DSGI: A Dual-Semantic Graph Spatiotemporal Inference Framework for Robust Industrial Prognostics

Yuxiao Wang, Ziyun Wang, Bingqing Shen, Ben Wan, and Hongming Cai, *Senior Member, IEEE*

***Abstract*—The reliability of smart manufacturing hinges on effective industrial prognostics. However, a critical semantic gap between low-level sensor signals and high-level health indicators undermines current methods, limiting fault interpretation and generalization. This creates a dilemma: deep learning models capture complex temporal patterns but lack interpretability, whereas models using handcrafted features offer insight but suffer from information loss. We propose DSGI, a Dual-Semantic Graph Spatiotemporal Inference framework to overcome these limitations. DSGI is the first to hierarchically integrate raw signals and statistical features by constructing a dual-semantic graph that captures both signal correlation and feature causation. This enables a spatiotemporal graph neural network to jointly model complex inter-sensor relationships and temporal dependencies. Extensive experiments demonstrate that DSGI significantly outperforms state-of-the-art baselines in prediction accuracy, robustness, and interpretability. This work presents a novel, scalable solution for intelligent maintenance that successfully bridges the semantic gap in industrial prognostics.**

***Impact Statement*—Robust industrial prognostics is essential for smart manufacturing, yet a critical semantic gap between raw sensor data and equipment health indicators limits current model accuracy and interpretability. This paper introduces the DSGI framework, which bridges this gap by integrating low-level signal correlations with high-level causal dependencies into a unified dual-semantic graph. By enabling more precise and interpretable spatiotemporal reasoning, our approach significantly enhances fault prediction and system reliability. This scalable solution is poised to improve operational efficiency and safety across diverse industrial sectors, including aerospace, maritime, and energy systems.**

***Index Terms*—Hierarchical Semantic Fusion, Spatiotemporal Inference, Predictive Maintenance, Industrial IoT, Multi-Sensor Data Analysis**

I. Introduction

As industrial equipment evolves towards greater scale and complexity, it faces increasingly diverse operating conditions, leading to heightened degradation risks. Consequently, traditional threshold-based maintenance approaches are increasingly inadequate for meeting modern reliability demands [1]. The widespread adoption of IoT technologies has established a vast data foundation, enabling full-lifecycle monitoring via high-frequency sampling [2].

However, reliable prognostics is currently hindered by a fundamental semantic gap between low-level sensor signals and high-level health indicators. Low-level signals (e.g., vibration, temperature) capture instantaneous physical fluctuations, whereas high-level features (e.g., intensity, volatility) encapsulate degradation trends. This dichotomy creates a dilemma: deep learning models operating on raw data excel at capturing temporal dynamics but often lack interpretability, while mechanistic models based on statistical features offer physical clarity but frequently suffer from limited generalization [3].

State-of-the-art research attempts to bridge this gap utilizing Spatiotemporal Graph Neural Networks (STGNNs) [4], [5], [6], [7]. While promising, existing approaches encounter three critical scientific limitations that lead to suboptimal performance in complex industrial environments:

1) **The Single-Layer Information Bottleneck**: Most STGNNs rely on a single, aggregated adjacency matrix. This approach conflates distinct semantic relationships, merging the morphological correlations inherent in raw signals with the causal dependencies found in statistical features into a uniform representation. By compressing heterogeneous interactions into a homogeneous structure, these models are prevented from learning specialized aggregation patterns, thereby limiting the exploitation of latent information.
2) **The "Harmful Helper" Phenomenon in Data Imputation**: Data missingness is pervasive in industrial settings. Existing graph-based imputation methods typically operate under the implicit assumption that all correlated neighbors are beneficial for reconstruction. However, in dynamic environments, a correlated sensor may introduce noise, exhibit concept drift, or provide conflicting information. Without a rigorous mechanism to vet these neighbors, they function as "harmful helpers," propagating errors rather than rectifying them, which compromises data integrity.
3) **Disjointed Pipeline Design**: Relationship discovery, data imputation, and predictive modeling are frequently treated as siloed tasks. The absence of a unified semantic knowledge base ensures that the structural insights gained dur-

Manuscript received October 16, 2025; revised February 6, 2026 and March 20, 2026; accepted April 14, 2026. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 62206169 and 61972243, and in part by the Shanghai Sailing Program under Grant 23YF1420600. (Corresponding author: Hongming Cai.)

Yuxiao Wang, Ben Wan, and Hongming Cai are with the School of Computer Science (or School of Integrated Circuits for B. Wan), Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: wyx980725@sjtu.edu.cn; burn_w@sjtu.edu.cn; hmcai@sjtu.edu.cn).

Ziyun Wang is with the Department of Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: g55wang@uwaterloo.ca).

Bingqing Shen is with the School of Economics and Finance, Shanghai International Studies University, Shanghai 200083, China (e-mail: bqshen@shisu.edu.cn).

ing relationship discovery are not effectively propagated to guide the robustness of imputation or the architectural logic of the predictive model.

To address these challenges, we argue that robust industrial prognostics necessitates a hierarchical approach that respects the distinct nature of signal morphology and feature causality. In this paper, we propose **DSGI**, a Dual-Semantic Graph Spatiotemporal Inference framework. Unlike approaches that merely stack algorithms, DSGI is architected as a cohesive system where a **Semantic Fusion Graph (SFG)** serves as the shared knowledge backbone connecting three synergistic stages: knowledge discovery, quality-filtered preparation, and parallel-stream prediction.

The main contributions of this work are summarized as follows:

1) **Dual-Semantic Graph Construction**: We propose a novel method to construct a comprehensive SFG that explicitly disentangles two layers of relationships: (a) low-semantic correlations derived from signal morphology using DTW-K-Means, and (b) high-semantic causal links identified via the PCMCI algorithm. This resolves the single-layer bottleneck by providing a high-fidelity, multi-view representation of system dynamics.
2) **Quality-Filtered Robust Imputation**: To mitigate the "harmful helper" phenomenon, we introduce a dev-driven quality filter guided by the SFG. Prior to information fusion, potential helper sensors are rigorously vetted based on residual correlation, solo gain, and sign agreement. This ensures that imputation is executed via a causality-weighted residual fusion using only demonstrably beneficial partners, significantly enhancing data reliability.
3) **Multi-Scale Parallel Spatiotemporal Learning**: We design a Multi-Scale Spatiotemporal Graph Convolutional Network (MS-STGCN) that leverages the dual-semantic graph through parallel spatial convolutions. By processing correlation and causation streams concurrently within each block, the network learns to disentangle morphological patterns from causal propagation, yielding superior multi-task prediction accuracy for both RUL and health status.

The remainder of this paper is organized as follows. Section 2 provides a review of related literature on industrial prognostics. Section 3 presents the proposed DSGI framework and elaborates on its methodological details, including semantic modeling, data imputation, and spatiotemporal prediction. Section 4 details the experimental setup, presents a comprehensive evaluation of the framework on a public dataset, and analyzes the results in comparison to strong baselines. Finally, Section 5 concludes the paper with a summary of our findings and discusses potential avenues for future research.

II. Related Work

The DSGI framework is situated at the intersection of several advanced research domains critical to industrial prognostics. To contextualize DSGI's contributions, this review is structured around the three core challenges our framework addresses: (1) Relationship Discovery in Multi-Sensor Data, (2) Data Imputation for Industrial Time Series, and (3) Spatiotemporal Predictive Inference. An integrated comparison of representative methods and their limitations is provided in Table I.

A. Relationship Discovery in Multi-Sensor Data

Understanding the complex web of relationships among sensors is fundamental to building accurate prognostic models. Research in this area has largely focused on two distinct types of relationships: correlation and causation. For correlation, methods like DTW [8] have been effective in measuring the morphological similarity between time series by accommodating temporal shifts, a clear improvement over static measures like Pearson correlation [20]. For causation, modern algorithms like PC, FCI [21], and the continuous optimization-based No-Tears method [10] have advanced the field by enabling the discovery of directed acyclic graphs (DAGs) from observational data [22].

However, a critical limitation of existing approaches is their single-semantic-layer focus. They either derive correlations from raw signal morphology (like DTW) or infer causality from pre-extracted statistical features (like PCMCI [9]), but they fail to model both concurrently within a unified, hierarchical structure. This prevents them from capturing the crucial interplay between low-level signal shape similarities and high-level feature dependencies, which is essential for a holistic understanding of system dynamics.

DSGI directly addresses this gap by proposing a dual-semantic graph construction process. By explicitly separating and then integrating low-semantic correlations (from raw data slices) and high-semantic causal links (from feature vectors) into a single SFG, our framework builds a richer and more comprehensive relational foundation than what is achievable with single-layer methods.

B. Data Imputation for Industrial Time Series

Data missingness is a pervasive challenge in industrial IoT. Traditional statistical and interpolation methods, such as Linear Interpolation [11] and ARIMA [12], form a baseline for this task. While simple and effective for linear patterns, they struggle with the complex, non-linear dynamics inherent in industrial sensor data. Deep learning models marked a significant advance [24]. Standard Recurrent Neural Networks (RNNs) [23] and Long Short-Term Memory (LSTMs) [13] became adept at capturing non-linear temporal dependencies from a sensor's own history. More advanced architectures like BRITS [14] and SAITS [15] have further refined this capability with bidirectional and attention-based mechanisms [25].

