

Article

xLSTM-m for Multivariate Structural Response Classification in Bridge Monitoring via Masked Mean Pooling

Truong T. N. Lien ¹, Le Van Vu ², Do Hong Phuc ¹, Do Duc Tho ³, Ly Hoang Mai ¹, Nguy Phan Tin ³, Kwanil Lee ⁴ and Nguyen Thi Cam Nhung ^{2,*}

¹ Vietnam-Korea Institute of Science and Technology, Hoa Lac High Tech Park, Hanoi 100000, Vietnam; truongtnlien@gmail.com (T.T.N.L.); dohongphuc98@gmail.com (D.H.P.); lhmai1811@gmail.com (L.H.M.)

² Faculty of Civil Engineering, University of Transport and Communications, Hanoi 100000, Vietnam; vulv2704@gmail.com

³ Faculty of Engineering Physics, Hanoi University of Science and Technology, Hanoi 100000, Vietnam; tho.doduc@hust.edu.vn (D.D.T.); tin.nguyphan@hust.edu.vn (N.P.T.)

⁴ Nanophotonics Research Center, Korea Institute of Science and Technology, Seoul 02792, Republic of Korea; klee21@kist.re.kr

* Correspondence: ncnhung@utc.edu.vn

Abstract

Multivariate structural-response classification supports bridge monitoring by integrating synchronized measurements from multiple sensors and assigning an observation window to a predefined structural or loading condition. The main technical challenge is to compress temporally and spatially distributed sensor information into a fixed-length representation without losing transient or localized condition-sensitive features. However, previous studies have mainly focused on backbone architecture design, whereas the influence of sequence-level aggregation has rarely been isolated under a controlled backbone and training configuration. In this study, xLSTM-m is proposed as a modified variant of the extended Long Short-Term Memory (xLSTM) network for multivariate bridge-response classification. The conventional last-hidden-state readout is replaced with masked mean pooling over all valid hidden states, while the hybrid sLSTM/mLSTM backbone and the remaining training configuration are kept unchanged. This controlled design isolates the sequence-level aggregation strategy as the only architectural variable. A comparative evaluation is conducted using eight neural models. These models include xLSTM-m, the baseline xLSTM, and six Transformer-based architectures. Three bridge-monitoring datasets are considered: an FBG strain-response dataset and a PCB Piezotronics acceleration-response dataset acquired from a laboratory-scale cable-stayed bridge model, together with the field-scale Z24-9Setup acceleration benchmark. A strict five-fold cross-validation protocol is adopted. The proposed xLSTM-m ranks first on all three datasets. It achieves a mean accuracy of 95.29%, a mean F1-score of 95.57%, and the lowest inter-fold standard deviation of 0.84. Relative to the strongest Transformer baseline for each dataset, the corresponding accuracy gains are 6.30 percentage points for FBG, 3.66 percentage points for PCB, and 31.76 percentage points for Z24-9Setup. These results indicate that, for the evaluated bridge strain and acceleration datasets, sequence-level aggregation substantially affects both classification accuracy and inter-fold stability. Further validation using quasi-static responses, environmental variables, other structural systems, and unseen operating conditions is required before broader applicability across the SHM domain can be established.

Keywords: bridge monitoring; multivariate time-series classification; structural response classification; sequence-level aggregation; xLSTM



Academic Editors: Lijun Liu, Yang Yang and Ning Yang

Received: 14 June 2026

Revised: 18 July 2026

Accepted: 19 July 2026

Published: 22 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Structural Health Monitoring (SHM) plays a pivotal role in the detection, localization, and characterization of structural degradation through the continuous or periodic analysis of sensor measurements, thereby facilitating condition-based maintenance and mitigating the risk of catastrophic failures [1,2]. Vibration-based sensing methodologies incorporating accelerometers, strain gauges, and Fiber Bragg Grating (FBG) sensors have gained significant prominence owing to their economic scalability for large-scale infrastructures such as bridges, buildings, and wind turbines [3,4]. These sensing systems generate complex multivariate time-series data in which each channel captures distinct physical quantities at specific spatial locations [5,6]. Consequently, the resulting classification task requires a synergistic analysis of both intricate temporal dynamics and interchannel correlations to ensure reliable damage detection [7,8].

In classification-oriented SHM, a multivariate observation segment represents the simultaneous structural response measured at several sensor locations. The classifier converts this segment into a condition label that can support automated condition screening and help prioritize subsequent inspection or maintenance actions. Structural changes generally affect the spatially correlated response of multiple sensors rather than an isolated measurement. Reliable classification therefore requires the model to capture both the temporal evolution within individual channels and the relationships among channels associated with the global structural response [9].

Recent advances in bridge SHM have addressed both the extraction of damage-sensitive information and the reliability of multichannel measurements. Jiang et al. [9] proposed a bridge damage identification method based on synchronous statistical moment theory and vehicle–bridge interaction responses, illustrating the value of coordinated acceleration measurements for structural-condition assessment. Xie et al. [10] developed an enhanced generative adversarial imputation network with unsupervised learning to recover randomly missing measurements when incomplete data occur across all sensors. These studies demonstrate that practical bridge monitoring requires not only effective damage-sensitive representations but also reliable multichannel data. The present study focuses specifically on the former issue by investigating sequence-level aggregation for the classification of complete synchronized structural-response windows.

Deep learning has significantly improved the accuracy of damage detection and classification when applied to these analyses [11,12]. In contrast to traditional feature-engineering approaches, which rely on handcrafted indicators such as modal frequencies, damping ratios, or wavelet coefficients, deep learning models autonomously learn hierarchical representations directly from raw or minimally preprocessed signals [13,14]. This paradigm shift has motivated the adaptation of neural architectures originally designed for natural language processing and computer vision to the time-series domain, most notably transformer-based models that leverage self-attention mechanisms [15,16].

However, a strong benchmark performance in forecasting tasks does not automatically transfer to classification settings [6,17]. In classification, the model must produce a single label for an entire input sequence, which requires an effective mechanism for aggregating temporal representations into fixed-length vectors. Many Transformer-based models have been primarily optimized for sequence-to-sequence forecasting. Consequently, their readout strategies may be suboptimal when discriminative features are distributed across the full temporal extent of the signal rather than concentrated near the sequence endpoints. This observation motivates a closer examination of the readout mechanism itself, the vital component that bridges the backbone’s sequential output and the classification head.

In SHM measurements, condition-sensitive information is rarely confined to a single time instant [18,19]. Structural changes may appear as transient impulses, variations in

vibration amplitude, altered strain distributions, or gradual changes in the measured response [20]. The temporal locations of these features may vary with the excitation event, sensor position, structural condition, and operational or environmental variability. A classification model must therefore not only encode temporal dependencies but also retain relevant evidence distributed throughout the observation window. This requirement makes the sequence-level readout directly relevant to SHM classification. The temporal backbone produces a hidden representation at each time step, whereas the readout determines which of these representations contribute to the final condition-level decision. Selecting only the last hidden state assumes that relevant information from all earlier time steps has been successfully propagated to the sequence endpoint. For localized, intermittent, or noisy condition-sensitive features, this assumption may cause useful information to be attenuated or discarded [21,22]. In contrast, aggregating hidden states across the observation window provides a direct mechanism through which evidence from multiple temporal regions can contribute to classification.

Accordingly, sequence-level aggregation should be examined as an explicit architectural choice in SHM classification rather than treated as a minor implementation detail [23,24]. Although considerable attention has been devoted to increasingly sophisticated temporal backbones, less emphasis has been placed on how their sequential outputs are transformed into fixed-length condition representations. This distinction is important because an effective encoder may still yield suboptimal classification performance if its readout discards temporally distributed condition-sensitive information. Mean pooling provides a simple and computationally efficient mechanism through which all valid hidden states contribute to the sequence-level representation. Nevertheless, its effect on modern recurrent architectures for SHM-oriented multivariate classification has not been systematically isolated under a controlled backbone and training configuration [25].

Mean pooling itself is not a new sequence-modeling operation. Previous recurrent sequence encoders have employed mean, max, mean-max, or attention-based pooling to construct fixed-length representations from temporally distributed hidden states [26]. For example, pooling has been applied to BiLSTM outputs in sentence-representation models, while self-attentive recurrent encoders assign learned importance to different sequence positions. These studies establish pooling as a general representation mechanism rather than a domain-specific innovation. Accordingly, the methodological contribution of the present study is not the introduction of mean pooling. Instead, it lies in isolating sequence-level readout as the only architectural variable within a hybrid xLSTM classifier. The baseline xLSTM and xLSTM-m share the same sLSTM-mLSTM encoder, input representation, model dimensions, optimization settings, and evaluation folds. This controlled design differs from architectures in which pooling is introduced together with changes to the encoder, attention mechanism, or feature-extraction pipeline, because it allows the observed performance difference to be attributed specifically to the readout strategy.

The engineering relevance of this modification arises from the temporal characteristics of structural-response signals. Damage- or condition-sensitive information may appear as short-duration impulses, amplitude changes, altered strain distributions, or gradual response variations [27]. These features may occur near the beginning, middle, or end of an observation window, depending on the excitation timing, sensor location, structural condition, and measurement noise. A last-hidden-state readout requires such information to remain encoded until the final recurrent state, creating a single-state information bottleneck and making the classification representation sensitive to the sequence endpoint. Masked mean pooling instead allows each valid hidden state to contribute directly to the final representation. It can therefore preserve evidence distributed across the observation window and reduce sensitivity to a single noisy or unrepresentative terminal state. This

mechanism provides a plausible explanation for the observed improvements in classification accuracy and inter-fold stability, although the present results should be interpreted as empirical evidence rather than a complete causal analysis.

Based on this motivation, this study introduces xLSTM-m as a controlled readout variant of the extended Long Short-Term Memory architecture [28]. The conventional last-hidden-state representation is replaced by masked mean pooling over all valid hidden states, while the hybrid sLSTM–mLSTM backbone, input representation, model dimensions, optimization settings, and cross-validation folds remain unchanged. The purpose of this modification is not to propose a new pooling operation, but to determine whether whole-sequence aggregation provides a more appropriate condition-level representation for multivariate structural-response signals. Within the paired comparison between xLSTM and xLSTM-m, the readout strategy is therefore the only modified architectural component.

Three core hypotheses are posited in this study:

- (1) The masked mean pooling strategy yields a consistently higher classification accuracy and superior inter-fold stability compared with the conventional last-hidden-state readout when applied to the xLSTM architecture.
- (2) The hybrid sLSTM–mLSTM backbone achieved competitive or superior classification performance relative to state-of-the-art transformer models when evaluated under a unified benchmarking protocol for SHM.
- (3) The accuracy gain afforded by xLSTM-m was sufficiently large to justify the additional computational overhead incurred, thereby confirming its viability for real-world SHM deployment.

The primary contributions of this study are as follows: First, we establish a unified benchmark of eight neural architectures using three bridge-response datasets comprising multichannel strain and acceleration measurements, with consistent preprocessing, training, and evaluation protocols. Second, we introduce a rigorous K-fold evaluation methodology that integrates epoch-wise training dynamics with pooled out-of-fold test predictions, thereby offering a reproducible template for standardized SHM classification reporting. Third, we provide extensive empirical evidence demonstrating the superior classification performance and inter-fold stability of the proposed xLSTM-m model compared with prominent transformer-based baselines across all evaluation metrics. Finally, through a controlled comparison in which the temporal backbone and training configuration remain unchanged, we demonstrate that the sequence-level readout can substantially affect classification accuracy and inter-fold stability across the evaluated bridge-response datasets.

The remainder of this paper is organized as follows. Section 2 reviews related literature on deep learning applications in SHM, Transformer-based time-series models, and existing sequence-level readout strategies. Section 3 details the proposed methodology, specifically focusing on the xLSTM-m architecture and the selected transformer baselines. The experimental setup, comprising three benchmark datasets, training protocols, and evaluation metrics, is described in Section 4. Section 5 presents a comprehensive analysis and discussion of the results, and Sections 6 and 7 address the study's limitations and provide concluding remarks.

