	D6	D13	D19
TimesNet	0.7924 ± 0.0114	0.4023 ± 0.0884	0.5854 ± 0.0656	0.9980 ± 0.0016	0.9635 ± 0.0018
FNO	0.8509 ± 0.0202	0.0078 ± 0.0000	0.4697 ± 0.0435	0.9718 ± 0.0313	0.9177 ± 0.0310
PatchTST	0.8104 ± 0.0695	0.1112 ± 0.0028	0.2702 ± 0.0252	0.9752 ± 0.0129	0.8502 ± 0.0368
DLinear	0.7176 ± 0.0088	0.0508 ± 0.0663	0.3386 ± 0.0293	0.9735 ± 0.0105	0.8812 ± 0.0660

Table 5. Few-Shot Learning Performance Validation

GFS task	D1	D5	D6	D13	D19
TimesNet	0.4282 ± 0.0769	0.2619 ± 0.1077	0.2636 ± 0.0132	0.3957 ± 0.0828	0.1282 ± 0.0558
FNO	0.4327 ± 0.0836	0.3780 ± 0.0449	0.3364 ± 0.0199	0.6996 ± 0.0722	0.2513 ± 0.0616
PatchTST	0.4260 ± 0.1148	0.3775 ± 0.1125	0.2611 ± 0.0412	0.5423 ± 0.0450	0.1966 ± 0.0076
DLinear	0.4666 ± 0.0000	0.2019 ± 0.0442	0.2427 ± 0.0141	0.3752 ± 0.0193	0.0994 ± 0.0025

Table 6. Foundation Model In-Distribution Test Performance

Backbone	Loss	Classification Acc	Anomaly Test Acc	RUL Mae
TimesNet	1.3060±0.0032	0.7776±0.0021	0.7506±0.0021	0.4830±0.0000
FNO	1.2750±0.0148	0.9627±0.0048	0.8276±0.1104	0.4829±0.0000
PatchTST	2.1664±0.0113	0.7781±0.0010	0.8307±0.0785	0.4834±0.0000
DLinear	1.4699±0.0079	0.9294±0.0188	0.8944±0.0467	0.4834±0.0000

retical approach to PHM research through seven fundamental spaces $(\mathbb{P}, \mathbb{D}, \mathbb{T}, \mathbb{M}, \mathbb{L}, \mathbf{\Pi}, \mathbb{E})$, enabling systematic meta-setting definitions and two-phase optimization protocols that address inconsistent evaluation and poor reproducibility in PHM research.

PHM-Vibench demonstrates practical implementation through factory pattern architecture with comprehensive dataset integration and UPHMP-compliant protocols. Experimental validation across domain generalization, cross-system transfer, and few-shot learning scenarios confirms the effectiveness of the framework for systematic PHM research.

5.1. Future Work

Building upon the UPHMP framework and PHM-Vibench platform demonstrated in this paper, we outline several promising directions for future research:

Equipment Diversity Expansion: While current demonstrations focus on bearing datasets due to their standardized formats and availability, future work will expand the platform to encompass gear systems, turbofan engines, and other industrial rotating machinery. This expansion will validate the versatility across diverse PHM applications and demonstrate its capacity for cross-equipment transfer learning scenarios.

Foundation Model Development: Leveraging the modular architecture of PHM-Vibench, we will develop PHM-specific foundation models pretrained on large-scale vibration datasets, enabling zero-shot and few-shot capabilities for new equipment types and fault conditions.

Physics-informed Learning Integration: Future iterations will incorporate physics-based constraints directly into the UPHMP framework's domain knowledge space $\mathbb{P}$, enabling seamless integration of physical models with data-driven approaches for improved robustness and interpretability.

Community Standardization: We will continue extending

UPHMP to multi-modal PHM domains, develop foundation models through protocol-aware pre-training and universal meta-learning, and promote community standardization efforts. The complete space-to-factory mapping $F : \Omega \to \mathscr{F}$ establishes theoretical grounding for PHM research, enabling reliable, generalizable solutions with formal theoretical backing for industrial applications.

REFERENCES

Feng, Y., Chen, J., Xie, J., Zhang, T., Lv, H., & Pan, T. (2022, January). Meta-learning as a promising approach for few-shot cross-domain fault diagnosis: Algorithms, applications, and prospects. *Knowledge-Based Systems*, *235*, 107646.

Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018, May). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*, 799–834.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier Neural Operator for Parametric Partial Differential Equations.

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers.

Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set (c-MAPSS).* NASA Prognostics Center of Excellence (PCoE) Data Repository.

Solís-Martín, D., Galán-Páez, J., & Borrego-Díaz, J. (2025). PHMD: An easy data access tool for prognosis and health management datasets. *SoftwareX*, *29*, 102039.

Teubert, C., Jarvis, K., Corbetta, M., Kulkarni, C., & Daigle, M. (2023). ProgPy: Python packages for prognostics and health management of engineering systems. *Journal of Open Source Software*, *8*(87), 5099.

von Hahn, T., & Mechefske, C. K. (2023, February). Computational Reproducibility Within Prognostics and Health Management. *Journal of Dynamics, Monitoring and Diagnostics*, 42–50.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2022). TimesNet: Temporal 2D-variation modeling for general time series analysis. , *abs/2210.02186*, null.

Wu, Y., Sicard, B., & Gadsden, S. A. (2024). Physics-informed machine learning: A comprehensive review on applications in anomaly detection and condition monitoring. *Expert Systems with Applications*, *255*, 124678.

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2022). Are transformers effective for time series forecasting?

Zhao, C., Zio, E., & Shen, W. (2024a, May). Domain generalization for cross-domain fault diagnosis: An application-oriented perspective and a benchmark study. *Reliability Engineering & System Safety*, *245*, 109964.

Zhao, C., Zio, E., & Shen, W. (2024b, May). Domain generalization for cross-domain fault diagnosis: An application-oriented perspective and a benchmark study. *Reliability Engineering & System Safety*, *245*, 109964.

Zhao, Z., Li, T., Wu, J., Sun, C., Wang, S., Yan, R., & Chen, X. (2020, December). Deep learning algorithms for rotating machinery intelligent diagnosis: An open source benchmark study. *ISA Transactions*, *107*, 224–255.

Zhao, Z., Zhang, Q., Yu, X., Sun, C., Wang, S., Yan, R., & Chen, X. (2021). Applications of unsupervised deep transfer learning to intelligent fault diagnosis: A survey and comparative study. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–28.

Zio, E. (2022). Prognostics and Health Management (PHM): Where are we and where do we (need to) go in theory and practice. *Reliability Engineering & System Safety*, *218*, 108119.