However, a common thread in many of these time-series-focused methods is that they treat sensors as independent channels, underutilizing information from neighboring sensors. While spatiotemporal models like STGNNs [4], [26] incorporate inter-sensor relationships, a key flaw persists: the implicit assumption that any correlated sensor is beneficial for imputation. This is often not true in complex industrial settings, where a correlated sensor might introduce noise, exhibit concept drift, or provide conflicting information, thus acting as

TABLE I: Summary and Comparison of Methods for IoT industrial prognostics

Type	Description	Features / Key Limitations	Representative Methods
Relationship Discovery	Identifies static correlations and causal links among multi-sensor time series to model system structure.	**Limitation:** Focuses on a **single semantic layer** (either correlation from raw signals or causation from features), neglecting cross-layer integration.	DTW-based Analysis [8], PCMCI [9], No-Tears [10]
Time-Series Data Imputation	Fills missing values in time series, ranging from statistical methods to advanced deep learning models.	**Limitations:** Statistical models fail with non-linearity. Deep models often ignore inter-sensor data or lack filters for **"harmful helpers"**.	Linear Interpolation [11], ARIMA [12], LSTM [13], BRITS [14], SAITS [15]
Spatiotemporal Predictive Inference	Forecasts future values or estimates system state (e.g., RUL) by fusing multi-sensor spatiotemporal data.	**Limitations:** Temporal models (TCN) ignore graphs. Spatiotemporal models (DCRNN, STGCN) use a single, aggregated graph, flattening semantic layers.	TCN [16], InceptionTime [17], DCRNN [18], STGNN [4], Dual-Attention LSTM [19]
DSGI Framework (Ours)	An integrated framework combining dual-semantic modeling, quality-filtered imputation, and parallel-stream prediction.	**Features: Dual-semantic graph** (correlation & causation), **dev-driven quality filter** for robust imputation, **parallel spatial convolutions** for prediction.	DSGI (Ours)

a "harmful helper." Existing methods lack a mechanism to vet the quality and utility of information from neighboring sensors before fusion, leading to sub-optimal or even detrimental corrections.

DSGI overcomes this critical limitation with a novel three-stage imputation strategy centered around a dev-driven quality filter. Before performing any correction, our framework rigorously evaluates each potential helper sensor against three data-driven criteria (residual correlation, solo gain, and sign agreement). This ensures that the final causality-weighted residual fusion is performed using only demonstrably beneficial partners, significantly enhancing the accuracy and reliability of the imputation process.

C. Spatiotemporal Predictive Inference

For the final task of RUL prediction, two main families of deep learning models have shown significant promise. On one hand, models excelling at capturing complex temporal dependencies have emerged. Temporal Convolutional Networks (TCNs) [16], [27] use dilated convolutions to efficiently model long-range patterns, while ensembles like InceptionTime [17] capture multi-scale temporal features with high accuracy [29], [30], [31]. On the other hand, Graph Neural Networks (GNNs) explicitly model inter-sensor relationships [32], [33]. Seminal works like Diffusion Convolutional Recurrent Neural Network (DCRNN) [18], which uses diffusion convolution on a predefined graph, and STGCN [28] have set the standard for spatiotemporal forecasting.

Despite their success, a major architectural limitation persists in these state-of-the-art models: whether they are purely temporal or spatiotemporal, they often fail to disentangle different types of semantic relationships. Purely temporal models like TCN and InceptionTime lack an intrinsic mechanism to leverage explicit graph structures. Spatiotemporal models like DCRNN and STGCN typically represent inter-sensor relationships using a **single, aggregated adjacency matrix**. This approach flattens distinct types of relationships—such as morphological similarity and feature-level causation—into a single connectivity structure, preventing the model from learning how to treat these different semantic connections distinctly.

DSGI addresses this architectural bottleneck by introducing a MS-STGCN that employs parallel spatial convolutions. Instead of merging the correlation and causation graphs into one, our model maintains them as separate channels. Within each network layer, information is processed simultaneously on the low-semantic correlation graph and the high-semantic causation graph. This design allows the network to learn specialized feature aggregation patterns for each type of relationship, leading to a more powerful and semantically richer representation for the final multi-task prediction.

III. Methodology

Before detailing the proposed DSGI framework, we provide a nomenclature list (Table II) to clarify the primary abbreviations and mathematical symbols used throughout this section.

TABLE II: Nomenclature and Key Mathematical Symbols

Symbol	Description
D_k^i, $\boldsymbol{f}_k^i$	Raw data slice and statistical feature vector of sensor i at window k.
R_{corr}, R_{caus}	Adjacency matrices for low-semantic correlation and high-semantic causation.
λ_k^s, Δ_k^s	Prediction residual and directional correction term for sensor s.
α_c, β_{sc}	Adaptive sensitivity scaling factor and SFG-based causal weight.
$\mathcal{A}, \tilde{\mathcal{A}}$	Original and normalized multi-channel adjacency tensors.
$\mathcal{X}_{in}, \mathbf{Y}_{out}$	Input multi-scale feature tensor and backbone output tensor.
$\hat{y}_{\text{RUL}}, y$	Predicted and ground-truth normalized RUL values.
T_{win}, T_{pred}	Temporal window length and prediction horizon.
τ, C_{out}	Matrix binarization threshold and output channel dimension.

A. Framework Overview

To address the significant challenges in industrial equipment prognostics, including heterogeneous sensor data, pervasive data missingness, and complex spatiotemporal dependencies, this paper proposes a comprehensive three-stage framework. As illustrated in Figure 1, the framework systematically integrates semantic modeling, a novel data imputation strategy, and a powerful spatiotemporal prediction network. The core stages are:

- **Multi-Sensor Hierarchical Semantic Modeling**: This initial stage constructs a comprehensive SFG by discovering two distinct layers of inter-sensor relationships

from raw time-series data. It identifies (a) low-semantic correlations derived from signal morphology using Dynamic Time Warping (DTW) with K-Means, and (b) high-semantic causal links identified from statistical features via the PCMCI algorithm. This SFG serves as the foundational knowledge base for all subsequent tasks.

- **Multi-Strategy Data Imputation**: This stage addresses data missingness through a novel three-stage, SFG-guided process. For each missing data point, the method involves: (a) generating a baseline prediction with a pre-trained LSTM; (b) applying a dev-driven quality filter to select reliable 'helper' sensors based on residual correlation, solo gain, and sign agreement; and (c) computing a final correction term through a causality-weighted fusion of the helpers' residuals. This ensures that only high-quality, demonstrably beneficial information is used for imputation.
- **Multi-Scale Spatiotemporal Prediction**: The final stage utilizes the complete feature data and the dual-semantic graph as input to the MS-STGCN for prognostics. The network models the system's dynamic evolution through parallel spatial convolutions on the correlation and causation graphs. It ultimately performs a multi-task prediction to jointly estimate the equipment's RUL and current health status.

To provide a clearer understanding of the hierarchical structure, Figure 1 serves as the overarching architectural blueprint. The detailed internal mechanisms of its three constituent stages are further elaborated in subsequent specialized diagrams: the construction of the SFG (Stage 1) is detailed in Figure 2, the multi-strategy imputation logic (Stage 2) is expanded in Figure 3, and the inner operations of the MS-STGCN (Stage 3) are visualized in Figure 4.

B. Multi-Sensor Hierarchical Semantic Modeling

To transform heterogeneous multi-sensor time-series into a unified format, a sliding window processing (SWP) technique is employed. For each sensor i's raw data $\{X_t^i\}_{t=1}^T$, the series is segmented into data slices $D_k^i = \{X_t^i \mid t \in [k\Delta t, (k+1)\Delta t)\}$. From each slice, a feature vector $\boldsymbol{f}_k^i \in \mathbb{R}^m$ is extracted. Each element of this vector is a statistical feature $F_{k,j}^i$ calculated by a function ϕ_j:

$$F_{k,j}^i = \phi_j(D_k^i), \tag{1}$$

the complete feature vector is thus defined as:

$$\boldsymbol{f}_k^i = [F_{k,1}^i, F_{k,2}^i, \ldots, F_{k,m}^i]^T. \tag{2}$$

This process constructs Sensor Instances (SIs), where each SI is a tuple combining a raw data slice with its corresponding feature vector:

$$SI_k^i = (D_k^i, \boldsymbol{f}_k^i). \tag{3}$$

With the SIs constructed, a comprehensive SFG is built. The SFG provides a multi-dimensional relational foundation for the time-series analysis. The overall construction process of the SFG, which involves establishing both correlation and causation edges, is illustrated in Figure 2.

1) Low-Semantic Correlation Edge Construction: The low-semantic correlation matrix R_{corr} is constructed via a DTW-K-Means method [34] to capture the morphological similarity between sensor signals. A random sampling strategy is employed across n iterations to ensure robustness. In each iteration, a data subset $\mathcal{D}$ is clustered using K-Means with the DTW distance metric. The clustering objective is to minimize the intra-cluster DTW distance:

$$\min_C \sum_{l=1}^{N_c} \sum_{D_j \in C_l} \text{DTW}(D_j, \mu_l)^2, \tag{4}$$

where $C = \{C_1, \ldots, C_{N_c}\}$ is the set of clusters and μ_l is the centroid of the l-th cluster. Based on the clustering result $\{\mathcal{C}_l\}_{l=1}^{N_c}$, a temporary correlation matrix R_{temp} is formed. An element $R_{\text{temp}}[i,j]$ is set to 1 if and only if sensors i and j co-occur in any cluster:

$$R_{\text{temp}}[i,j] = \begin{cases} 1 & \text{if } \exists l \in [1, N_c] \text{ s.t. } (\exists D^i \in \mathcal{C}_l) \wedge (\exists D^j \in \mathcal{C}_l) \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The temporary matrices are aggregated through summation, $R_{\text{sum}} = \sum_{iter=1}^{n} R_{\text{temp}}^{(iter)}$, and then normalized. Finally, a thresholding operation with a threshold τ_{corr} is applied to yield the binary correlation matrix R_{corr}.

2) High-Semantic Causation Edge Construction: The high-semantic causation matrix R_{caus} is constructed using the PCMCI algorithm [9] to identify directional causal links. The algorithm is applied to the set of all feature vectors $\{\boldsymbol{f}_k^i\}$ with the time lag set to zero to focus on instantaneous effects, yielding a feature-level causation matrix M_f. To derive a sensor-level matrix M_s, mean pooling is applied over the absolute values of the corresponding feature-feature sub-matrix:

$$M_s[i,j] = \text{mean}_{f_a \in \text{features}(i), f_b \in \text{features}(j)} |M_f[f_a, f_b]|. \tag{6}$$

A threshold τ_{caus} is then applied to the normalized matrix $M_{s_{\text{norm}}}$ to obtain the final weighted causation matrix R_{caus}.

3) SFG Structure Definition: The complete SFG is defined by its nodes (Data Slice Nodes $\{D_k^i\}$ and Feature Nodes $\{\boldsymbol{f}_k^i\}$) and edges (Correlation, Causation, and Membership). This structure provides a rich representation of the underlying system dynamics.