2. Related Work

2.1. Deep Learning for SHM

The integration of deep learning with SHM has progressed rapidly over the past decade. Early research primarily applied convolutional neural networks (CNNs) to spectrograms or wavelet transforms of vibration signals [29]. Recent methodologies have shifted toward processing raw time-series data directly using 1D-CNNs, recurrent networks, or hybrid architectures [30]. For instance, Quyet et al. [31] proposed a 1D-CNN architecture

featuring channel attention mechanisms for steel-frame damage classification, achieving a 93.6% test accuracy on the Qatar University Grandstand Simulator dataset. Similarly, hybrid CNN-LSTM models have demonstrated robust performance by synergizing spatial feature extraction with temporal dependency modeling [32]. In the context of bridge monitoring, Le et al. [33] developed a ResUNet-based model for damage detection utilizing time-series data derived from large-scale numerical simulations. Recent advancements have further extended the application of deep learning to sophisticated damage classification tasks in the domain of structural mechanics.

Recent research has expanded beyond conventional convolutional and recurrent architectures [34,35]. Message-passing and graph neural networks have been used to represent the spatial relationships among sensors and structural components, particularly for cable-damage localization and infrastructure performance prediction [36]. Transformer-based models have also been adopted to model long-range temporal dependencies in bridge strain, vibration, and environmental monitoring data [37,38]. In parallel, transfer learning, generative models, unsupervised learning, and physics-guided architectures have been investigated to address the scarcity of labeled damage data and improve cross-structure transferability [5,39].

FBG sensor networks have gained significant traction in SHM applications owing to their exceptional sensitivity, immunity to electromagnetic interference, and robust multiplexing capabilities [7]. The Z24 bridge dataset [40] has also gained similar traction, emerging as a canonical benchmark for damage detection, demonstrating the critical value of controlled, progressive damage experiments for validating novel methodologies. These established studies collectively underscore the necessity of aligning the neural architecture design with the unique characteristics of SHM sensor data, a principle central to this research. Instead of merely increasing the backbone complexity, this study suggests that a more effective temporal representation of existing hidden states through optimized aggregation can be a primary driver of performance gains in SHM.

Despite these advances, most existing studies have concentrated on improving the feature-extraction or temporal-encoding backbone, including CNN modules, recurrent units, graph operations, attention mechanisms, and hybrid physics–data architectures [41]. Comparatively less attention has been given to the sequence-level readout that converts a series of latent temporal states into the fixed-length representation required for classification. This issue is particularly relevant for multivariate structural-response signals, in which condition-sensitive information may be transient, intermittent, or distributed across multiple temporal regions and sensing channels. The present study therefore isolates sequence-level aggregation as a controlled architectural variable by comparing two xLSTM configurations with an identical hybrid sLSTM–mLSTM backbone and training protocol, differing only in their readout strategy.

2.2. Transformer-Based Models for Time-Series Analysis

Time-series analysis concerns the extraction and modeling of information from observations recorded in chronological order. Depending on the application, common tasks include forecasting future responses, classifying complete observation sequences, detecting anomalies, imputing missing values, and learning transferable temporal representations. In multivariate time series, the model must capture not only the temporal evolution within each variable but also the dependencies among variables measured at different sensing locations or representing different physical quantities. These requirements are particularly relevant to structural monitoring, where the measured response reflects interacting effects of structural condition, excitation, sensor position, and environmental or operational variability. Recent advances in artificial intelligence have broadened the methodologi-

cal landscape of time-series analysis beyond conventional convolutional and recurrent networks [42,43]. Emerging directions include self-supervised representation learning, diffusion-based probabilistic modeling, graph-based spatial-temporal learning, selective state-space models, and pretrained time-series foundation models. For example, diffusion-based models have been developed for probabilistic multivariate forecasting, whereas foundation models such as MOMENT and decoder-only pretrained time-series models aim to transfer learned representations across datasets and downstream tasks [44]. Many of these developments adopt Transformer-based tokenization, masking, or attention mechanisms, while state-space architectures have emerged as an alternative for modeling long sequences with lower sequence-length complexity.

The original transformer architecture [15] has catalyzed a vast ecosystem of specialized models for time-series analysis, many of which are systematically benchmarked via the Time-Series Library (TSLib) [45]. Within this landscape, TimesNet [46] reformulated time series as two-dimensional tensors to effectively capture multiperiodicity through inception-based blocks. To mitigate the quadratic complexity of the standard self-attention in long sequences, the Informer [47] employs a ProbSparse attention mechanism. Similarly, Autoformer [48] integrated series decomposition with an autocorrelation mechanism, whereas FEDformer [49] utilizes frequency-domain decomposition to enhance temporal representations. More recent innovations include PatchTST [50], which segments time series into patches within a channel-independent framework, and iTransformer [51], which inverts the attention dimension to more effectively model cross-variate dependencies. These architectures demonstrate several advantages for multivariate time-series analysis. First, attention mechanisms can assign data-dependent importance to different temporal regions or variables rather than relying on a fixed receptive field. Second, their global interactions facilitate the modeling of long-range temporal and interchannel dependencies. Third, their modular encoder structures can be combined with task-specific heads for forecasting, classification, anomaly detection, and imputation. Finally, patching, decomposition, frequency-domain modeling, and inverted tokenization provide alternative mechanisms for adapting Transformer representations to the temporal and multivariate characteristics of different datasets.

Despite achieving competitive results in forecasting benchmarks, comprehensive evaluations, often characterized as ‘bake-off’ studies, have revealed that model performance remains highly contingent upon dataset characteristics, normalization strategies, and specific evaluation protocols [52,53]. Notably, Foumani et al. [6] emphasized that classification necessitates a fundamentally different aggregation strategy than forecasting, as the model must distill an entire input sequence into a single, highly discriminative representation rather than predict future values. This critical distinction underscores the necessity of a systematic comparison in this research.

2.3. Recurrent Models and Sequence-Level Readout Strategies

Long Short-Term Memory (LSTM) networks [54,55] remain a foundational architecture for sequential modeling owing to their gating mechanisms that mitigate vanishing gradient problems. The recently introduced xLSTM [28] modernizes this family by incorporating exponential gating and revised memory structures through two complementary blocks, scalar memory (sLSTM) and matrix memory (mLSTM), resulting in enhanced capacity and improved trainability. In the time-series domain, xLSTMTime [56] has already demonstrated competitive performance against transformer baselines for long-term forecasting. However, the potential of xLSTM architectures for classification tasks remains underexplored, where a critical yet often overlooked design choice is the readout strategy.

The default readout approach, which selects the last valid hidden state, assumes that all discriminative information is effectively propagated to the final time step. Although this assumption holds for many natural language tasks, it may be suboptimal for SHM sensor signals, where damage-distinguishing patterns can appear at arbitrary temporal positions. Alternatively, mean pooling over all valid hidden states treats every time step equally and reduces the representation variance; however, its specific impact on recurrent architectures within the SHM domain has not been systematically investigated. This study addresses this gap by isolating the readout mechanism as a controlled variable and comparing two xLSTM configurations that differ exclusively in their aggregation strategy.

3. Methodology

3.1. Problem Formulation

This study considers a supervised classification problem in which each sample comprises synchronized measurements collected from multiple sensor channels over a fixed observation window. The objective is to assign one class label to the complete multivariate sequence, where the label represents a predefined loading condition, structural state, or damage-related scenario. The following formulation defines the input and output spaces of this classification problem and provides the basis for examining how temporal representations are aggregated before classification. Given a multivariate time-series input,

$$X = [x_1, x_2, \dots, x_T], \quad x_t \in \mathbb{R}^C \quad (1)$$

Equation (1) represents one multivariate observation segment. Where T denotes the number of time steps, C denotes the number of sensor channels, and the classification task is to learn a mapping. Each row corresponds to the simultaneous measurements obtained from all sensor channels at one time step, whereas each column describes the temporal response of an individual channel over the observation window. Based on this input representation, the classification objective is to learn a mapping from the complete multivariate segment to one of K predefined classes, as expressed in Equation (2):

$$f : \mathbb{R}^{T \times C} \rightarrow \{1, \dots, K\} \quad (2)$$

where K denotes the number of classes. The mapping in Equation (2) converts the input sequence into a class-level prediction, rather than producing an independent output at each time step. Therefore, the sequential features generated by the temporal encoder must first be transformed into a fixed-length representation before being passed to the classification head.

3.2. Input Representation and Patch Embedding

Before being processed by the temporal encoder, each raw multivariate observation segment is converted into a sequence of non-overlapping temporal patches. Equation (3) defines the construction of the j -th patch token from the raw multivariate response segment:

$$\mathbf{s}_j = \text{vec}(\mathbf{X}_{(j-1)P+1:jP,:}) \in \mathbb{R}^{PC}, \quad j = 1, \dots, L, \quad L = \frac{T}{P} \quad (3)$$

In Equation (3), $\mathbf{X} \in \mathbb{R}^{T \times C}$ denotes the raw multivariate observation segment, T is the number of time steps, C is the number of sensor channels, and P is the number of consecutive time steps included in each non-overlapping patch. The vector $\mathbf{s}_j \in \mathbb{R}^{PC}$ denotes the j -th flattened patch token, $\text{vec}(\cdot)$ denotes the vectorization operation, j is the patch index, and $L = \frac{T}{P}$ is the resulting number of patch tokens. Stacking the L patch

tokens produces the patch sequence $S \in \mathbb{R}^{L \times PC}$. In the present study, $T = 2000$ and $P = 10$, resulting in $L = 200$ patch tokens.

This patch embedding serves three purposes: (i) it reduces the effective sequence length from 2000 to 200, improving training stability and reducing memory requirements; (ii) each token encodes a richer local context compared to individual sample points; and (iii) it provides a uniform input format across all eight models, ensuring that performance differences reflect architectural choices rather than input representation discrepancies. Normalization statistics were computed exclusively on the training split of each fold and applied consistently to the validation and test splits, preventing information leakage across data partitions.

3.3. xLSTM-m Architecture Overview

The proposed xLSTM-m architecture is designed to transform a patch-embedded multivariate input sequence into a fixed-length representation for structural-condition classification. The model comprises three principal processing stages. First, each input token is projected into a latent embedding space. Second, the resulting sequence is processed by a hybrid xLSTM encoder composed of sLSTM and mLSTM blocks. Third, the sequence of encoded hidden states is aggregated and passed to a linear classification head. The proposed model and the baseline xLSTM share the same input-projection and hybrid-encoder stages and differ only in the sequence-level readout used in the final stage. The complete processing pipeline and the distinction between the two readout strategies are illustrated in Figure 1.

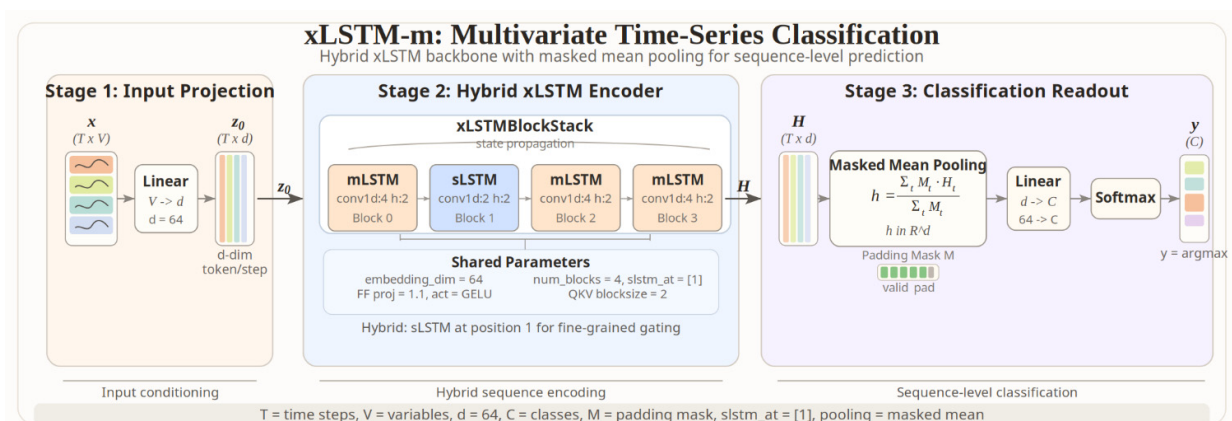


Figure 1. Architecture of the proposed xLSTM-m for multivariate time-series classification.

As shown in Figure 1, the patch-embedded input is first mapped to the model embedding dimension and subsequently propagated through the hybrid xLSTMBlockStack. The encoder produces one hidden representation for each input token. In the baseline xLSTM, the final valid hidden state is used as the sequence-level representation, whereas xLSTM-m averages all valid hidden states through masked mean pooling. The following subsections describe the three processing stages in detail.