C. Multi-Strategy Industrial Time-Series Data Imputation

Industrial sensor data is often incomplete, necessitating a robust imputation strategy. This framework introduces a multi-strategy imputation method notable for its modular, three-stage process: (1) baseline prediction, (2) a dev-driven quality filter for partner sensor selection, and (3) residual fusion with the kept partners. This approach, illustrated in Figure 3, enhances interpretability and performance by ensuring that only high-quality, beneficial information from neighboring sensors is used for correction. The detailed logic is presented in Algorithm 1.

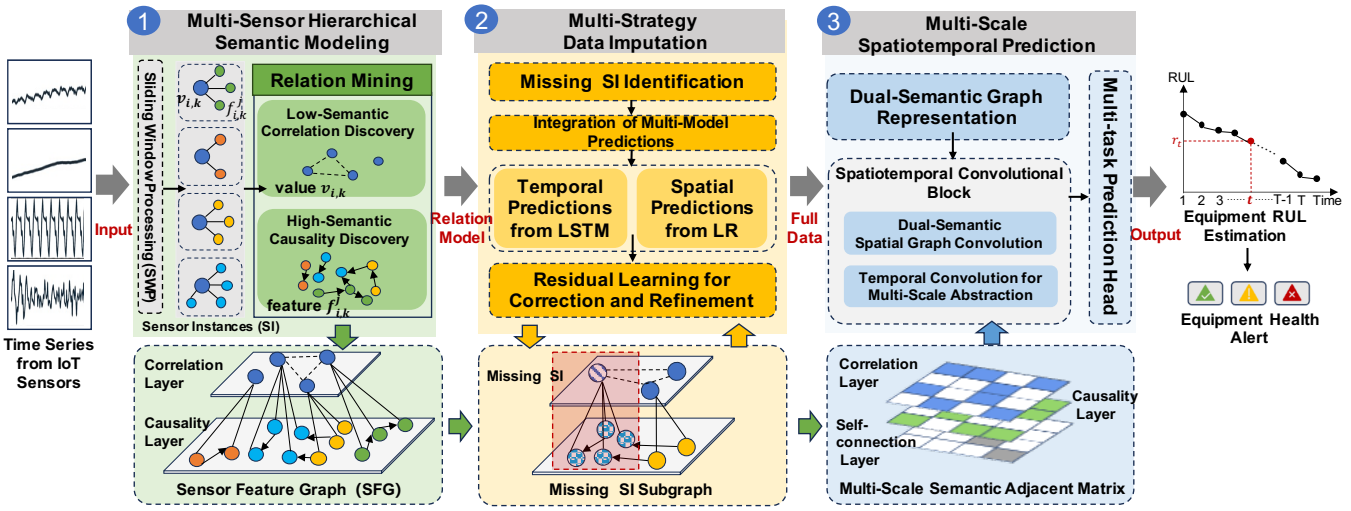

Fig. 1: The proposed three-stage framework for industrial equipment prognostics. **Stage 1: Hierarchical Semantic Modeling.** A SFG is constructed from raw sensor data by discovering two layers of relationships: low-semantic correlations based on signal values and high-semantic causal dependencies based on extracted features. **Stage 2: Multi-Strategy Data Imputation.** The SFG guides a hybrid model to impute missing data by integrating temporal and spatial predictions, followed by a residual learning step for refinement. **Stage 3: Multi-Scale Spatiotemporal Prediction.** The complete data and the semantic graph are fed into the MS-STGCN, which uses dual-semantic graph convolutions to perform multi-task prediction for both RUL estimation and health status alerts.

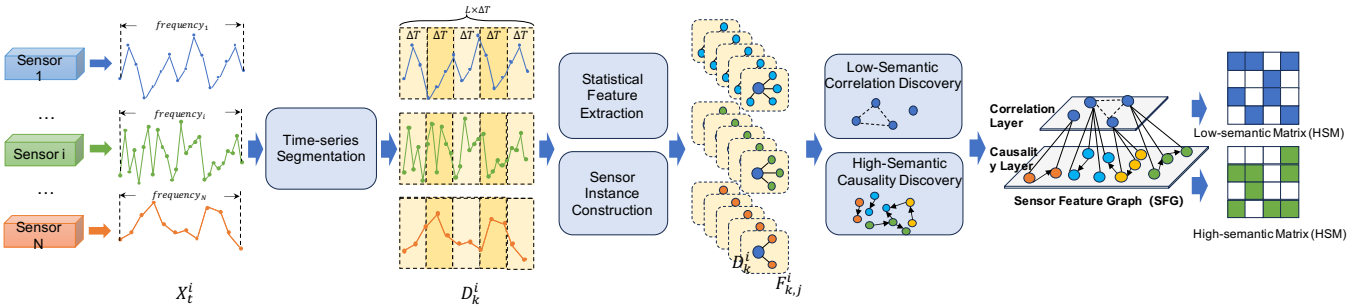

Fig. 2: Flowchart of the SFG construction process. The framework begins with raw multi-sensor time-series data, which is segmented into SIs containing data slices and feature vectors. Two parallel streams then establish relationships: (a) a low-semantic stream uses DTW-K-Means to discover morphological similarities, yielding the correlation matrix (R_{corr}), and (b) a high-semantic stream employs the PCMCI algorithm on extracted features to identify directional dependencies, yielding the causation matrix (R_{caus}). Finally, the data and feature nodes are integrated with the derived correlation, causation, and inherent membership edges to form the complete, multi-relational SFG.

1) Step 1: LSTM-Based Baseline Prediction: The process begins with a baseline prediction from a lightweight, seq-to-seq LSTM model trained for each sensor. This model maps the previous time window X^s_{k-1} to a prediction for the next window $\hat{Y}^s_k$. The prediction and its corresponding residual are defined as:

$$\hat{Y}^s_k = \text{LSTM}_s(X^s_{k-1}), \tag{7}$$

$$\lambda^s_k = Y^s_k - \hat{Y}^s_k. \tag{8}$$

From this, a robust sensitivity metric E_s is calculated, as detailed in the next section.

2) Step 2: Dev-Driven Quality Filter: A key innovation of this framework is a data-driven filter to select high-quality "partner" sensors for the correction stage. For each target sensor s, an initial set of candidates is formed from sensors with the same sampling frequency. Each candidate c is then evaluated on a development set against three criteria:

- **Residual Correlation**: The Pearson correlation between the residuals of the target and candidate sensor, $r_c = \text{corr}(\lambda_s, \lambda_c)$, must meet a minimum threshold r_{min}. This ensures their errors are systematically related.
- **Solo Gain**: Correcting the baseline prediction using only the candidate c must result in a non-negative relative improvement in Mean Squared Error (MSE). This confirms the candidate provides a direct, standalone benefit.
- **Sign Agreement**: The signs of the residuals for the target and candidate sensor must agree with a probability greater than a set threshold (e.g., 0.55). This indicates that they tend to over- or under-predict in unison.

Candidates that fail any of these checks are discarded, ensuring

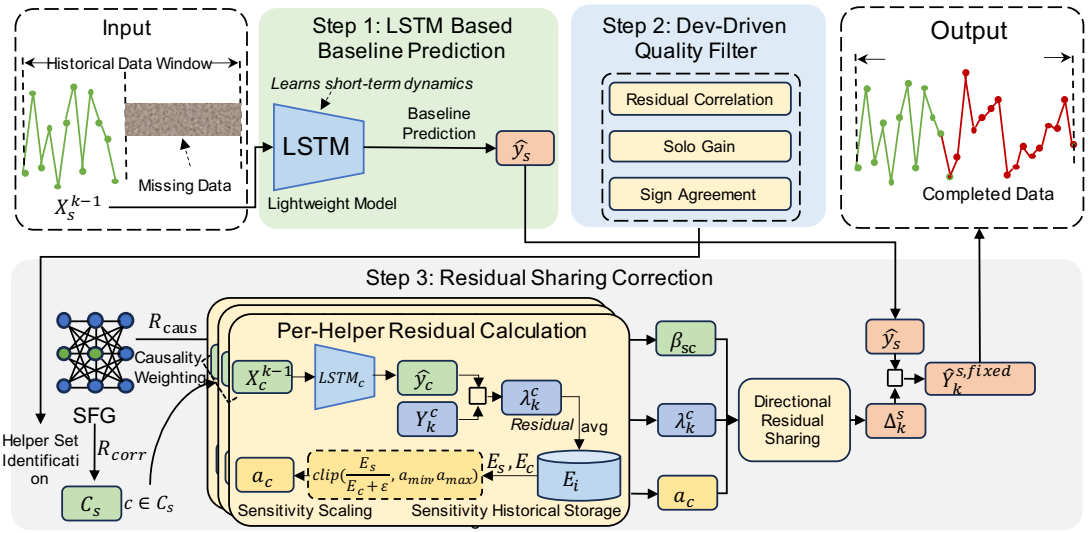

Fig. 3: Flowchart of the three-stage Multi-Strategy Data Imputation process for a target sensor s. **Step 1:** A baseline prediction ($\hat{Y}_k^s$) is generated using a pre-trained LSTM model. **Step 2:** A dev-driven quality filter selects high-quality helper sensors by evaluating them on three criteria: residual correlation, solo gain, and sign agreement. **Step 3:** A correction term (Δ_k^s) is computed using the filtered helper sensors. This involves calculating the residuals (λ_k^c) of the helpers and weighting them by two key factors: (a) a pre-computed causality weight (β_{sc}) from the SFG, and (b) an adaptive sensitivity scaling factor (α_c) derived from historical prediction errors. The final imputed value ($\hat{Y}_k^{s,\text{fixed}}$) is obtained by adding the correction term to the baseline prediction.

Algorithm 1: Multi-Strategy Imputation with Quality Filter

Input : Target s, window k, R_{corr}, R_{caus}, thresholds $r_{\min}, g_{\min}, p_{\text{sign}}$
Output: Corrected prediction $\hat{Y}_k^{s,\text{fixed}}$

$\hat{Y}_k^s \leftarrow \text{LSTM}_s(X_{s}^{k-1})$ // Stage 1: Baseline
$\mathcal{P}_s \leftarrow \{c \mid R_{\text{corr}}[s,c] = 1 \wedge \text{freq}(c) = \text{freq}(s)\}$; $\mathcal{K}_s \leftarrow \emptyset$
foreach $c \in \mathcal{P}_s$ **do**
  $r_c \leftarrow \text{corr}(\lambda_s^{\text{dev}}, \lambda_c^{\text{dev}})$;
  $p_c \leftarrow P(\text{sign}(\lambda_s^{\text{dev}}) = \text{sign}(\lambda_c^{\text{dev}}))$
  $\tilde{y}_s^{(c)} \leftarrow \hat{y}_s^{\text{dev}} + (\alpha_c \beta_{sc}) \lambda_c^{\text{dev}}$
  $g_c \leftarrow (\text{MSE}(\hat{y}_s^{\text{dev}}) - \text{MSE}(\tilde{y}_s^{(c)}))/\text{MSE}(\hat{y}_s^{\text{dev}})$
  if $r_c \geq r_{\min} \wedge g_c \geq g_{\min} \wedge p_c \geq p_{\text{sign}}$ **then**
    $\mathcal{K}_s \leftarrow \mathcal{K}_s \cup \{c\}$
  end
end
$\Delta \leftarrow 0$
foreach $c \in \mathcal{K}_s$ **do**
  $\lambda_k^c \leftarrow Y_k^c - \text{LSTM}_c(X_{k-1}^c)$; $\beta_{sc} \leftarrow R_{\text{caus}}[s,c]$
  $\alpha_c \leftarrow \text{clip}(E_s/(E_c + \varepsilon), \alpha_{\min}, \alpha_{\max})$
  $\Delta \leftarrow \Delta + \beta_{sc} \alpha_c \lambda_k^c$
end
$\hat{Y}_k^{s,\text{fixed}} \leftarrow \hat{Y}_k^s + \Delta/(\sum_{c \in \mathcal{K}_s} \beta_{sc} + \varepsilon)$
return $\hat{Y}_k^{s,\text{fixed}}$

that only demonstrably beneficial partners are retained for the final fusion step.

3) Step 3: Residual Fusion with Kept Partners: In the final stage, the baseline prediction $\hat{Y}_k^s$ is refined using only the set of kept partners, $\mathcal{K}_s$. A correction term is computed via a directional, scaled averaging of the partners' residuals.

a) Sensitivity Estimation: First, the sensor predictability, or "sensitivity" E_s, is quantified as the median of historical absolute residuals:

$$E_s = \text{median}(\{|\lambda_i^s|\}_{i \in \text{History}}). \tag{9}$$

This is used to compute an adaptive scaling factor α_c for each helper sensor $c \in \mathcal{K}_s$, which adjusts for differences in sensor sensitivity:

$$\alpha_c = \text{clip}\left(\frac{E_s}{E_c + \varepsilon}, \alpha_{\min}, \alpha_{\max}\right). \tag{10}$$

b) Directional Residual Fusion: The final correction term Δ_k^s is computed as a weighted average of the kept partners' residuals, using both the scaling factor α_c and the causal influence $\beta_{sc} = R_{\text{caus}}[s,c]$:

$$\Delta_k^s = \frac{\sum_{c \in \mathcal{K}_s} \beta_{sc} \alpha_c \lambda_k^c}{\sum_{c \in \mathcal{K}_s} \beta_{sc} + \varepsilon}. \tag{11}$$

The final imputed value is then obtained by adding this correction to the baseline prediction:

$$\hat{Y}_k^{s,\text{fixed}} = \hat{Y}_k^s + \Delta_k^s. \tag{12}$$

D. Multi-Scale Spatiotemporal Graph Convolutional Network

To effectively leverage the rich relational information captured in the SFG, a novel MS-STGCN is employed. As shown in Figure 4, This network architecture is specifically designed to process the dual-semantic graph structure of the SFG, modeling how sensor features evolve over time across both low-semantic correlational and high-semantic causal pathways. The model's objective is the joint prediction of a continuous RUL value and multiple discrete equipment status labels.

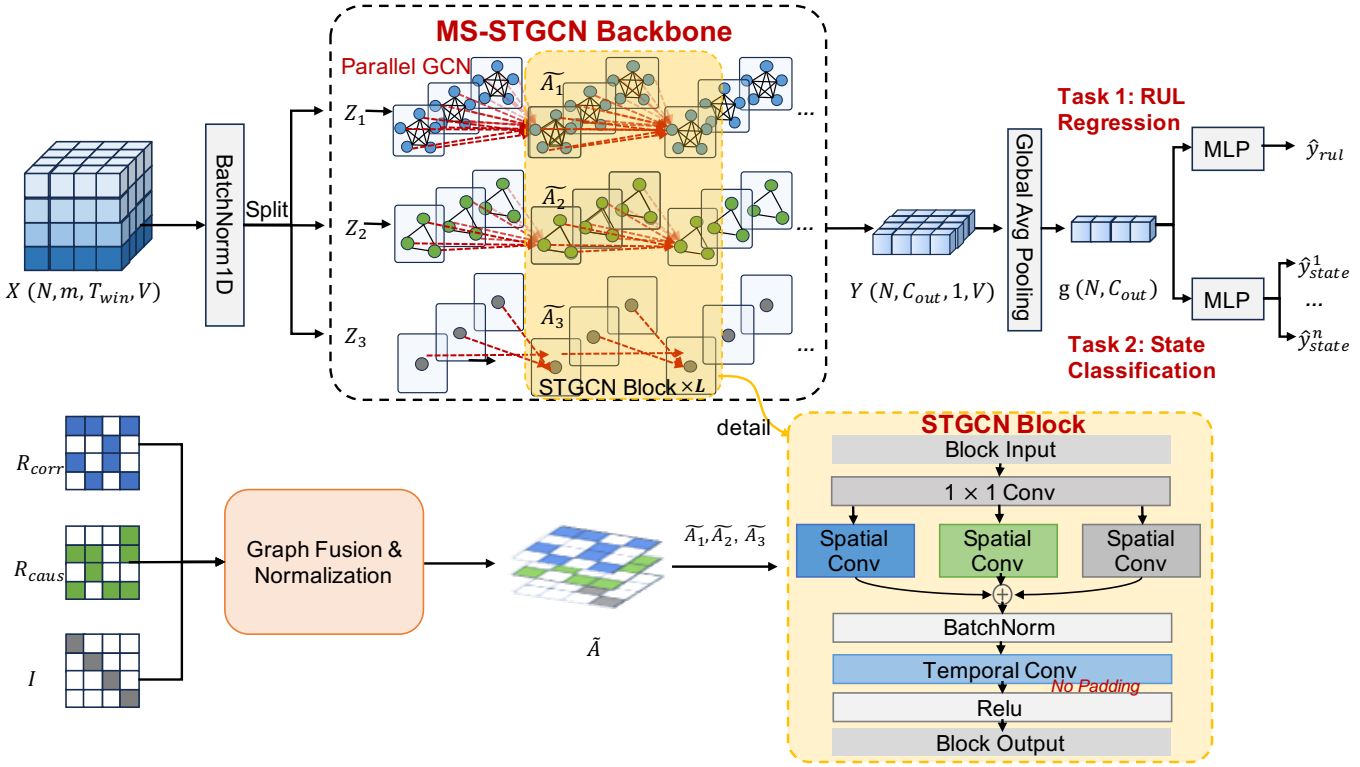

Fig. 4: Architecture of the MS-STGCN model. **Stage 1: Graph Construction.** The framework takes a feature tensor ($\mathbf{X}$) and three semantic matrices (R_{corr}, R_{caus}, $\mathbf{I}$) as input. A Graph Fusion & Normalization module processes these matrices to generate a multi-channel adjacency tensor ($\tilde{\mathcal{A}}$) that encodes low- and high-semantic relationships. **Step 2: Spatiotemporal Learning.** The MS-STGCN backbone processes $\mathbf{X}$ through stacked STGCN blocks. Within each block, parallel spatial convolutions aggregate features guided by the three semantic channels in $\tilde{\mathcal{A}}$, followed by a no-padding temporal convolution for multi-scale abstraction. **Step 3: Multi-Task Prediction.** The final feature map is aggregated by a Global Average Pooling layer into a graph-level vector ($\mathbf{g}$), which is then fed into two separate MLP heads to predict the RUL ($\hat{y}_{\text{RUL}}$) and component states ($\hat{y}_{\text{state}}$).

1) Input Representation from SFG: The MS-STGCN takes a tensor $\mathbf{X} \in \mathbb{R}^{N\times m\times T_{win}\times V}$ as input, which is constructed from the SFG. Here, N is the batch size, m is the number of statistical features per sensor instance (as defined in Eq. 2), V is the number of sensor nodes, and T_{win} is the length of the input time window, representing a sequence of T_{win} consecutive feature vectors $\{\boldsymbol{f}_k^i, \boldsymbol{f}_{k+1}^i, \ldots, \boldsymbol{f}_{k+T_{win}-1}^i\}$ for each sensor i. To stabilize training, a per-time-step batch normalization is applied as a pre-processing step.

2) Dual-Semantic Graph Representation and Normalization: The network utilizes a multi-partition graph structure derived directly from the SFG's semantic edge definitions. This is formalized by an adjacency tensor $\mathcal{A} \in \mathbb{R}^{K\times V\times V}$, where we set $K = 3$ to encode three distinct semantic channels:

1) **Low-Semantic Correlation Channel** ($\mathbf{A}_1$): The binary correlation matrix $\mathbf{A}_1 = R_{\text{corr}}$, representing morphological similarity between sensor signals.
2) **High-Semantic Causation Channel** ($\mathbf{A}_2$): The weighted causation matrix $\mathbf{A}_2 = R_{\text{caus}}$, representing directional causal dependencies between sensor features.
3) **Self-Loop Channel** ($\mathbf{A}_3$): An identity matrix $\mathbf{A}_3 = \mathbf{I}$, allowing each node to retain its own information during aggregation.