Stage 1 (Input projection): A linear layer maps each patch-embedded token from the input dimension $V = P \cdot C$ to the hidden dimension $d = 64$, producing the initial embedding sequence $z_0 \in \mathbb{R}^{L \times d}$.

Stage 2 (Hybrid xLSTM encoder): The hybrid xLSTM encoder processes the projected patch sequence and generates one hidden representation for each input token. Equation (4) defines the sequence of hidden states produced by the hybrid xLSTM encoder:

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_L]^T \in \mathbb{R}^{L \times d} \quad (4)$$

In Equation (4), $\mathbf{H}$ denotes the complete output sequence of the hybrid xLSTM encoder, $\mathbf{h}_i \in \mathbb{R}^d$ is the hidden representation associated with the i -th patch token, $i \in \{1, 2, \dots, L\}$ is the token index, L is the number of patch tokens, d is the hidden or embedding dimension, and the superscript T denotes the transpose operation. In the present architecture, $d = 64$.

The sLSTM block extends the classical LSTM cell with exponential gating and a revised memory structure. Unlike the standard LSTM sigmoid-bounded gates, sLSTM introduces an exponential input gate that dynamically amplifies or suppresses updates to the cell state, providing greater flexibility in controlling information flow while preserving sequential processing semantics. Each sLSTM block used a 1D convolution with a kernel size of four and two heads (conv1d:4 h:2).

The mLSTM block replaces the scalar memory cell with a matrix-valued memory, enabling each block to store richer associative patterns analogous to the key-value memory in attention mechanisms, but within a recurrent framework. Matrix memory captures more complex temporal dependencies, particularly for inputs exhibiting multiscale dynamics. Each mLSTM block also uses a 1D convolution with kernel sizes of four and two heads, with a QKV block size of two.

The baseline xLSTM and the proposed xLSTM-m share the same backbone architecture and parametric configuration. Their only architectural difference is the sequence-level readout strategy: the baseline xLSTM uses the last valid hidden state, whereas xLSTM-m applies masked mean pooling over all valid hidden states. The shared architectural settings are summarized in Table 1.

Table 1. Architectural hyperparameters and structural specifications for the xLSTM family.

Parameter	Value
Embedding dimension (d)	64
Number of blocks	4
Block composition	mLSTM–sLSTM–mLSTM–mLSTM
sLSTM position (slstm_at)	[1]
Conv1d kernel size	4
Number of heads	2
QKV blocksize	2
Feed-forward projection factor	1.1
Activation function	GELU
Pooling strategy (xLSTM-m)	Masked mean
Pooling strategy (xLSTM baseline)	Last hidden state

Stage 3 (Classification readout): The baseline xLSTM converts the encoder output sequence into a fixed-length representation by selecting the final valid hidden state. Equation (5) defines the last-hidden-state readout used by the baseline xLSTM:

$$\mathbf{z}_{\text{last}} = \mathbf{h}_{L_{\text{valid}}} \quad (5)$$

In Equation (5), $\mathbf{z}_{\text{last}} \in \mathbb{R}^d$ denotes the sequence-level representation used for classification, $\mathbf{h}_{L_{\text{valid}}}$ is the hidden state corresponding to the final valid token, and L_{valid} denotes the index of the last non-padded token in the sequence. Because all sequences in the present benchmark have the same length and contain no padded positions, $L_{\text{valid}} = L$.

The selected sequence-level representation is subsequently mapped to the class-logit space by the linear classification head. Equation (6) defines the class-logit vector obtained from the baseline xLSTM readout:

$$\ell_{\text{last}} = \mathbf{W}_c \mathbf{z}_{\text{last}} + \mathbf{b}_c \in \mathbb{R}^K \quad (6)$$

In Equation (6), $\ell_{\text{last}} \in R^K$ denotes the vector of class logits, $\mathbf{W}_c \in R^{K \times d}$ and $\mathbf{b}_c \in R^K$ are the weight matrix and bias vector of the linear classification head, respectively, K is the number of predefined classes, and d is the hidden dimension. The class probabilities are obtained by applying the softmax function to ℓ_{last} and the predicted class corresponds to the class with the highest probability.

This readout strategy relies on the assumption that all discriminative information has been propagated to the final time step, which may not hold when class-distinguishing features are distributed across the temporal extent of the signal.

3.4. Proposed Modification: Mean Pooling Readout

The proposed xLSTM-m retains the same input projection, hybrid xLSTM encoder, and classification head as the baseline xLSTM, but replaces the last-hidden-state readout with masked mean pooling. Equation (7) defines the masked mean pooling operation applied to the encoder hidden states:

$$\mathbf{z}_{\text{mean}} = \frac{\sum_{i=1}^L m_i \mathbf{h}_i}{\sum_{i=1}^L m_i}. \quad (7)$$

In Equation (7), $\mathbf{z}_{\text{mean}} \in R^d$ denotes the pooled sequence-level representation, $\mathbf{h}_i \in R^d$ is the hidden representation of the i -th token, L is the total number of token positions, and $m_i \in \{0, 1\}$ is the binary validity-mask indicator for the i -th token. Specifically, $m_i = 1$ for a valid token and $m_i = 0$ for a padded token.

Because all sequences used in the present benchmark have an identical length and contain no padded tokens, $m_i = 1$ for all i . Equation (8) presents the resulting simplified mean-pooling operation:

$$\mathbf{z}_{\text{mean}} = \frac{1}{L} \sum_{i=1}^L \mathbf{h}_i \quad (8)$$

In Equation (8), $\mathbf{z}_{\text{mean}} \in R^d$ is the mean-pooled sequence representation, L is the number of valid patch tokens, and $\mathbf{h}_i \in R^d$ is the hidden representation associated with the i -th token. The same linear classification head defined in Equation (6) is then applied to $\mathbf{z}_{\text{mean}}$, followed by the softmax function to obtain the class probabilities.

The classification head then computes $\hat{y} = \mathbf{W}_c \mathbf{z}_{\text{mean}} + \mathbf{b}_c$, followed by softmax to produce class probabilities, with the final prediction obtained via argmax.

The novelty of xLSTM-m lies not in the pooling operation itself, which is well established in NLP and speech processing but in the controlled demonstration that this single change, applied to a modern recurrent architecture for SHM classification, produces substantial and consistent improvements. As illustrated in Figure 1, all parameters in Table 1 remain identical between xLSTM and xLSTM-m; only the pooling strategy differs, ensuring that any performance difference is solely attributable to the aggregation mechanism.

3.5. Transformer Baselines

The benchmark includes six transformer-oriented models (shown in Table 2) from TSLib [45], which were selected to represent diverse architectural philosophies.

TimesNet transforms a one-dimensional time series into a set of two-dimensional temporal representations based on its dominant periods. Inception-style convolutional blocks are then applied to capture both intraperiod and interperiod variations. Although TimesNet does not employ conventional self-attention, it was included as a contemporary

time-series baseline because its period-based representation has demonstrated competitive performance across forecasting, classification, imputation, and anomaly detection tasks.

Table 2. Summary of the contemporary time-series baseline architectures evaluated in the benchmark.

Model	Key Mechanism	Original Task Focus
TimesNet [46]	2D temporal variation modeling via inception blocks	General time-series analysis
Informer [47]	ProbSparse attention for long sequences	Long-term forecasting
Autoformer [48]	Series decomposition + Auto-Correlation	Long-term forecasting
FEDformer [49]	Frequency-enhanced decomposition	Long-term forecasting
PatchTST [50]	Patch embedding + channel-independent Transformer	Long-term forecasting
iTransformer [51]	Inverted attention over variate axis	Multivariate forecasting

Informer, Autoformer, and FEDformer were developed primarily to improve the efficiency and representation capability of Transformer-based long-sequence forecasting. Informer employs ProbSparse self-attention, which emphasizes the most informative query–key interactions and reduces the computational cost of standard attention for long sequences. Autoformer progressively separates the temporal signal into trend and seasonal components and replaces conventional attention with an autocorrelation mechanism designed to capture periodic dependencies at different time delays. FEDformer also incorporates series decomposition but performs temporal representation learning in the frequency domain, using selected Fourier or wavelet components to capture dominant global patterns while suppressing less informative fluctuations.

PatchTST divides each sensor channel into nonoverlapping or partially overlapping temporal patches and treats these patches as input tokens to a shared Transformer encoder. Its channel-independent design processes each variable separately while sharing model parameters across channels, thereby reducing the token sequence length and preserving local temporal context. In contrast, iTransformer inverts the conventional tokenization strategy by representing each variable as a token. Self-attention is therefore applied across variables to learn interchannel relationships, while the temporal observations of each variable are encoded within its token representation. This design is particularly relevant to multivariate sensor data in which structural-state information may be reflected through correlated responses across sensing locations.

In the present benchmark, these architectures were implemented using the TSLib classification framework. Their task-specific forecasting outputs were replaced by the corresponding classification interface so that each model produced one class prediction for the complete input segment. The same preprocessed input segments, fold assignments, checkpoint-selection criterion, and evaluation metrics were used across the comparative models.

4. Experimental Setup

4.1. Datasets

The proposed models were evaluated using three bridge-monitoring datasets: two laboratory-scale datasets comprising FBG strain responses and PCB acceleration responses acquired from a cable-stayed bridge model, and one field-scale acceleration dataset derived from the Z24 bridge benchmark in Switzerland. This selection enables evaluation across two sensing technologies and both laboratory- and field-scale bridge conditions. The primary characteristics of these datasets following preprocessing and segmentation are summarized in Table 3.

Table 3. Statistical summary and structural characteristics of the SHM benchmarks following data preprocessing.

Dataset	Classes	Segments	Channels (C)	Sampling Rate	Seq. Length	Source
FBG	8	1000	14	800 Hz	2000	FBG sensor array on cable-stayed bridge model
PCB	8	3036	8	2000 Hz	2000	PCB accelerometers on the same bridge model
Z24-9Setup	17	4896		100 Hz	2000	Aggregated from 9 measurement setups on the Z24 bridge

Specifically, laboratory datasets offer controlled conditions to validate the classification performance for diverse strain and vibration signals. In contrast, the Z24 benchmark serves as a rigorous test for model generalizability, featuring a field-scale structure characterized by progressive damage scenarios and environmental variability. This dual-scale design provides evidence of performance across the evaluated laboratory and field bridge datasets, while broader robustness across other sensing modalities and monitoring contexts remains to be established.

Each sample was represented as a multivariate matrix with 2000 time steps and C synchronized sensor channels. The FBG dataset contained 14 strain-response channels, including eight deck-mounted and six cable-mounted sensors, whereas the PCB dataset contained eight deck acceleration channels. For the Z24-9Setup dataset, five dynamic-response channels common to all nine measurement setups were retained. Each complete segment was assigned a single categorical label representing one of the predefined loading, structural-condition, or damage-related states described in Tables 4 and 5.

Table 4. Characterization of the eight operational loading and structural damage scenarios for the FBG and PCB benchmarks.

Case	Deck Loading
1	Uniform: 10 kg at all 6 positions
2	Asymmetric: 20 kg at position 1, 10 kg elsewhere
3	Concentrated: 30 kg at position 2, 10 kg elsewhere
4	Alternating: 20 kg at positions 1, 3; 10 kg elsewhere
5	Asymmetric: 20 kg at positions 1, 3; 10 kg elsewhere
6	Symmetric: 20 kg at positions 2, 6; 10 kg elsewhere
7	Alternating: 20 kg at positions 1, 3, 5; 10 kg elsewhere
8	Central-symmetric: 20 kg at positions 2, 5; 10 kg elsewhere

Table 5. Characterization of the seventeen distinct structural configurations and damage scenarios in the Z24-9Setup benchmark.