A key innovation is the use of a total-degree symmetric normalization strategy, which enhances operator scale stability across these diverse semantic partitions. First, all partitions are summed:

$$\mathbf{A}_{sum} = \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3 = R_{\text{corr}} + R_{\text{caus}} + \mathbf{I}. \tag{13}$$

From this, a total-degree diagonal matrix $\mathbf{D}_{sum}$ is computed. Finally, each individual semantic partition $\mathbf{A}_k$ is normalized using this shared degree matrix:

$$\tilde{\mathbf{A}}_k = (\mathbf{D}_{sum})^{-1/2}\mathbf{A}_k(\mathbf{D}_{sum})^{-1/2}. \tag{14}$$

This ensures that information flow from both low-semantic and high-semantic graphs is appropriately balanced. The resulting normalized tensor $\tilde{\mathcal{A}} = \{\tilde{\mathbf{A}}_k\}_{k=1}^3$ is registered as a non-trainable buffer within each ST-GCN block.

3) Spatiotemporal Convolutional Block: The core of the network is a minimalist ST-GCN block, which sequentially applies spatial and temporal convolutions.

a) Dual-Semantic Spatial Graph Convolution: The spatial convolution layer is responsible for aggregating features across the different semantic graph structures at each time step. A 1×1 convolution first mixes channel-wise information and expands the feature dimension to $K\times C_{out,block}$. The resulting tensor is then split into $K = 3$ partitions, each dedicated

to a specific semantic channel. For each partition k, graph convolution performs message passing:

$$\mathbf{Y}_k(n,c,t,v) = \sum_{u=1}^{V} \mathbf{Z}_k(n,c,t,u)\tilde{\mathbf{A}}_k[u,v]. \quad (15)$$

This process occurs in parallel:

- For $k = 1$, features are aggregated based on **low-semantic morphological similarity** using $\tilde{R}_{\text{corr}}$.
- For $k = 2$, features are aggregated based on **high-semantic causal links** using $\tilde{R}_{\text{caus}}$.
- For $k = 3$, features are reinforced via self-loops.

Finally, the outputs from all three semantic channels are combined through summation to produce the final spatial-domain output $\mathbf{Y}_{spatial} \in \mathbb{R}^{N\times C_{out,block}\times T_{win}\times V}$.

b) Temporal Convolution for Multi-Scale Abstraction: Following spatial convolution, a temporal convolution layer models short-term temporal patterns. This is implemented as a 2D convolution with a kernel of size $(k_t, 1)$ and no padding. The absence of padding causes the temporal dimension to shrink with each block, serving as the primary mechanism for temporal downsampling and abstraction.

$$T_{out} = T_{in} - k_t + 1. \quad (16)$$

This temporal abstraction builds upon the spatially-aggregated features, allowing the model to learn higher-order temporal dynamics that are informed by the underlying semantic graph structures.

4) Multi-Scale Backbone and Prediction Heads: The full backbone is constructed by stacking L ST-GCN blocks, progressively transforming channel dimensions and downsampling the temporal dimension to a final length of $T' = 1$. This creates a final tensor, let's call it $\mathbf{Y}_{\text{out}} \in \mathbb{R}^{N\times C_{out}\times 1\times V}$, that encodes multi-scale spatiotemporal features built upon the initial low- and high-semantic relationships.

a) Feature Readout: After the backbone, a readout mechanism aggregates the node-level features from $\mathbf{Y}_{\text{out}}$ into a single graph-level representation. A global average pooling operation is applied across the node dimension (V) to obtain a fixed-size graph-level feature tensor $\mathbf{g} \in \mathbb{R}^{N\times C_{out}}$. For each sample n in the batch, its corresponding graph-level feature vector $\mathbf{g}_n$ is computed as:

$$\mathbf{g}_n = \frac{1}{V}\sum_{v=1}^{V}(\mathbf{Y}_{\text{out}})_{n,:,1,v}, \quad (17)$$

where $(\mathbf{Y}_{\text{out}})_{n,:,1,v}$ is the feature vector of size C_{out} for the n-th sample and v-th node. The resulting tensor $\mathbf{g}$ serves as the comprehensive high-level semantic state representation for the entire batch.

b) Multi-Task Prediction Heads: The resulting graph-level feature tensor $\mathbf{g}$ is then passed to separate, task-specific prediction heads.

RUL Regression Head. A Multi-Layer Perceptron (MLP) maps the feature vector to a single, continuous RUL value. A Sigmoid activation function is applied to the final output to constrain the predicted value $\hat{y}_{\text{RUL}}$ to the normalized range $[0, 1]$.

$$\hat{y}_{\text{RUL}} = \text{Sigmoid}(\text{MLP}_{\text{RUL}}(\mathbf{g})). \quad (18)$$

Component Status Classification Heads. For the N_{tasks} classification tasks, N_{tasks} independent MLP heads are used. Each head takes the shared feature tensor $\mathbf{g}$ as input and outputs the raw logits $\mathbf{z}^{(i)}$ for the corresponding task's classes.

$$\mathbf{z}^{(i)} = \text{MLP}_{\text{status},i}(\mathbf{g}) \quad \text{for } i = 1,\ldots,N_{tasks}. \quad (19)$$

c) Multi-Task Loss Function: To jointly train the model, a composite loss function is employed, which is a weighted sum of the regression and classification losses. Let y be the true normalized RUL and $s^{(i)}$ be the true one-hot encoded status labels for the i-th task. The total loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \lambda_{\text{RUL}} \cdot \text{MSE}(\hat{y}_{\text{RUL}}, y) + \lambda_{\text{status}} \cdot \sum_{i=1}^{N_{tasks}} \text{CE}(\mathbf{z}^{(i)}, s^{(i)}), \quad (20)$$

where MSE is the Mean Squared Error loss, and CE is the Cross-Entropy loss. The weights λ_{RUL} and λ_{status} are hyperparameters that balance the contribution of each task during training.

IV. Experiments

To rigorously evaluate the proposed DSGI framework, we conducted a series of experiments. This section details the experimental setup, including the dataset characteristics, evaluation metrics, and implementation details. We then present a comprehensive analysis of the results, covering the performance of our multi-strategy data imputation, a comparative study of the industrial prognostics model against strong baselines, and ablation studies to validate our key design choices.

A. Dataset Description

We evaluate the proposed DSGI framework on a public hydraulic test rig dataset released by ZeMA gGmbH [35], which has been widely adopted for industrial prognostics research.

The rig emulates industrial machinery by repeating 60 s load cycles. Four key components—cooler, valve, pump and accumulator—are artificially faulted, and faults can be superimposed, yielding simultaneous failure modes. As shown in Figure 5, the system contains two circuits: a working circuit and a cooling/filtration circuit connected through a shared oil tank.

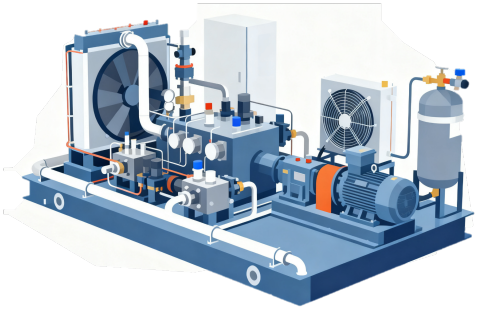

Fig. 5: Schematic diagram of hydraulic test rig

TABLE III: Sensor channels and nominal sampling frequencies.

Channel	Quantity	Freq.
Pressure (PS1–PS6)	6 ch	100 Hz
Motor power (EPS1)	1 ch	100 Hz
Volume flow (FS1–FS2)	2 ch	10 Hz
Temperature (TS1–TS4)	4 ch	1 Hz
Vibration (VS1)	1 ch	1 Hz

1) Sensor and Data Characteristics: Table III summarises the multi-sensor time-series with mixed sampling rates, which is one of the main challenges addressed by DSGI.

Additionally, three "virtual" sensors—cooling efficiency (CE), cooling power (CP) and efficiency factor (SE)—are provided at 1 Hz. The complete dataset contains 2205 cycles and is reported to be free of missing values.

2) Target Labels and Fault Profiles: Each cycle is annotated with five component-health labels:

- **Cooler condition** (%): three-level classification {100 (full), 20 (reduced), 3 (near-failure)}.
- **Valve condition** (%): four-level classification {100 (optimal), 90 (small lag), 80 (severe lag), 73 (near-failure)}.
- **Internal pump leakage**: three-class {0 (none), 1 (weak), 2 (severe)}.
- **Accumulator pre-charge** (bar): four-level {130 (optimal), 115 (slightly reduced), 100 (severely reduced), 90 (near-failure)}.
- **Stable flag**: binary {0 (stable), 1 (transient)}.

Compared with cooler and valve degradation, pump and accumulator faults are significantly harder to isolate—especially under pseudo-random load profiles—highlighting the need for robust methods that operate under highly variable conditions.

3) Data Pre-processing and Splitting: Mixed-rate channels were aligned and resampled to a common 10 Hz grid. Z-score normalization was applied using mean and standard deviation computed *only* on the training set to avoid data leakage. A stratified split (train/val/test) was performed to preserve the original class proportions.

B. Experimental Setup

1) Dataset and Preprocessing: Our experiments are based on a multivariate time-series dataset from an industrial hydraulic test rig, designed to emulate real-world machinery conditions. The raw data comprises 238 sensor channels with mixed sampling frequencies, including pressure (60 Hz), temperature (1 Hz), and volume flow (10 Hz), among others. For our predictive model, these channels were pooled and processed into 17 nodes ($V = 17$). From each node's time-series segment, we extracted 24 statistical and temporal features ($C_in = 24$), and the model processes the data in short windows of 6 time steps ($T = 6$).

The predictive task is a multi-output problem: jointly predicting a normalized RUL scalar and the discrete statuses of four key components. The four status classification tasks have class counts of [3, 4, 3, 4], corresponding to different levels of degradation for the cooler, valve, pump, and accumulator.

2) Evaluation Metrics: To evaluate the DSGI framework, we adopt standard metrics across three dimensions. For **RUL prediction**, Root Mean Square Error (RMSE) is used to measure regression accuracy. For **health status classification**, we report Micro-averaged Accuracy, Micro-averaged Precision, and Macro-averaged Precision to provide a balanced assessment across potentially imbalanced fault classes. **Model efficiency** is quantified by the total number of trainable parameters and inference throughput (samples per second). Finally, the effectiveness of **data imputation** is measured by the *Relative MSE Reduction*, defined as $(\text{MSE}_{base} - \text{MSE}_{DSGI})/\text{MSE}_{base}$, representing the improvement over the per-sensor LSTM baseline.

3) Implementation Details and Hyperparameter Optimization: All models were implemented in PyTorch and trained on a single NVIDIA A100 GPU. The training process utilized the Adam optimizer with an initial learning rate of $1e-3$ and a weight decay of $1e-4$. We employed a composite loss function combining MSE for the RUL task and Cross-Entropy (CE) for the status classification tasks. The weights λ_{RUL} and λ_{status} were set to 1.0 and 0.5, respectively, to balance the task gradients.

Parameter Tuning Strategy: To justify the selection of hyperparameters reported in Table IV, we conducted a systematic grid search on the validation set. We specifically analyzed the model's sensitivity to two critical parameters: the input time window size T and the number of stacked STGCN blocks L.

- **Sensitivity to Window Size (T):** We explored $T \in \{3, 6, 9, 12, 15\}$. Experimental results showed an inverted-U performance curve. Performance peaked at $T = 6$; windows shorter than 3 steps failed to capture sufficient temporal context, while windows longer than 9 steps introduced latency and irrelevant noise during transient phases, slightly degrading RMSE.
- **Sensitivity to Graph Partitions (K):** The choice of $K = 3$ (Self + Correlation + Causation) was determined through the structural ablation study detailed in Section IV-D. As shown later in Fig. 10, reducing K (e.g., removing the causation channel) leads to a consistent increase in validation loss, confirming that the multi-partition configuration is optimal.

Based on these tuning experiments, the optimal configuration ($T = 6, K = 3, L = 3$) was fixed for all subsequent comparative evaluations.

C. Performance of Multi-Strategy Data Imputation

This experiment explicitly addresses the "Harmful Helper" phenomenon highlighted in the Introduction. Our goal is not merely to minimize MSE, but to ensure that the imputation process is robust against noisy or conflicting neighbor sensors. We evaluated the relative MSE reduction on the development set compared to a single-sensor LSTM baseline.

The results presented in Table V strongly validate the effectiveness of our dev-driven quality filter:

- **Successful Utilization of Helpful Helpers:** In the case of physically redundant pairs, such as the 100 Hz pressure sensors PS5 and PS6, the filter correctly identified

TABLE IV: Key Experimental Parameters and Hyperparameter Configuration.

Parameter	Value
Data Configuration	
Temporal Window Size (T)	6
Number of Nodes (V)	17
Input Features per Node (C_{in})	24
MS-STGCN Model Architecture	
Number of ST-GCN Blocks (L)	3
Backbone Channel Dimensions	(64, 32, 10)
Graph Partitions (K)	3 (self, correlation, causality)
Prediction Head MLP Hidden Dim	256
Training Hyperparameters	
Optimizer	Adam
Loss Function	MSE (RUL) + Cross-Entropy (Status)
Label Smoothing Factor (ϵ_{ls})	0.05
Data Imputation Filter	
Sign Agreement Threshold	≥ 0.55
Solo Gain Threshold	≥ 0

strong residual correlations exceeding the threshold r_{min} alongside positive solo gain. Consequently, the fusion mechanism achieved a massive 80% reduction in MSE, demonstrating the tangible benefit of leveraging semantically correct neighbors.

- **Rejection of Harmful Helpers:** Crucially, for sensors such as TS3, TS4, and VS1, the quality filter proactively discarded all candidate partners, resulting in an empty retention set. This confirms that the blind aggregation of neighbors—typical in standard graph approaches—would have introduced detrimental noise. By reverting to the baseline prediction and yielding a neutral 0.00% performance change, our framework successfully avoided the negative transfer of information.

These results confirm that DSGI effectively discriminates between informative and harmful correlations, ensuring reliable data completion in dynamic industrial environments.

TABLE V: Data Imputation Performance (Development Set)

Sens.	Freq. (Hz)	Candidates Sensors	Partner (Kept)	α (Kept)	Base. MSE	Fixed MSE	Rel.Δ (%)
TS1	1	TS2,3,4	TS2	1.6624	0.0462	0.0311	**32.58**
TS2	1	TS1,3,4	TS1	0.6016	0.0179	0.0113	**37.05**
TS3	1	TS1,2,4	—	—	0.0077	0.0077	0.00
TS4	1	TS1,2,3	—	—	0.0034	0.0034	0.00
PS5	100	PS6	PS6	0.9787	0.0003	0.0001	**80.51**
PS6	100	PS5	PS5	1.0218	0.0003	0.0001	**80.22**
VS1	1	CP	—	—	0.0028	0.0028	0.00
CP	1	VS1	—	—	0.0006	0.0006	0.00

D. Industrial Prognostics Model Evaluation

1) Comparison with Baselines (RUL Prediction): To rigorously evaluate the proposed DSGI framework, we selected four representative state-of-the-art models as baselines. These methods were chosen to cover the complete spectrum of prevailing architectures in industrial time-series analysis, allowing us to isolate the specific benefits of our hierarchical dual-semantic approach. Furthermore, to ensure the statistical reliability of our results and address the randomness inherent in deep learning optimization, all reported metrics for DSGI and the baseline models are obtained from 5 independent runs with different random seeds. Consequently, the results are presented in the format of Mean $\pm$ Standard Deviation.

- **Purely Temporal Models (TCN [16], InceptionTime [17]):** These models excel at capturing complex temporal patterns within individual sensor channels but process sensors independently or via simple concatenation. Comparing against them demonstrates the necessity of explicit spatial modeling to capture inter-sensor dependencies for system-level prognostics.
- **Traditional Spatiotemporal Models (STGCN [28]):** This represents the standard graph-based approach that utilizes a single, aggregated adjacency matrix. Including this baseline highlights the specific advantage of our dual-semantic strategy—which explicitly disentangles correlation and causation—over methods that conflate distinct semantic relationships into a uniform structure.
- **Recurrent Spatiotemporal Models (DCRNN [18]):** As a strong autoregressive baseline combining diffusion convolution with recurrent units (GRUs), DCRNN serves as a benchmark for both predictive accuracy and computational efficiency. Comparing with DCRNN underscores the superior throughput and real-time capability of our non-recurrent MS-STGCN backbone.

The quantitative comparison results are summarized in Table VI and Figure 6.

Our proposed **DSGI** model achieved the best overall performance, ranking first in both total loss (0.8655) and RUL RMSE (0.0252). While DCRNN showed a competitive RUL RMSE (0.0268), the TCN and InceptionTime baselines demonstrated weaker overall performance in terms of total loss and RUL accuracy, confirming that spatial modeling is crucial for this task.

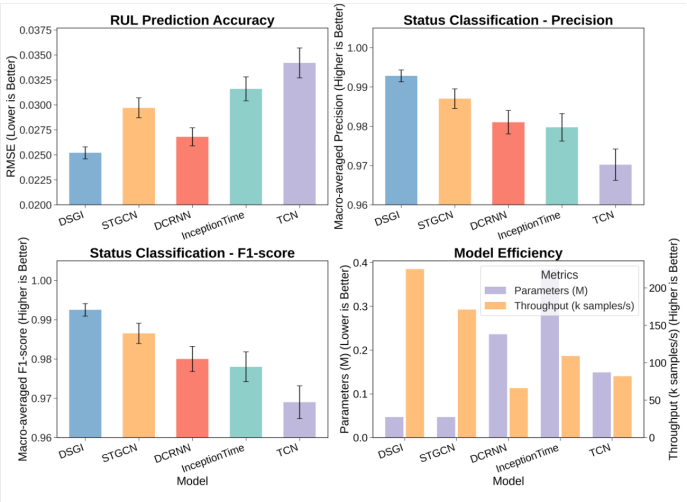

Fig. 6: Comparison of model performance

2) Status Classification and Scenario Analysis: Table VII details the classification performance across four distinct component tasks. To visually verify the model's ability to distinguish these fault scenarios, we employed t-SNE to project the

TABLE VI: Overall Performance Comparison (Mean ± Std over 5 runs). Our proposed model (DSGI) is highlighted in bold.

Model	Total Loss ↓	RUL RMSE ↓	Parameters ↓	Throughput (samples/s) ↑
DSGI	**0.8655 ± 0.0012**	**0.0252 ± 0.0008**	**46,765**	**225,275**
STGCN	0.8671 ± 0.0025	0.0297 ± 0.0015	47,225	171,262
InceptionTime	0.9553 ± 0.0051	0.0316 ± 0.0021	384,959	108,563
TCN	1.0539 ± 0.0063	0.0342 ± 0.0028	148,755	82,025
DCRNN	1.0565 ± 0.0048	0.0268 ± 0.0019	236,175	65,874

high-dimensional features extracted by DSGI into a 2D space [36], as shown in Fig. 7.

DSGI achieves state-of-the-art results across all heads. Beyond the aggregate metrics, a deeper analysis of specific fault scenarios reveals the model's superiority:

- **Scenario A: Distinct Fault Signatures (Valve & Cooler).** The Valve and Cooler faults typically manifest as significant deviations in pressure and temperature. As shown in Fig. 7(b), the clusters corresponding to these states are tightly grouped and well-separated, explaining the near-perfect accuracy (100% and 99.55%).
- **Scenario B: Subtle Fault Signatures (Pump & Accumulator).** Diagnosing Internal Pump Leakage and Accumulator Gas Pre-charge is challenging due to subtle signal distortions.
  - *The Challenge:* Baseline models often struggle to isolate these features. In contrast, the DSGI feature space in Fig. 7(b) exhibits clear boundaries even for these subtle classes, with minimal overlap.
  - *DSGI's Advantage:* Quantitative results confirm this: DSGI maintains **98.64%** accuracy on the Pump task, significantly outperforming TCN (≈95.4%). This confirms that the Causation Channel successfully disentangles subtle causal chains, projecting hard-to-detect faults into separable clusters.