State	Category	Damage Scenario
1–2	Reference	Baseline measurements (before and after pier installation)
3	Reference	Intermediate reference measurement
4–6	Settlement	Pier settlement at 40 mm, 80 mm, and 95 mm
7	Tilt	Foundation tilt (6 mm relative difference at Koppigen pier)
8	Reference	Reference measurement after settlement/tilt sequence
9–10	Spalling	Concrete spalling at 12 m ² and 24 m ²
11	Landslide	Simulated landslide at the abutment
12	Hinge failure	Failure of concrete hinges at abutment piers
13–14	Anchor failure	Failure of 1 and 4 anchor heads in post-tensioning system
15–17	Tendon rupture	Progressive tendon rupture (2, 4, and 6 tendons cut)

4.1.1. Experimental Setup and Physical Model

The FBG and PCB datasets were acquired concurrently from a laboratory-scale cable-stayed bridge model at the Department of Bridge and Tunnel Engineering, University of Transport and Communications (UTC), Hanoi, Vietnam. The experimental configuration of the laboratory-scale cable-stayed bridge is presented in Figures 2 and 3. Figure 2 shows the physical bridge model used for data acquisition, whereas Figure 3 provides a schematic representation of the bridge geometry, including the cable attachment and loading positions, deck and cable sensor locations, supports, and pylon reference points. The physical model has a total span of 3760 mm and is supported by a central pylon and six stay cables symmetrically arranged, with three cables on each side. The bridge deck was anchored at both extremities using pin and roller boundary conditions. The measurement positions were indexed from S1 to S8, progressing from right to left at a uniform interval of 250 mm, with the exception of S1 and S4 (which were placed at the end of the stay cables) and a 60 mm longitudinal gap at the pylon interface.



Figure 2. The cable-stayed bridge model utilized for data acquisition at the Bridge and Tunnel Department, University of Transportation and Communication (UTC), Hanoi, Vietnam.

Two types of sensors were deployed concurrently to capture complementary physical quantities.

FBG Sensor Network and Data Acquisition: The sensing system comprised 14 Fiber Bragg Grating (FBG) sensors interconnected in a serial configuration and interrogated at a sampling rate of 800 Hz. These sensors were strategically installed across the cable-stayed bridge model to ensure maximum sensitivity to structural deformation and vibrations. Specifically, eight sensors were clamped to the bridge deck, whereas the remaining six were surface-mounted on the stay cables (one sensor each on cables 1–3 and three sensors along cable 6). To facilitate the spatial identification of structural changes, wavelength-division multiplexing (WDM) was employed using 14 distinct center wavelengths ranging from 1525 to 1570 nm. Any localized deformation or vibration induces a measurable shift in

the corresponding center wavelength, allowing precise damage localization. The optical interrogation unit consisted of a superluminescent LED (SLED) source, optical circulator, and isolator. Data acquisition was performed using an I-MON 512 high-speed interrogator to monitor real-time wavelength shifts in the sensor network.

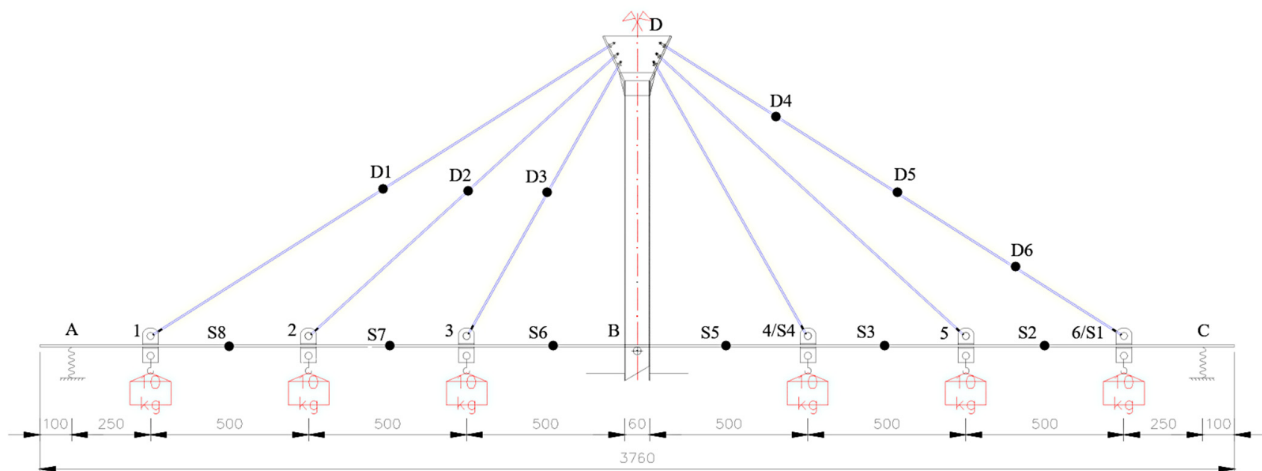


Figure 3. Schematic diagram of the cable-stayed bridge model. 1–6: cable attachment points and weight hanging place; S1–S8: FBG and PCB sensor locations on the deck; D1–D6: FBG sensors on cables; A, B, C: bridge supports; D: pylon top.

PCB Accelerometers and Data Acquisition: Vibration responses were captured using eight high-precision accelerometers from PCB Piezotronics, Inc., sampled at a rate of 2000 Hz. These sensors leverage Integrated Circuit Piezoelectric (ICP) technology, which transduces mechanical acceleration into proportional voltage signals via the piezoelectric effect. Characterized by high sensitivity, broad frequency response, and robust electromagnetic interference (EMI) immunity, these sensors are ideal for high-fidelity measurements in structural monitoring. Eight PCB 353B34 piezoelectric accelerometers were installed in a co-located configuration with deck-mounted FBG sensors. This synchronized spatial arrangement facilitates a rigorous comparative analysis between strain-based and acceleration-based damage features under identical structural excitations. The data acquisition (DAQ) architecture comprised a National Instruments cDAQ-9188 chassis equipped with two NI 9234 four-channel dynamic signal acquisition modules. This configuration provides eight simultaneous IEPE-conditioned channels, ensuring high-resolution temporal data for subsequent feature extraction.

Synchronized data acquisition was maintained across both sensor systems, ensuring that the FBG and PCB datasets captured identical physical phenomena through distinct sensing modalities: strain and acceleration, respectively. This dual-modality framework facilitates rigorous assessment of model robustness across heterogeneous signal types derived from the same structural system, providing a comprehensive basis for evaluating the generalizability of the proposed architectures.

Structural excitation was performed for each loading scenario using an impact hammer to provide consistent impulse loads at the S3 sensor location (see Figure 3) at 1 s intervals. The structural responses were recorded continuously and synchronously using both sensing systems. For preprocessing, the raw vibration signals were partitioned into nonoverlapping fixed-length windows of 2000 time steps. This segmentation process resulted in 1000 segments for the FBG dataset and 3036 segments for the PCB dataset. This discrepancy in the total number of segments is directly attributed to the higher sampling frequency of the PCB system (2000 Hz) than that of the FBG system (800 Hz) over the same recording duration.

Eight distinct loading and damage scenarios were established to investigate various structural states by systematically modulating the mass distribution and cable connectivity (Table 4). These scenarios were designed to replicate realistic structural changes ranging from operational load variations to critical component failures, thereby providing a robust dataset for evaluating the diagnostic capabilities of the proposed models.

Cases 1 through 5 represent a progressive damage sequence on the left side of the bridge with increasing load asymmetry and successive cable disconnections. Cases 6–8 introduce symmetric or central loading patterns with varying cable modifications. This design creates classification challenges that require the model to distinguish subtle differences in the strain or vibration response caused by changes in loading pattern and structural stiffness. After patch embedding with $P = 10$, both datasets had an effective sequence length of 200.

4.1.2. Z24 Bridge Benchmark Dataset

The Z24 dataset utilized data from the Z24 bridge, a post-tensioned concrete highway bridge in the canton of Bern, Switzerland. This structure was the focus of a seminal SHM experimental campaign from August to September 1998, prior to its planned demolition [13]. During this period, the bridge was subjected to a systematic sequence of progressive damage interventions, and its dynamic responses were continuously monitored. As summarized in Table 5, the experimental program included 17 discrete structural states. These scenarios transition from nondestructive reference baselines through reversible support modifications (e.g., settlement and tilt) to irreversible material degradation (e.g., spalling and hinge failure). This sequence culminates in critical failures of the prestressing system, including anchor-head loss and tendon rupture, providing a rigorous testbed for evaluating the diagnostic sensitivity of the xLSTM-m architecture.

Measurements from nine sensor setups were aggregated; only the five channels common across all setups were retained, yielding 4896 segments across 17 classes. This dataset represents the most challenging classification problem in the benchmark because of the large number of classes (17 vs. 8), heterogeneity of measurement conditions across setups, and broad variability in environmental and operational conditions during the month-long campaign.

4.2. Training Protocol and Comparison Fairness

To ensure a reproducible and procedurally consistent comparison, all eight models were evaluated using the same stratified five-fold cross-validation partitions. In each fold, one subset was held out for testing, while the remaining data were used for model training and validation. The training subset was used to estimate the normalization statistics, which were subsequently applied unchanged to the corresponding validation and test subsets. The validation subset was used exclusively for early stopping and checkpoint selection, and the held-out test subset remained unseen until the final evaluation.

All models were trained using a common experimental protocol, including identical fold assignments, raw sequence length, patch size, effective token length, learning rate, maximum number of epochs, early-stopping criterion, checkpoint-selection rule, and evaluation metrics. This design controls the principal data-processing and training conditions across architectures. The Transformer-oriented baselines were implemented using the TSLib classification framework, whereas the two xLSTM variants shared an identical hybrid sLSTM–mLSTM backbone and differed only in their sequence-level readout.

It should be noted that the present benchmark employed a predefined common training protocol rather than an exhaustive architecture-specific hyperparameter search. Therefore, the reported comparison represents the performance of the evaluated models

under consistent experimental conditions and should not be interpreted as the globally optimal performance attainable by every architecture. Architecture-specific tuning could alter the absolute performance of individual comparative models. Nevertheless, the direct comparison between xLSTM and xLSTM-m remains controlled because their architectural and optimization settings were identical, with masked mean pooling being the only modified component. Table 6 summarizes the common training, validation, and evaluation settings applied to all evaluated models and indicates the relevance of each experimental component to ensuring a fair and procedurally consistent comparison.

Table 6. Common training, validation, and evaluation settings applied to all evaluated models.

Experimental Component	Setting	Relevance to Comparison Fairness
Cross-validation	Stratified five-fold cross-validation	Maintains comparable class proportions across folds
Fold assignments	Identical fold assignments for all models	Ensures that all architectures are evaluated on the same data partitions
Maximum training epochs	500 epochs per fold	Provides the same maximum training duration for all models
Early-stopping criterion	Validation loss	Prevents the test subset from influencing the stopping decision
Checkpoint selection	Checkpoint with the lowest validation loss within each fold	Applies the same model-selection criterion across architectures
Learning rate	1×10^{-3}	Uses a common optimization step size across all evaluated models
Raw sequence length	2000 time steps	Provides an identical observation-window length
Patch size	$P = 10$	Applies the same temporal segmentation procedure
Effective sequence length	$L = 200$ tokens	Ensures that all models receive sequences of the same effective length
Normalization	Statistics computed exclusively from the training subset of each fold	Prevents information leakage from validation and test data
Validation/test transformation	Training-subset normalization statistics applied unchanged	Ensures consistent preprocessing within each fold
Transformer implementation	TSLib classification framework	Provides a common implementation framework for the six Transformer-oriented baselines
Test-prediction aggregation	Out-of-fold predictions pooled across the five test folds	Ensures that each sample contributes once to the final test evaluation
Reported test metrics	Accuracy, macro-F1-score, macro-precision, and macro-recall	Applies identical evaluation criteria to all models
Variability reporting	Mean and standard deviation across five folds for fold-level quantities	Quantifies sensitivity to data partitioning
Runtime reporting	Mean wall-clock runtime across five folds	Applies a consistent summary of computational cost
Hyperparameter optimization	Predefined common configuration; no exhaustive model-specific search	Clarifies the scope and limitation of the cross-architecture comparison
xLSTM–xLSTM-m comparison	Identical backbone and training configuration; only the readout differs	Isolates the effect of masked mean pooling

Within each cross-validation fold, each model was trained for a maximum of 500 epochs, with early stopping based on the validation loss. The checkpoint associated with the lowest validation loss was retained and subsequently evaluated on the corresponding held-out test subset. Thus, validation loss, rather than macro-averaged F1-score or test accuracy, was used as the checkpoint-selection criterion. No test-set metric was used for early stopping or checkpoint selection.

The five-fold evaluation reduces dependence on a single data partition but does not guarantee that every architecture reaches its global optimum. In particular, it does not replace repeated training with multiple random initializations or an exhaustive architecture-specific hyperparameter search. This limitation should be considered when interpreting the absolute performance differences among distinct model families.

4.3. Evaluation Metrics and Aggregation Strategy

A stratified five-fold cross-validation procedure was used to evaluate all models. Each dataset was divided into five mutually exclusive folds with approximately preserved class proportions. In each iteration, one fold was held out for testing, while the remaining samples were used for training and validation. The same fold assignments were applied to all models, and each sample appeared in the test subset exactly once across the five iterations. The test fold was not used for model training, early stopping, or checkpoint selection.

A methodologically important aspect of this study is the aggregation of 5-fold cross-validation results. Rather than selecting a single representative fold, we constructed two types of summary.

A methodologically important aspect of this study is the aggregation of the five-fold cross-validation results. Rather than selecting a single representative fold, the learning dynamics were summarized across all folds at each training epoch. Equation (9) defines the epoch-wise mean of a tracked metric across the cross-validation folds:

$$\bar{m}_e = \frac{1}{F} \sum_{f=1}^F m_e^{(f)} \quad (9)$$

In Equation (9), $\bar{m}_e$ denotes the mean value of the tracked metric at epoch e , $m_e^{(f)}$ is the value of that metric at epoch e for fold f , f is the fold index, and F is the total number of cross-validation folds. In the present study, $F = 5$. The tracked metric may represent the training loss, validation loss, training accuracy, or validation accuracy.

Equation (10) defines the corresponding inter-fold standard deviation of the tracked metric at epoch e :

$$\sigma_e = \sqrt{\frac{1}{F} \sum_{f=1}^F \left(m_e^{(f)} - \bar{m}_e \right)^2} \quad (10)$$

In Equation (10), σ_e denotes the standard deviation of the tracked metric across the F cross-validation folds at epoch e , $m_e^{(f)}$ is the metric value for fold f , and $\bar{m}_e$ is the corresponding cross-fold mean defined in Equation (9). The learning-curve plots therefore show $\bar{m}_e$ as the mean trajectory, with the shaded region representing one standard deviation above and below the mean.

In addition to the cross-validation aggregation statistics, model performance was evaluated using the loss function and four classification metrics: accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score. The same multiclass cross-

entropy loss was calculated for the training, validation, and test partitions. Equation (11) defines the cross-entropy loss used in this study:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K y_{nk} \log(p_{nk}). \quad (11)$$

In Equation (11), $\mathcal{L}_{\text{CE}}$ denotes the mean multiclass cross-entropy loss, N is the number of samples in the evaluated partition, K is the number of classes, y_{nk} is the one-hot encoded indicator that equals 1 when sample n belongs to class k and 0 otherwise, and p_{nk} is the predicted probability that sample n belongs to class k . A lower loss indicates that the predicted probability distribution is more consistent with the true class labels. During training, the loss calculated on the training subset was used for parameter optimization. The validation loss was used for early stopping and checkpoint selection, whereas the test loss was calculated only after checkpoint selection to assess the final model on the held-out test subset.

Equation (12) defines classification accuracy as the proportion of correctly classified samples among all evaluated samples:

$$\text{Accuracy} = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(\hat{y}_n = y_n) \quad (12)$$

In Equation (12), N is the total number of evaluated samples, y_n is the true class label of sample n , $\hat{y}_n$ is the predicted class label, and $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 when the predicted and true labels are identical and 0 otherwise.

Because the datasets contain multiple structural-condition classes, precision was macro-averaged across all classes. Equation (13) defines the macro-averaged precision:

$$\text{Precision}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{TP_k}{TP_k + FP_k} \quad (13)$$

In Equation (13), K is the number of classes, TP_k denotes the number of true-positive predictions for class k , and FP_k denotes the number of samples incorrectly predicted as class k . Precision therefore quantifies the proportion of predictions assigned to a class that is correct, while macro-averaging gives equal weight to each class.

Equation (14) defines the macro-averaged recall:

$$\text{Recall}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{TP_k}{TP_k + FN_k} \quad (14)$$

In Equation (14), FN_k denotes the number of samples belonging to class k that are incorrectly assigned to another class. Recall measures the proportion of samples belonging to a given class that are correctly identified, and macro-averaging assigns equal importance to all classes.

Equation (15) defines the macro-averaged F1-score as the arithmetic mean of the class-wise F1-scores:

$$\text{F1}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{2P_k R_k}{P_k + R_k} \quad (15)$$

In Equation (15), $P_k = \frac{TP_k}{TP_k + FP_k}$ and $R_k = \frac{TP_k}{TP_k + FN_k}$ denote the precision and recall of class k , respectively. The class-wise F1-score is the harmonic mean of precision and recall, and the macro-averaged F1-score assigns equal weight to each of the K classes.

For the final classification evaluation, the out-of-fold predictions from the five test folds were pooled before calculating accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score. Because each sample appeared in a held-out test fold exactly once, these pooled metrics summarize the classification performance over the complete dataset without using any sample for both training and testing within the same cross-validation iteration. Fold-wise accuracy and loss values were additionally retained to quantify inter-fold variability and to examine fitting and generalization behaviors.

5. Results and Discussion

5.1. Overall Accuracy Comparison

In this study, the classification performance and inter-fold stability of the eight evaluated models were compared across the FBG, PCB, and Z24-9Setup datasets under the same five-fold cross-validation framework. The resulting test accuracy and corresponding inter-fold variability are summarized in Table 7.

Table 7. Summary of classification accuracy and statistical variability across 5-fold cross-validation. Results are reported as the mean $\pm$ std (%) for each evaluated benchmark.

Model	FBG	PCB	Z24
xLSTM-m	97.40 $\pm$ 1.56	91.11 $\pm$ 0.56	97.37 $\pm$ 0.42
xLSTM	94.40 $\pm$ 2.13	87.65 $\pm$ 0.99	89.97 $\pm$ 0.99
TimesNet	74.20 $\pm$ 6.86	85.80 $\pm$ 1.40	65.60 $\pm$ 1.30
Informer	76.40 $\pm$ 4.75	82.28 $\pm$ 1.12	65.11 $\pm$ 0.82
Autoformer	13.10 $\pm$ 3.02	87.45 $\pm$ 1.26	31.78 $\pm$ 2.98
FEDformer	54.90 $\pm$ 13.30	84.62 $\pm$ 0.99	40.85 $\pm$ 10.76
PatchTST	91.10 $\pm$ 4.40	79.68 $\pm$ 1.45	40.13 $\pm$ 0.93
iTransformer	78.20 $\pm$ 22.01	81.06 $\pm$ 1.13	52.72 $\pm$ 0.99

The generalization behavior of each model was assessed jointly from its accuracy and loss values. Evidence of overfitting was considered present when training accuracy was substantially higher than validation or test accuracy and when this difference was accompanied by an increase in validation or test loss. In contrast, low performance across all partitions was interpreted as underfitting or optimization failure, whereas validation or test accuracy exceeding training accuracy was not treated as conventional overfitting.

To facilitate a direct visual comparison of classification performance across the three SHM datasets, Figure 4 presents the test accuracy achieved by the eight evaluated models on the FBG, PCB, and Z24-9Setup datasets. The grouped bar chart enables comparison of both the relative ranking of the models within each dataset and the variation in model performance across different sensing modalities and structural-monitoring conditions.

As shown in Figure 4, xLSTM-m achieved the highest test accuracy on all three datasets, followed by the baseline xLSTM. In contrast, the relative performance of the Transformer-oriented baselines varied considerably across datasets. For example, Autoformer achieved an accuracy of 87.45% on the PCB dataset but only 13.10% on the FBG dataset. Similarly, PatchTST achieved 91.10% accuracy on the FBG dataset but decreased to 79.68% on the PCB dataset. The largest performance difference was observed for the field-scale Z24-9Setup dataset, where xLSTM-m achieved 97.37% accuracy, compared with 65.60% for TimesNet, the strongest Transformer-oriented baseline on this dataset. These results indicate that the relative effectiveness of the comparative architectures is strongly dataset dependent,

whereas the two xLSTM variants exhibited more consistent performance across the three evaluated benchmarks.

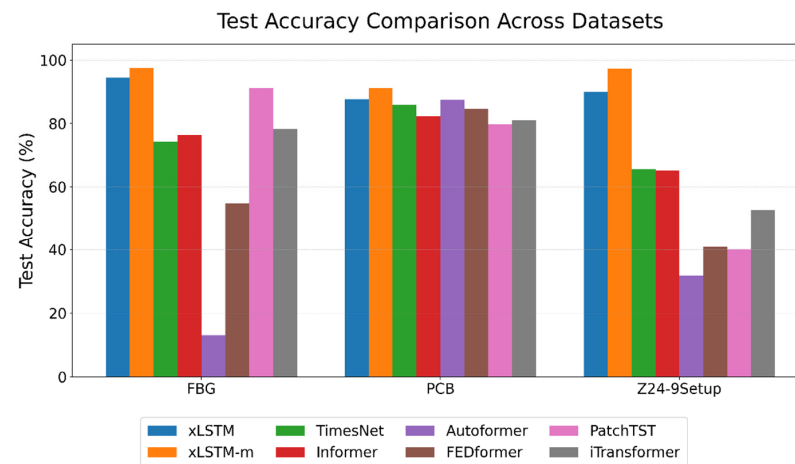


Figure 4. Comparative analysis of test accuracy across eight models and three SHM datasets.

In addition to the mean classification accuracy, the stability of each model across different data partitions was examined using the fold-wise test accuracy distributions. Figure 5 presents boxplots of the test accuracy obtained across the five cross-validation folds for each model on the FBG, PCB, and Z24-9Setup datasets. These distributions provide a visual assessment of inter-fold variability and complement the mean and standard deviation results reported in Table 7.

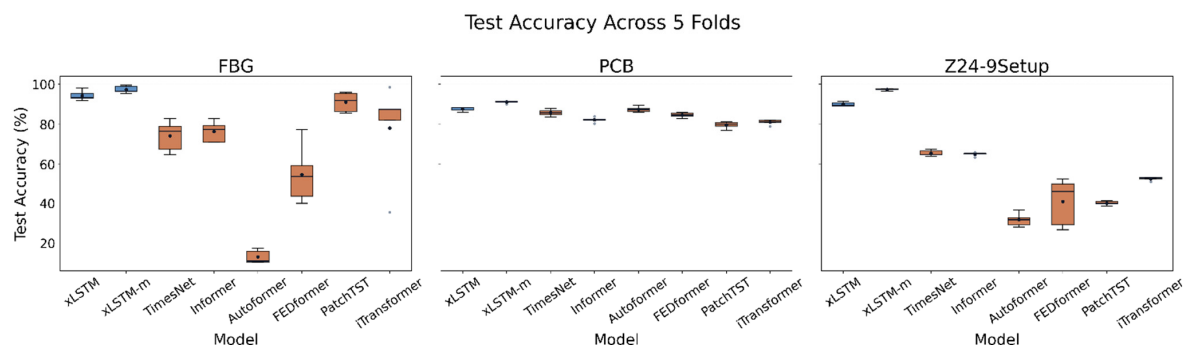


Figure 5. Statistical distribution of test accuracy across 5-fold cross-validation.

As shown in Figure 5, xLSTM-m exhibited relatively compact fold-wise accuracy distributions across all three datasets. On the FBG dataset, its median accuracy was approximately 98%, with a relatively narrow overall range, whereas iTransformer and FEDformer showed substantially greater inter-fold variability. The distributions were generally more compact on the PCB dataset, where xLSTM-m maintained the highest median accuracy and a narrow spread. On the Z24-9Setup dataset, xLSTM-m showed the most concentrated distribution near 97% accuracy, whereas the Transformer-oriented baselines achieved lower median accuracies. Overall, the boxplots support the quantitative variability measures reported in Table 7 and indicate that xLSTM-m combined high classification accuracy with comparatively low sensitivity to the cross-validation partitioning in the three evaluated datasets.