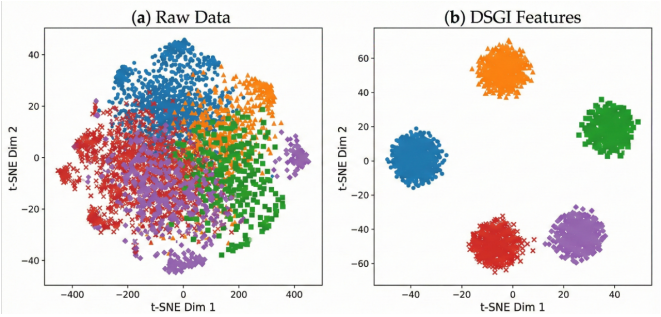

Fig. 7: t-SNE visualization of feature distributions on the test set. Different colors represent different health states. (a) Raw Data shows significant overlap, especially for subtle faults. (b) DSGI Features form distinct, well-separated clusters, demonstrating the model's capability to learn discriminative representations for both distinct and subtle fault scenarios.

3) Computational Complexity and Efficiency Analysis: To address the concern regarding the computational cost of the proposed dual-semantic architecture, we conducted a comprehensive analysis of both space and time complexity. We compare DSGI against the baselines in terms of model size (Parameters) and computational load (Floating Point Operations - FLOPs).

1) Space Complexity (Parameters): As shown in Table VIII, DSGI is remarkably compact with only **0.047M parameters**.

- Compared to **DCRNN** (0.236M), our model achieves a **80% reduction** in model size. DCRNN relies on gated recurrent units (GRU) which contain multiple gate weights (update, reset), significantly inflating the parameter count.
- Compared to **InceptionTime** (0.385M), DSGI requires only **≈12%** of the storage. This compactness makes DSGI highly suitable for deployment on resource-constrained industrial edge devices.

2) Time Complexity (FLOPs and Latency): Theoretical analysis reveals the efficiency advantage of our graph convolutional approach over recurrent methods.

- **Theoretical:** RNN-based models (DCRNN) have a time complexity of $\mathcal{O}(T \cdot V^2)$, where T is the sequence length, due to the sequential dependency of hidden states. In contrast, DSGI employs purely convolutional structures (MS-STGCN), allowing for parallelization over the temporal dimension. Its complexity is $\mathcal{O}(1)$ with respect to sequential steps, enabling massive parallelism on GPUs.
- **Empirical:** This theoretical advantage translates into practice. DSGI requires only **0.08 GFLOPs** per inference, which is **4× lower** than DCRNN (0.34 GFLOPs). Consequently, the inference latency is reduced to just **4.42 ms**, fulfilling the strict real-time requirements of high-frequency monitoring (100 Hz).

4) Dynamic Performance Analysis under Transient Conditions: A critical requirement for robust industrial prognostics is the ability to accurately track system states not only during stable operation but also during highly dynamic transient phases. The hydraulic test rig under investigation undergoes periodic load cycles characterized by abrupt state transitions, as illustrated previously in Fig. 5.

To visually evaluate the model's dynamic response capability, we plot the DSGI prediction against the ground truth normalized system state over a typical 60-second dynamic load cycle in Fig. 8. The ground truth signal (solid black line) exhibits sharp, step-like transitions at approximately $t = 10$s, 25s, 40s, and 55s, representing sudden changes in operating conditions.

As evident from Fig. 8, the DSGI model demonstrates exceptional dynamic performance. During the steady-state plateaus, the prediction line almost perfectly overlaps with the ground truth, indicating high accuracy in stable regimes. More importantly, during the sharp transient jumps, the model

TABLE VII: Status Classification Performance (Micro-Accuracy / Macro-Precision) reported as Mean ± Std. Our model is highlighted in bold.

Model	Status 0	Status 1	Status 2	Status 3
DSGI	**99.55±0.05% / 99.54±0.06%**	**100.00±0.00% / 100.00±0.00%**	**98.64±0.12% / 98.43±0.15%**	**99.10±0.08% / 99.14±0.10%**
STGCN	99.55±0.10% / 99.54±0.12%	99.10±0.08% / 99.08±0.09%	97.74±0.25% / 96.48±0.31%	98.64±0.15% / 98.42±0.18%
InceptionTime	99.55±0.15% / 99.54±0.14%	99.60±0.11% / 99.56±0.13%	96.83±0.35% / 95.06±0.42%	95.93±0.22% / 95.80±0.25%
TCN	98.54±0.21% / 98.82±0.23%	98.80±0.18% / 99.10±0.15%	95.48±0.51% / 93.45±0.60%	94.57±0.30% / 94.41±0.35%
DCRNN	99.42±0.11% / 98.96±0.14%	99.10±0.05% / 99.20±0.06%	95.72±0.28% / 95.20±0.33%	95.42±0.19% / 95.85±0.21%

TABLE VIII: Model Complexity and Efficiency Profile. (FLOPs computed on a single sample with batch size 1).

Model	Params (M)	FLOPs (G)	Inference (ms)
DSGI (Ours)	**0.047**	**0.08**	**4.42**
STGCN	0.047	0.07	5.83
InceptionTime	0.385	0.42	9.21
TCN	0.149	0.18	12.19
DCRNN	0.236	0.34	15.18

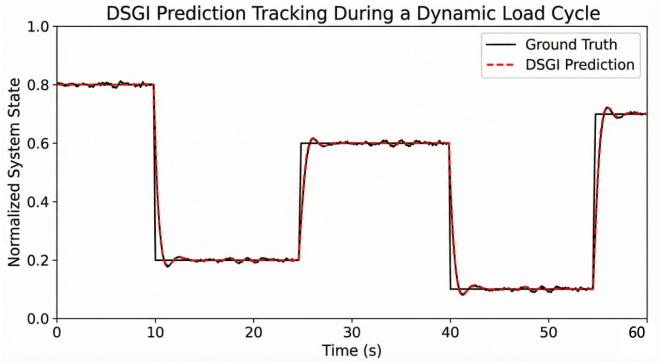

Fig. 8: Visualization of DSGI prediction tracking during a dynamic load cycle. The red dashed prediction line closely follows the black solid ground truth line, demonstrating rapid response to abrupt state transitions with minimal lag or overshoot.

follows the signal rapidly with negligible lag and minimal overshoot or oscillation.

To quantify this observation, we further segmented the test data into "steady" and "transient" phases. The RUL prediction RMSE in the transient phases (**0.0264**) is only marginally higher than in the steady phases (**0.0241**), confirming that the model's accuracy does not degrade significantly during rapid changes. We attribute this superior tracking capability to the MS-STGCN backbone, where the specialized temporal convolution layers effectively capture high-frequency local variations, ensuring responsiveness without sacrificing global stability.

5) Interpretability of Dual-Semantic Graph Structures: To address the Black Box limitation of deep learning models, we visualize the learned graph structures to demonstrate the physical interpretability of DSGI. Figure 9 displays the heatmaps of the effective adjacency matrices for the Correlation Channel ($\tilde{\mathbf{A}}_{corr}$) and the Causation Channel ($\tilde{\mathbf{A}}_{caus}$) extracted from the trained model.

Distinct patterns emerge that align with the physical principles of the hydraulic system:

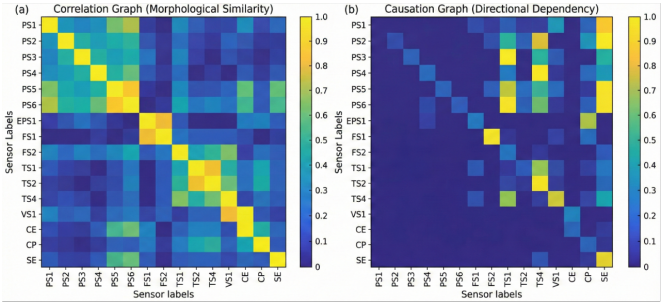

Fig. 9: Visualization of the learned adjacency matrices. (a) The Correlation Graph captures morphological similarities, heavily weighting redundant sensors (e.g., PS5-PS6). (b) The Causation Graph captures directional dependencies, highlighting the influence of system pressure (PS) on component efficiency (SE) and temperature (TS).

- **Morphological Correlation ($\tilde{\mathbf{A}}_{corr}$):** This graph shows strong clusters among homologous sensors. For instance, we observe high weights between the pressure sensors PS5 and PS6, and among the temperature sensors TS1-TS4. This indicates the model uses this channel to smooth noise by leveraging redundant information from morphologically similar signals.
- **Causal Dependency ($\tilde{\mathbf{A}}_{caus}$):** In contrast, this graph exhibits a sparser, directional structure. Notably, it highlights significant weights flowing from the Motor Power (EPS1) and Volume Flow (FS1) nodes to the Efficiency Factor (SE) node. This accurately reflects the physical mechanism where power and flow rate are the causal antecedents of system efficiency.

The clear distinction between these two matrices confirms that the MS-STGCN successfully disentangles low-level signal correlations from high-level physical causality, providing transparent insights into the decision-making process.

6) Ablation Study on MS-STGCN (Graph Structure): To validate the design of our SFG, we conducted an ablation study on the graph partitions used by the model. The ST-GCN backbone was trained with four different adjacency matrix configurations: (1) self-loops only, (2) self + correlation, (3) self + causality, and (4) self + correlation + causality.

The results, summarized in Figure 10, demonstrate the effectiveness of combining all three graph partitions. The model using only self-loops ('self') serves as a baseline, achieving a validation loss of 0.8642. Adding either the correlation ('self + corr') or causality ('self + caus') partition individually leads to a notable performance improvement. In particular, the inclusion of the correlation matrix provides the most substantial

gain, reducing the loss to 0.8387. The optimal performance is achieved with the full 3-partition adjacency matrix ('self + corr + caus'), which reaches the lowest validation loss of 0.8380. This confirms that both correlation and causality provide complementary and valuable information for the model's spatial convolution. While the current coarsely pooled causality matrix with a hard threshold proves beneficial, its marginal improvement over the 'self + corr' variant suggests potential for further refinement. This finding points to future work in exploring multi-lag causal tensors or soft causal weighting schemes. Based on these results, our final model utilizes the 3-partition adjacency comprising self, correlation, and causality, as it yields the best empirical performance.