5.2. Analysis of Training Dynamics and Generalization Performance

To examine the fitting and generalization behavior of the evaluated models, Table 8 provides a detailed comparison of the training, validation, and test loss and accuracy

across the three SHM benchmarks. These results should be interpreted within the common evaluation framework adopted in this study, including identical data partitions, validation-based checkpoint selection, and a consistent test protocol across all models. Although the absolute performance of each architecture may depend on model-specific hyperparameter choices and the available computational budget, the common experimental procedure limits procedural bias and provides a consistent basis for cross-architecture comparison.

Table 8. Comparative performance and training dynamics across all benchmarks.

Dataset	Model	Train Loss	Val Loss	Test Loss	Train Acc	Val Acc	Test Acc
FBG	xLSTM-m	0.085 ± 0.034	0.089 ± 0.038	0.091 ± 0.024	97.90 ± 0.87	97.36 ± 1.11	97.40 ± 1.56
	xLSTM	0.175 ± 0.067	0.169 ± 0.048	0.169 ± 0.049	94.33 ± 2.73	94.58 ± 2.08	94.40 ± 2.13
	PatchTST	1.702 ± 1.018	1.497 ± 0.224	1.422 ± 0.757	83.08 ± 13.68	87.64 ± 5.60	91.10 ± 4.40
	iTransformer	1.129 ± 0.348	0.680 ± 0.429	0.821 ± 0.367	53.69 ± 12.56	74.44 ± 20.23	78.20 ± 22.01
	Informer	0.936 ± 0.181	0.568 ± 0.101	0.665 ± 0.184	60.03 ± 6.37	72.64 ± 7.28	76.40 ± 4.75
	TimesNet	0.871 ± 0.159	0.586 ± 0.049	0.624 ± 0.095	60.00 ± 5.31	71.67 ± 6.33	74.20 ± 6.86
	FEDformer	0.956 ± 0.748	1.367 ± 0.526	1.598 ± 0.599	62.90 ± 27.16	54.44 ± 22.74	54.90 ± 13.30
	Autoformer	2.213 ± 0.080	2.176 ± 0.036	2.374 ± 0.125	11.37 ± 1.09	14.31 ± 3.84	13.10 ± 3.02
PCB	xLSTM-m	0.230 ± 0.020	0.240 ± 0.025	0.262 ± 0.008	92.22 ± 0.85	91.76 ± 1.04	91.11 ± 0.56
	xLSTM	0.314 ± 0.050	0.337 ± 0.041	0.379 ± 0.017	89.51 ± 1.74	89.33 ± 1.49	87.64 ± 0.99
	Autoformer	0.051 ± 0.036	0.471 ± 0.058	0.591 ± 0.071	98.62 ± 1.10	86.88 ± 1.12	87.45 ± 1.26
	TimesNet	0.289 ± 0.177	0.486 ± 0.080	0.715 ± 0.123	90.87 ± 5.28	84.60 ± 3.16	85.80 ± 1.40
	FEDformer	0.205 ± 0.158	0.572 ± 0.062	0.890 ± 0.194	93.70 ± 5.24	83.43 ± 3.19	84.62 ± 0.99
	Informer	0.235 ± 0.133	0.579 ± 0.067	0.906 ± 0.147	92.42 ± 4.51	81.11 ± 2.66	82.28 ± 1.12
	iTransformer	0.288 ± 0.207	0.648 ± 0.056	0.889 ± 0.099	90.29 ± 7.61	78.32 ± 2.70	81.06 ± 1.13
	PatchTST	5.744 ± 2.222	2.883 ± 0.915	3.697 ± 0.374	38.65 ± 24.20	47.87 ± 19.50	79.68 ± 1.45
Z24-9Setup	xLSTM-m	0.003 ± 0.003	0.085 ± 0.024	0.088 ± 0.020	99.89 ± 0.08	97.56 ± 0.54	97.37 ± 0.42
	xLSTM	0.005 ± 0.007	0.465 ± 0.049	0.452 ± 0.041	99.87 ± 0.21	89.64 ± 1.03	89.97 ± 0.99
	TimesNet	0.251 ± 0.154	1.386 ± 0.051	1.582 ± 0.103	92.14 ± 5.12	61.61 ± 2.83	65.61 ± 1.30
	Informer	0.363 ± 0.156	1.402 ± 0.104	1.948 ± 0.167	88.63 ± 5.14	59.34 ± 2.83	65.12 ± 0.82
	iTransformer	0.797 ± 0.254	1.634 ± 0.028	2.286 ± 0.239	74.90 ± 8.26	48.55 ± 3.14	52.72 ± 0.99
	FEDformer	1.208 ± 0.823	2.497 ± 0.290	4.783 ± 1.121	62.52 ± 24.25	28.92 ± 8.44	40.91 ± 10.76
	PatchTST	6.523 ± 0.986	3.887 ± 0.306	8.265 ± 0.477	14.83 ± 12.25	16.21 ± 11.29	40.13 ± 0.93
	Autoformer	2.075 ± 0.733	2.645 ± 0.105	5.782 ± 1.222	36.46 ± 20.70	19.17 ± 3.27	31.77 ± 2.98

Based on the training, validation, and test metrics reported in Table 8, the fitting and generalization behavior of the evaluated models is discussed below for each dataset. On the FBG dataset, xLSTM-m exhibited a negligible training-to-test accuracy gap of 0.50 percentage points, accompanied by only a 0.006 increase in loss, from 0.085 to 0.091. The baseline xLSTM showed similarly consistent training, validation, and test performance, with accuracies of 94.33%, 94.58%, and 94.40%, respectively. These results provide little evidence of overfitting for either xLSTM variant. FEDformer exhibited a larger training-to-test accuracy decrease of 8.00 percentage points and an increase in loss from 0.956 to 1.598, together with substantial inter-fold variability, indicating weaker generalization. Autoformer remained close to chance-level performance across all partitions and therefore showed underfitting rather than overfitting. PatchTST, iTransformer, Informer, and TimesNet produced validation or test accuracies higher than their reported training accuracies. Consequently, their FBG results are more indicative of unstable optimization or partition-dependent behavior than conventional overfitting.

On the PCB dataset, xLSTM-m retained the smallest generalization gap, with training and test accuracies of 92.22% and 91.11%, respectively, and a training-to-test loss increase

of only 0.032. The baseline xLSTM also showed a limited gap of 1.87 percentage points. In contrast, Autoformer displayed the clearest overfitting pattern: its training accuracy of 98.62% decreased to 86.88% on validation and 87.45% on testing, while its loss increased from 0.051 to 0.471 and 0.591, respectively. Informer, iTransformer, and FEDformer showed training-to-test accuracy reductions of 10.14, 9.23, and 9.08 percentage points, respectively, accompanied by substantial increases in test loss. TimesNet exhibited a smaller but still observable gap of 5.07 percentage points. PatchTST did not exhibit a conventional overfitting pattern because its test accuracy exceeded both its training and validation accuracies. However, the large variability in its training and validation metrics indicates unstable convergence and sensitivity to the data partition.

The Z24-9Setup dataset produced the clearest separation in generalization behavior. Although xLSTM-m reached a training accuracy of 99.89%, its validation and test accuracies remained high at 97.56% and 97.37%, corresponding to a training-to-test gap of only 2.52 percentage points. Its test loss also remained low at 0.088. The baseline xLSTM showed a more noticeable gap, decreasing from 99.87% training accuracy to 89.97% test accuracy, with loss increasing from 0.005 to 0.452, indicating moderate overfitting. TimesNet, Informer, and iTransformer exhibited larger training-to-test gaps of 26.53, 23.51, and 22.18 percentage points, respectively, together with substantial increases in loss, indicating pronounced overfitting on this 17-class problem. FEDformer also showed poor generalization, with accuracy decreasing from 62.52% to 40.91% and loss increasing from 1.208 to 4.783. By contrast, PatchTST and Autoformer achieved relatively low training accuracies of 14.83% and 36.46%, respectively. Their poor validation and test results therefore reflect insufficient learning or optimization failure rather than memorization of the training data.

xLSTM-m demonstrated the most consistent balance between fitting capacity and generalization across the three datasets. Nevertheless, the results also show that the behavior of the comparative models is dataset-dependent: some exhibit classical overfitting, whereas others fail to fit the training data adequately or show substantial inter-fold instability. Accordingly, performance differences should be interpreted using accuracy, loss, and fold variability jointly rather than from a single train–test accuracy gap.

The joint analysis of loss and accuracy reveals distinct fitting and generalization behaviors among the evaluated architectures. Among the two xLSTM variants, xLSTM-m exhibited the most consistent training-to-test behavior across the three datasets. Its training-to-test accuracy gaps were 0.50, 1.11, and 2.52 percentage points for the FBG, PCB, and Z24-9Setup datasets, respectively. The corresponding increases in loss were 0.006, 0.032, and 0.085. In contrast, the baseline xLSTM showed a substantially larger generalization gap on Z24-9Setup, where its accuracy decreased from 99.87% during training to 89.97% during testing and its loss increased from 0.005 to 0.452. These results indicate that the masked mean pooling variant provided more consistent generalization than the last-hidden-state baseline across the three evaluated datasets.

Several comparative models exhibited markedly different behaviors depending on the dataset. On Z24-9Setup, Autoformer and PatchTST achieved relatively low training accuracies of 36.46% and 14.83%, respectively, indicating insufficient fitting or optimization difficulty rather than conventional overfitting. Cross-dataset variability was also evident. For example, Autoformer achieved a test accuracy of 87.45% with a test loss of 0.591 on the PCB dataset but only 13.10% accuracy with a test loss of 2.374 on the FBG dataset. This substantial variation indicates that the effectiveness of these architectures is strongly dependent on the characteristics of the input data and the evaluated task. However, the present results do not isolate the specific architectural mechanisms responsible for these differences. For xLSTM-m, the generally small training-to-validation and training-to-test gaps, together with the corresponding loss values, indicate limited evidence of

severe overfitting across the three datasets. Overall, the results suggest that xLSTM-m achieved a more consistent balance between fitting capacity and generalization than the comparative models under the present experimental protocol. Nevertheless, this conclusion is restricted to the evaluated datasets, preprocessing procedure, and model configurations and should not be interpreted as evidence of universal superiority over Transformer-based architectures.

5.3. Comprehensive Assessment of Training Dynamics

To bridge the gap between the aggregate performance metrics and actual model behavior, Figures 6–8 illustrate the complete training dynamics for the eight candidate models. These four-panel composites provide a synchronized view of the accuracy and loss evolution across both training and validation partitions. By visualizing the mean performance alongside the ± 1 std interval over 5-fold cross-validation, these figures reveal the intricate nuances of the learning process, offering critical insights into the convergence speed, stability, and generalization capacity of each architecture.

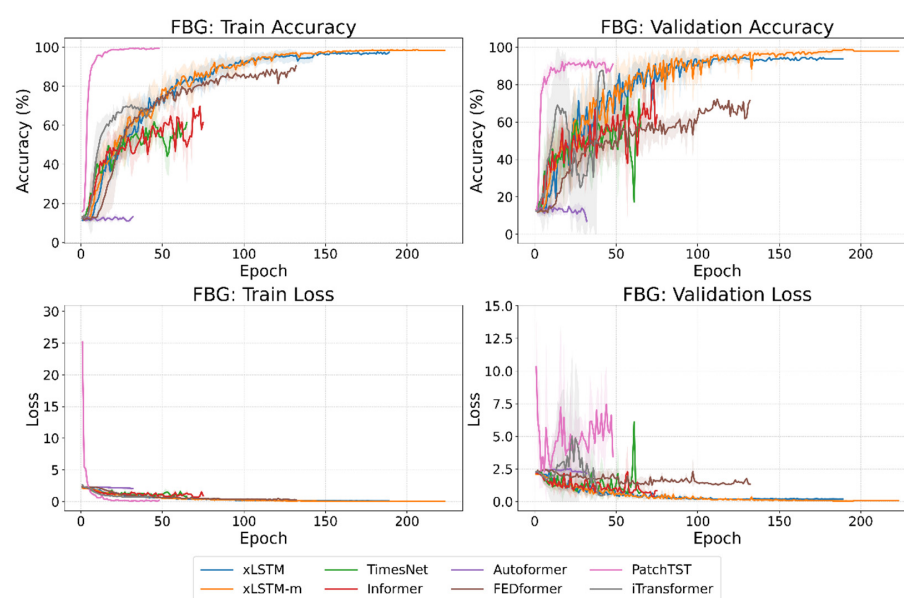


Figure 6. Training and validation dynamics of the evaluated models for the FBG dataset.

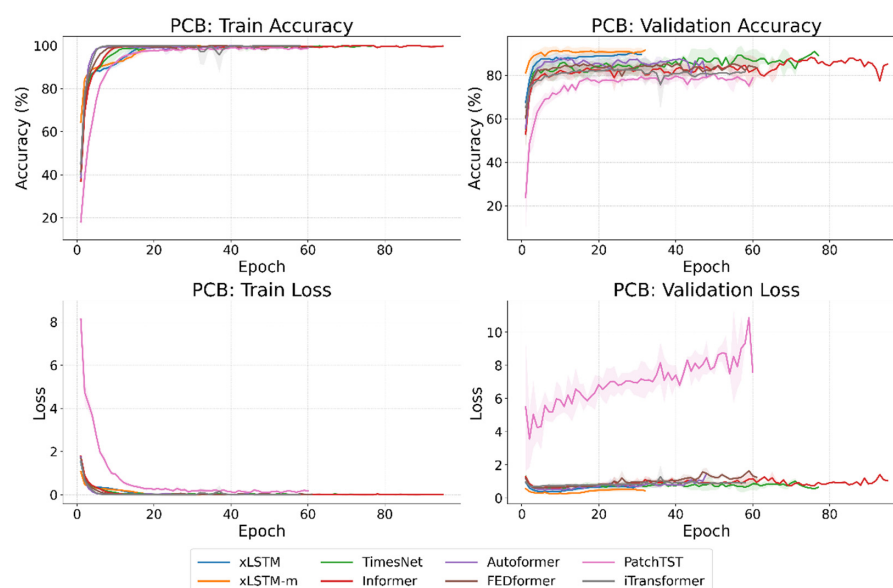


Figure 7. Evolution of training and validation dynamics for the PCB dataset.

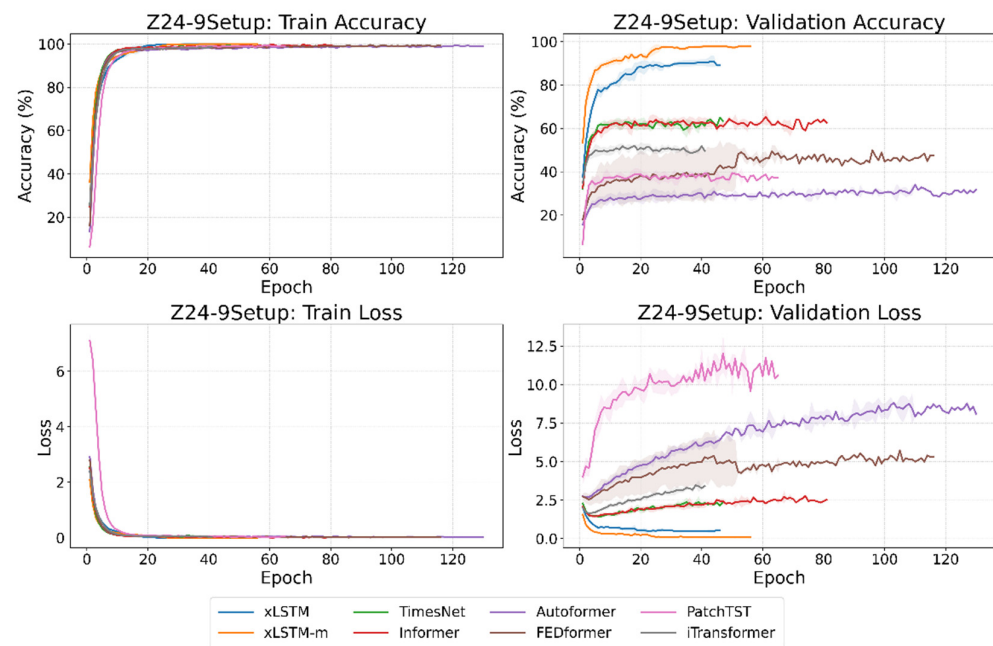


Figure 8. Training and validation dynamics for the Z24-9Setup benchmark.

5.3.1. FBG Dataset

Figure 6 presents the evolution of training and validation accuracy and loss for the FBG dataset. The solid curves represent the mean values across the five cross-validation folds, while the shaded regions indicate ± 1 standard deviation and therefore reflect inter-fold variability. Among the evaluated models, xLSTM-m exhibits the most rapid convergence, reaching near-perfect training accuracy by approximately epoch 30. It also achieves the highest validation accuracy, approximately 97%, while maintaining a narrow uncertainty band. The baseline xLSTM follows a similar but slightly slower trajectory and converges to approximately 95% validation accuracy by epoch 50.

In stark contrast, Autoformer exhibited a catastrophic failure mode, stagnating near the random baseline of 12.5% (equivalent to 1/8 classes) throughout the training duration. This suggests that its decomposition-based architecture is fundamentally ill-suited for extracting meaningful diagnostic features from FBG strain signals. While PatchTST achieves the most competitive performance among transformer baselines ($\approx 91\%$), it requires nearly 100 epochs to plateau, roughly threefold the optimization time required for xLSTM-m. Furthermore, the validation loss profiles show significant stability disparities, whereas TimesNet and iTransformer suffer from pronounced optimization instability with loss spikes peaking above 5.0 and 12.0, respectively, and both xLSTM variants maintain smooth, monotonically decreasing trajectories below 0.5. These dynamics provide definitive evidence of the superior training stability and inductive bias provided by the xLSTM framework in structural health monitoring tasks.

5.3.2. PCB Dataset

Figure 7 presents the epoch-wise training and validation accuracy and loss trajectories of the eight evaluated models for the PCB dataset, enabling comparison of their convergence behavior and generalization performance.

As shown in Figure 7, the training trajectories for the PCB dataset reveal substantial differences in convergence and validation behavior among the evaluated models. While most models demonstrated rapid convergence by exceeding 90% training accuracy within the first 20 epochs, the validation metrics revealed a clear three-tier separation in performance. xLSTM-m defines the superior tier, stabilizing at approximately 91% validation

accuracy, followed by a second tier consisting of the baseline xLSTM and autoformer at approximately 88%. The final tier includes the remaining transformer models, which cluster between 80 and 86%. Notably, Informer exhibits the slowest learning rate, taking nearly 60 epochs to plateau, suggesting limitations in the efficiency of its ProbSparse attention mechanism at this specific data scale. The most significant finding in the PCB dynamics is the catastrophic overfitting observed in PatchTST. Despite a monotonically decreasing training loss, its validation loss sharply diverges after epoch 30, increasing from ≈ 2.0 to exceeding 10.0, the most pronounced training-validation divergence recorded in the entire benchmark. Conversely, both xLSTM variants maintained a consistently low validation loss below 0.5, underscoring the effective regularization and superior inductive bias provided by the recurrent framework in mitigating overfitting for structural vibration signals.

5.3.3. Z24 Dataset

Figure 8 presents the corresponding training and validation accuracy and loss trajectories of the eight evaluated models for the Z24-9Setup benchmark, providing a visual comparison of their fitting capacity, convergence behavior, and generalization performance.

As shown in Figure 8, the Z24-9Setup benchmark reveals substantial differences among the evaluated models in both fitting capacity and generalization behavior. The Z24-9Setup benchmark revealed substantial differences in both fitting capacity and generalization behavior. xLSTM-m and the baseline xLSTM reached nearly 100% training accuracy, but xLSTM-m retained a considerably smaller validation and test gap. TimesNet, Informer, and iTransformer achieved relatively high training accuracies but substantially lower validation and test performance, indicating pronounced overfitting. FEDformer exhibited both a large generalization gap and high inter-fold variability. In contrast, PatchTST and Autoformer did not fit the training data adequately and should therefore be characterized as showing optimization failure or underfitting rather than overfitting.

5.4. Aggregate Performance, Stability, and Computational Efficiency

To provide a holistic assessment of the candidate architectures and determine their suitability for real-world structural health monitoring deployment, Table 9 consolidates the aggregate diagnostic performance, statistical reliability, and computational overhead across the entire experimental suite. This unified perspective allows for a rigorous comparison of xLSTM-m with various baselines, specifically balancing its superior predictive power with execution efficiency.

Table 9. Comparative summary of aggregate diagnostic fidelity, statistical reliability, and computational overhead across all benchmarks.

Model	Mean Acc (%)	Mean F1 (%)	Mean Prec (%)	Mean Rec (%)	Mean Std	Avg Rank	Runtime (s)
xLSTM-m	95.29	95.57	96.02	95.33	0.84	1.00	129.57
xLSTM	90.67	91.10	91.92	90.70	1.37	2.00	109.27
TimesNet	75.20	75.46	76.23	75.29	3.19	4.33	79.68
Informer	74.60	74.77	75.23	74.71	2.23	5.00	58.95
iTransformer	70.66	71.32	72.93	70.77	8.04	5.33	30.22
PatchTST	70.30	70.44	70.72	70.46	2.26	6.00	67.05
FEDformer	60.12	60.48	61.75	60.15	8.35	6.00	754.17
Autoformer	44.11	44.03	45.01	44.14	2.42	6.33	70.69

The proposed xLSTM-m architecture leads superiorly across all four classification metrics, achieving a Mean F1-score of 95.57%, representing a 4.47 percentage point improvement over the baseline xLSTM (91.10%) and a substantial 20.11 percentage point margin over the best-performing Transformer, TimesNet (75.46%). Furthermore, the precision-recall gap for xLSTM-m is the narrowest in the benchmark (96.02% vs. 95.33%, $\Delta = 0.69$),

confirming that the model maintains a high degree of diagnostic equilibrium without systematic over-prediction or under-prediction of specific damage states.

The stability analysis underscores the reliability of the xLSTM family in safety-critical applications. xLSTM-m achieved the lowest Mean Standard Deviation (0.84) and a perfect Average Rank of 1.00, indicating that it secured first place on every evaluation dataset. In stark contrast, iTransformer and FEDformer exhibit approximately tenfold higher variability (with standard deviations of 8.04 and 8.35, respectively), primarily driven by their erratic convergence and sensitivity to the initial conditions of the FBG signals.

Regarding the accuracy-to-cost trade-off, xLSTM-m is highly cost-effective for real-time monitoring. While it introduces a modest 20.3 s overhead compared to the baseline xLSTM (129.57 s vs. 109.27 s), it yields a significant 4.62 percentage point gain in Mean Accuracy. Although the iTransformer is the fastest model (30.22 s), its diagnostic fidelity remains insufficient at 70.66%. Conversely, FEDformer emerges as the least efficient option, being both the slowest architecture (754.17 s, approximately $5.8\times$ slower than xLSTM-m) and among the least accurate, demonstrating that increased computational complexity in attention-based decomposition does not necessarily translate to improved discriminative power in structural health monitoring.

The present results demonstrate within-dataset generalization under five-fold cross-validation but do not establish direct transferability to different structures, sensor layouts, or unseen environmental and operating conditions. Changes in structural dynamics, sensor number or placement, sampling characteristics, and environmental variability may alter both temporal patterns and interchannel relationships, potentially reducing classification performance. Although masked mean pooling may remain beneficial when condition-sensitive features occur at different temporal locations, cross-domain deployment may require retraining, transfer learning, domain adaptation, or out-of-distribution detection.

6. Limitations

Despite the promising results, the scope of the present validation is limited. The FBG and PCB datasets were obtained from a single laboratory-scale cable-stayed bridge model, whereas the Z24-9Setup dataset represents one full-scale bridge. Moreover, the evaluated inputs consist primarily of dynamic strain and acceleration responses. The study does not include quasi-static quantities such as cable force and girder deflection, nor environmental variables such as temperature and humidity. Consequently, the reported performance should not be interpreted as evidence of general applicability across the full SHM domain. The statistical characteristics and temporal scales of these unexamined measurements may differ substantially from those of the current datasets. For example, environmental variables generally exhibit slow seasonal or diurnal variation, while quasi-static structural responses may depend strongly on traffic loading and thermal effects. Future work should therefore assess xLSTM-m using heterogeneous multimodal datasets that combine dynamic, quasi-static, and environmental measurements, as well as different structural types, sensor layouts, and previously unseen operating conditions.