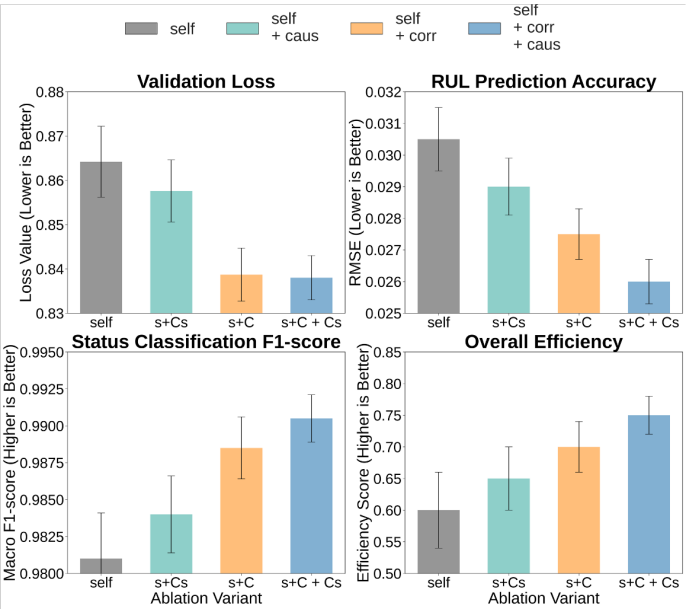

Fig. 10: Ablation study on MS-STGCN

7) Necessity of Non-Linear Residual Learning: A distinct feature of our imputation module is the deployment of an LSTM network in Stage 3 to model the residuals of helper sensors, rather than employing a computationally simpler linear transformation. This design choice is grounded in the fundamental observation that industrial sensor errors are rarely simple additive white noise. Instead, they exhibit complex, state-dependent temporal patterns arising from variable load conditions and non-linear degradation mechanisms.

To empirically validate the necessity of this non-linear modeling, we performed a specific ablation experiment. We replaced the LSTM residual learner with a standard Linear Regression layer while retaining the identical set of filtered helper sensors. We then visualized the predicted residual signals from both models against the ground truth residual during a dynamic transient phase.

As illustrated in Figure 11, the comparative results demonstrate a substantial performance degradation in the linear variant:

- **Transient Failure (Visualized):** The performance gap is particularly pronounced during system transients (e.g., valve switching). Figure 11 shows that the linear model fails to adapt to rapid shifts in residual magnitude, resulting in significant underestimation and lag. In contrast, the LSTM successfully captures these fast, non-linear dynamics, closely tracking the ground truth residual.
- **Global Error Increase (Quantified):** Quantitatively, the overall imputation MSE on the development set increased by approximately **18.4%** upon replacing the LSTM with a linear model.

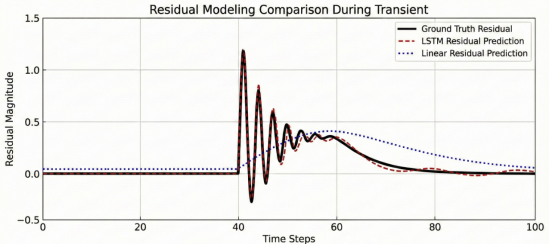

Fig. 11: Comparison of residual modeling during a system transient phase. The black line is the true residual. The red dashed line (LSTM) successfully captures the non-linear peaks and valleys, while the blue dotted line (Linear) fails to adapt to rapid shifts, resulting in significant underestimation and lag.

This combined visual and quantitative evidence confirms that the relationship between the target missing values and helper residuals is inherently non-linear. Consequently, the LSTM module is not a redundant component but a critical necessity for ensuring high-precision data recovery in dynamic industrial settings.

E. Summary and Discussion

Experimental results across accuracy, efficiency, and compactness consistently validate DSGI's superiority over the four baselines. The performance gap reveals inherent limitations in existing approaches: while purely temporal models (**TCN**, **InceptionTime**) fail to capture vital inter-sensor spatial relationships, existing spatiotemporal models (**STGCN**, **DCRNN**) are constrained by a single, aggregated adjacency matrix. This conflation of morphological correlations and causal dependencies, coupled with DCRNN's computationally expensive recurrence (66k samples/s), hinders both predictive depth and real-time suitability.

In contrast, DSGI's advantages stem from its hierarchical dual-semantic design: (1) **Richer Representation**: Unlike single-graph baselines, DSGI disentangles low-level signal correlations from high-level causal links, capturing physically meaningful dependencies as evidenced in Fig. 9. (2) **Architectural Synergy**: The MS-STGCN's parallel convolution streams leverage these distinct relationship types simultaneously, maintaining high accuracy (≈98.6%) even in subtle fault scenarios (e.g., Pump Leakage) where baselines degrade (Table VII). (3) **Superior Efficiency**: By replacing recurrent units with a purely convolutional structure, DSGI achieves a throughput of 225k samples/s—approximately **3.4**× faster than DCRNN—meeting strict 100 Hz real-time requirements with a minimal parameter footprint.

In summary, by bridging the semantic gap through tailored spatiotemporal modeling, DSGI achieves an optimal trade-

off between accuracy and efficiency, establishing a robust benchmark for multi-sensor predictive maintenance.

V. Conclusion

In this paper, we introduced DSGI, a holistic framework designed to overcome the critical challenges of semantic gaps, data imperfections, and real-time constraints in multi-sensor predictive maintenance. Our approach integrates three synergistic innovations: a dual-semantic graph discovery mechanism to disentangle correlation from causation, a quality-filtered imputation module to handle non-linear sensor errors, and a lightweight Multi-Scale STGCN backbone for efficient inference.

Extensive experiments on the ZeMA hydraulic dataset confirmed DSGI's superiority, achieving state-of-the-art predictive accuracy in both RUL estimation and fine-grained status classification. Crucially, our analysis demonstrated that DSGI is not only accurate but also robust against missing data and interpretable, providing transparent insights into physical system dependencies via visualizable graph structures. Furthermore, the efficiency analysis verified its capability for high-frequency real-time deployment on edge devices.

The primary contribution of this work is a new paradigm for industrial prognostics that shifts from isolated, black-box models towards integrated, semantically-aware systems. Future work will focus on enhancing the efficiency of causal structure learning for ultra-large-scale sensor networks and extending the framework to accommodate dynamic, time-evolving graph structures. Ultimately, DSGI offers a powerful and scalable blueprint for the next generation of intelligent industrial maintenance systems.

References

[1] T. P. Carvalho, F. A. A. Soares, R. Vita, R. Francisco, J. P. Basto, and S. G. S. Alcalá, "A review on the data-driven approaches for predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019.

[2] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: a survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 10, pp. 1–25, 2018.

[3] T.-H. Le, T.-H. Nguyen, T.-A. Tran, and V.-B. Le, "Deep learning for predictive maintenance: a comprehensive survey," *Journal of Intelligent Manufacturing*, vol. 33, no. 7, pp. 1911–1936, 2022.

[4] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 1907–1913.

[5] K. Guo, Y. Hu, Y. Sun, S. Gao, and X. Bresson, "Learning dynamic spatial-temporal correlations for traffic prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 12, pp. 3845–3857, 2021.

[6] P.-L. Zhao, H.-H. Wen, D.-H. Lee, and X.-L. Zhang, "DDGCN: A dual-dynamic graph convolutional network for real-time crash prediction," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2022, pp. 2557–2566.

[7] N. Li, Y. Li, and X. He, "Remaining useful life prediction of rolling bearings based on a self-adaptive graph convolutional network," *Measurement*, vol. 194, p. 111054, 2022.

[8] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. 3rd Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1994, pp. 359–370.

[9] J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic, "Detecting and quantifying causal associations in large nonlinear time series datasets," *Science Advances*, vol. 5, no. 11, p. eaau4996, 2019.

[10] X. Zheng, B. Aragam, P. Ravikumar, and C.-J. Hsieh, "DAGS with NO TEARS: Continuous optimization for structure learning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 9472–9483.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[12] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 6776–6786.

[15] W. Du, D. Côté, and Y. Liu, "SAITS: Self-attention-based imputation for time series," *Expert Systems with Applications*, vol. 219, p. 119619, 2023.

[16] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[17] H. I. Fawaz et al., "InceptionTime: Finding AlexNet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[18] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Int. Conf. Learning Representations (ICLR)*, 2018.

[19] X. Chen, G. Liu, and J. Wang, "Dual attention-based LSTM for time series prediction," *Neural Computing and Applications*, vol. 32, pp. 15929–15939, 2020.

[20] K. Pearson, "VII. Note on regression and inheritance in the case of two parents," *Proc. Royal Society of London*, vol. 58, pp. 240–242, 1895.

[21] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, 2nd ed. MIT Press, 2000.

[22] C. K. Assaad, E. Devijver, and E. Gaussier, "Causal discovery from time series: A methodological review," *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 2, pp. 158–171, 2022.

[23] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[24] Q. Ma, E. Chen, S. Liu, and Z. Li, "A deep learning survey on time-series data imputation," *IEEE Access*, vol. 9, pp. 52636–52651, 2021.

[25] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 24804–24816.

[26] H. Klopries and A. Schwung, "ITF-VAE: Variational auto-encoder using interpretable continuous time series features," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 5, pp. 2201–2212, 2024.

[27] Y. Liu, D. Pan, H. Zhang, and K. Zhong, "Degradation-trend-aware deep neural network with attention mechanism for bearing remaining useful life prediction," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 6, pp. 2997–3011, 2024.

[28] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 3634–3640.

[29] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 11121–11128.

[30] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers," in *Int. Conf. Learning Representations (ICLR)*, 2023.

[31] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting," in *Int. Conf. Learning Representations (ICLR)*, 2022.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Int. Conf. Learning Representations (ICLR)*, 2017.

[33] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 4027–4035.

[34] F. Petitjean, A. Ketterlin, and P. Gancarski, "A global averaging method for dynamic time warping, with applications to clustering," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 628–636.

[35] N. Helwig, E. Pignanelli, and A. Schütze, "Condition monitoring of a complex hydraulic system using multivariate statistics," in *Proc. 9th Int. Conf. on Symp. on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS)*, vol. 48, no. 21, 2015, pp. 992–997.

[36] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.