The present study focused on one hybrid xLSTM configuration. The backbone depth, hidden dimension, number of heads, and block composition were kept fixed to ensure a controlled comparison between the baseline xLSTM and the proposed xLSTM-m. This design supports a fair assessment of the readout mechanism. However, it does not provide a complete sensitivity analysis of the model architecture. Future studies should examine the influence of these hyperparameters on classification accuracy, stability, and computational cost.

Only masked mean pooling was investigated as the proposed aggregation strategy. This choice was motivated by its simplicity, low computational cost, and ability to sum-

marize all valid hidden states. Nevertheless, other sequence-level aggregation methods may also be effective. Attention pooling, max pooling, mean-max pooling, and learnable pooling mechanisms should be compared in future work.

The evaluation was mainly based on segment-level cross-validation. Although stratified five-fold cross-validation provides a rigorous benchmark, it may not fully assess generalization across independent measurement campaigns, structural systems, or operational conditions. Future work should consider grouped cross-validation protocols. Entire measurement runs, loading scenarios, or sensor setups should be withheld during testing to evaluate more realistic generalization performance.

The computational analysis primarily considered wall-clock training time. This metric is useful for comparing training efficiency. However, practical SHM deployment also requires information on inference latency, parameter count, memory consumption, and hardware requirements. These aspects should be examined before real-time or edge-device implementation.

The present study reports global classification metrics. These metrics demonstrate the overall effectiveness of the proposed model. However, they do not fully explain which damage states are most difficult to distinguish. Future work should include confusion matrices, per-class F1-scores, and interpretability analysis. Such analyses would provide deeper insight into the diagnostic behavior of xLSTM-m and its relevance for practical SHM decision-making.

From a deployment perspective, masked mean pooling is parameter-free and therefore does not increase the number of trainable parameters or the model size relative to the baseline xLSTM. It only introduces an averaging operation over the encoded hidden states, whose computational overhead is expected to be small compared with that of the xLSTM backbone. However, hardware-specific inference latency and peak memory consumption were not evaluated in the present study. Therefore, suitability for real-time or edge deployment remains to be established through device-level profiling, model compression, and quantization studies.

7. Conclusions

This study investigated the influence of sequence-level aggregation on multivariate structural-response classification through a controlled comparison between the baseline xLSTM and the proposed xLSTM-m. Eight neural architectures were evaluated on three bridge-monitoring datasets under a consistent five-fold cross-validation protocol. By replacing the last-hidden-state readout with masked mean pooling while retaining the same hybrid sLSTM-mLSTM backbone and training configuration, xLSTM-m achieved the highest overall classification performance and the lowest inter-fold variability among the evaluated models. Across the three datasets, it attained a mean accuracy of 95.29% and a mean macro-F1-score of 95.57%.

The results demonstrate that the sequence-level readout is an important architectural component in structural-response classification. Aggregating all valid hidden states enables temporally distributed information to contribute directly to the final representation, thereby reducing dependence on a single terminal state. Under the evaluated conditions, this modification consistently improved both predictive performance and stability relative to the conventional xLSTM readout and the Transformer-oriented baselines.

These findings are limited to the investigated bridge strain and acceleration datasets and should not be interpreted as evidence of universal superiority across the broader SHM domain. Further validation is required using other structural systems, sensing modalities, environmental and quasi-static responses, and evaluation settings involving

unseen operating conditions. Alternative aggregation mechanisms and hardware-specific deployment performance should also be examined in future work.

Author Contributions: Conceptualization, T.T.N.L., D.H.P., L.V.V., K.L. and N.T.C.N.; methodology, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; software, D.H.P., L.H.M., L.V.V., D.D.T., N.P.T. and K.L.; validation, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; formal analysis, T.T.N.L. and N.T.C.N.; investigation, D.H.P., L.V.V., D.D.T. and N.T.C.N.; resources, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; data curation, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; writing—original draft preparation, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; writing—review and editing, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; visualization, T.T.N.L., D.H.P., L.H.M., L.V.V., D.D.T., N.P.T., K.L. and N.T.C.N.; supervision, N.T.C.N.; project administration, N.T.C.N.; funding acquisition, T.T.N.L. and N.T.C.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Vietnam-Korea Protocol Project under contract number NDT/KR/22/11.

Data Availability Statement: Data will be made available on request.

Acknowledgments: The authors would like to express their gratitude to the Ministry of Science and Technology of Vietnam for funding this research through the Vietnam-Korea Protocol Project under the contract number NDT/KR/22/11.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Farrar, C.R.; Worden, K. An Introduction to Structural Health Monitoring. In *New Trends in Vibration Based Structural Health Monitoring*; Deraemaeker, A., Worden, K., Eds.; CISM Courses and Lectures; Springer: Vienna, Austria, 2010. [\[CrossRef\]](#)
2. Carden, E.P.; Fanning, P. Vibration Based Condition Monitoring: A Review. *Struct. Health Monit.* **2004**, *3*, 355–377. [\[CrossRef\]](#)
3. Avci, O.; Abdeljaber, O.; Kiranyaz, S.; Hussein, M.; Gabbouj, M.; Inman, D.J. A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications. *Mech. Syst. Signal Process.* **2021**, *147*, 107077. [\[CrossRef\]](#)
4. Azimi, M.; Eslamlou, A.D.; Pekcan, G. Data-Driven Structural Health Monitoring and Damage Detection through Deep Learning: State-of-the-Art Review. *Sensors* **2020**, *20*, 2778. [\[CrossRef\]](#) [\[PubMed\]](#)
5. Jia, J.; Li, Y. Deep Learning for Structural Health Monitoring: Data, Algorithms, Applications, Challenges, and Trends. *Sensors* **2023**, *23*, 8824. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Foumani, N.M.; Miller, L.; Tan, C.W.; Webb, G.I.; Forestier, G.; Salehi, M. Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey. *ACM Comput. Surv.* **2024**, *56*, 1–45. [\[CrossRef\]](#)
7. Barrias, A.; Casas, J.R.; Villalba, S. A Review of Distributed Optical Fiber Sensors for Civil Engineering Applications. *Sensors* **2016**, *16*, 748. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Wang, G.; Ke, J. Literature Review on the Structural Health Monitoring (SHM) of Sustainable Civil Infrastructure: An Analysis of Influencing Factors in the Implementation. *Buildings* **2024**, *14*, 402. [\[CrossRef\]](#)
9. Liu, J.; Liu, Z.; Ma, Z.; Tong, Y.; Wang, C.; Zhu, L.; Liu, Y. Joint Action of Wind and Temperature for a Long-Span Cable-Stayed Bridge in Plateau Canyon Regions Using SHM Data and Copula-Based Probabilistic Modeling. *Buildings* **2026**, *16*, 916. [\[CrossRef\]](#)
10. Xie, X.; Lei, Y.; Xiang, C.; Li, Y.; Liu, L. An Enhanced Generative Adversarial Imputation Network with Unsupervised Learning for Random Missing Data Imputation of All Sensors. *Struct. Control. Health Monit.* **2025**, *2025*, 8419570. [\[CrossRef\]](#)
11. Tran, M.Q.; Sousa, H.S.; Matos, J.C.; Dang, S.N.; Nguyen, H.X. A Digital Twin Framework for Structural Health Monitoring of Existing Large-Span Bridges. *Sensors* **2026**, *26*, 3293. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Shan, Y.; Wang, X.; Jing, Q.; Wu, W.; Xia, Y. Yong Xia GNN-LSTM-based full-field temperature prediction of bridges using sparse sensor data and heat-transfer analysis. *Eng. Struct.* **2025**, *342*, 120922. [\[CrossRef\]](#)
13. Tran, M.; Nhung, N.T.C.; Van, T.N.; Komarizadehasl, S. Seyedmilad Komarizadehasl Innovative experimental assessment of direct and drive-by monitoring on two truss bridges. *Measurement* **2026**, *278*, 121693. [\[CrossRef\]](#)
14. Ngo Van, T.; Quang, M.T. Damage Detection in Bridges by Hybrid Particle Swarm Optimization–Simulated Annealing (PSO–SA) Approach. *IEEE Access* **2026**, *14*, 14147–14167. [\[CrossRef\]](#)

15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2023**. [[CrossRef](#)]
16. Xu, Y.; Lin, C.; Bao, Y.; Li, H. Deep Graph Learning for Spatial-Temporal Group-Correlated In-Service Performance Prediction of Transportation Infrastructure System. *Adv. Eng. Inform.* **2026**, *73*, 104791. [[CrossRef](#)]
17. Abraham, J.N.; Tran, M.Q.; Jayaraj, J.S.; Matos, J.C.; Valluzzi, M.R.; Dang, S.N. Unsupervised Learning-Based Anomaly Detection for Bridge Structural Health Monitoring: Identifying Deviations from Normal Structural Behaviour. *Sensors* **2026**, *26*, 561. [[CrossRef](#)] [[PubMed](#)]
18. Harikumar, V.; Bijudas, C.R. Sensor integrity assessment in guided wave SHM using convolutional neural networks. *Smart Mater. Struct.* **2026**, *35*, 055039. [[CrossRef](#)]
19. Atencio, E.; Komarizadehasl, S.; Lozano-Galant, J.A.; Aguilera, M. Using RPA for Performance Monitoring of Dynamic SHM Applications. *Buildings* **2022**, *12*, 1140. [[CrossRef](#)]
20. Huang, J.; Huang, Z.; Xie, C.; Zhang, Y.; Ostachowicz, W.; Cao, M. Unsupervised deep learning framework for damage identification under ambient excitations: Trait of damage localization and demonstrative applications. *Mech. Syst. Signal Process.* **2026**, *250*, 114123. [[CrossRef](#)]
21. Vijayan, D.S.; Sivasuriyan, A.; Devarajan, P.; Krejsa, M.; Chalecki, M.; Żółtowski, M.; Kozarzewska, A.; Koda, E. Development of Intelligent Technologies in SHM on the Innovative Diagnosis in Civil Engineering—A Comprehensive Review. *Buildings* **2023**, *13*, 1903. [[CrossRef](#)]
22. Neto, M.S.; Finotti, R.; Barbosa, F.; Cury, A. Structural Damage Identification Using Autoencoders: A Comparative Study. *Buildings* **2024**, *14*, 1014. [[CrossRef](#)]
23. Lu, Y.; Zhu, Z.; Li, Y.; Liu, H.; Hou, J.; Zhou, C.; Wahab, M. LDS-former: A lightweight dual-stream transformer for real-time acoustic emission monitoring of crack evolution in offshore steel structures. *Adv. Eng. Inform.* **2026**, *74*, 104635. [[CrossRef](#)]
24. Lu, Y.; Zhang, Y.; Qiu, X.; Ren, W.; Zhao, C.; Chen, M.; Li, Y.; Bader, S.; Liu, H. Structural health monitoring of offshore pipelines via a novel spatial-topological adaptive graph neural network. *Struct. Health Monit.* **2026**. [[CrossRef](#)]
25. Dzuwa, C.; Ndovie, S.; Onososen, A.; Musonda, I.; Matsimbe, J.; Masi, F.; Masi, J.; Chibwana, J.; Nkhonjera, G.; Kafodya, I. Deep Time Series in Structural Health Monitoring of Civil Structures: A Review of Architectures, Applications and Challenges. *Preprints* **2025**, 2025052422. [[CrossRef](#)]
26. Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; Bordes, A. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017. [[CrossRef](#)]
27. Tran, M.Q.; Sousa, H.S.; Ngo, T.V.; Nguyen, B.D.; Nguyen, Q.T.; Nguyen, H.X.; Baron, E.; Matos, J.; Dang, S.N. Structural Assessment Based on Vibration Measurement Test Combined with an Artificial Neural Network for the Steel Truss Bridge. *Appl. Sci.* **2023**, *13*, 7484. [[CrossRef](#)]
28. Beck, M.; Pöppel, K.; Spanring, M.; Auer, A.; Prudnikova, O.; Kopp, M.; Klambauer, G.; Brandstetter, J.; Hochreiter, S. xLSTM: Extended Long Short-Term Memory. *arXiv* **2024**. [[CrossRef](#)]
29. Tran, M.Q.; Sousa, H.S.; Nguyen, N.T.C.; Nguyen, Q.H.; Campos e Matos, J. Opportunities and Challenges of Digital Twins in Structural Health Monitoring. In *Proceedings of the 4th International Conference on Sustainability in Civil Engineering*; Nguyen-Xuan, T., Nguyen-Viet, T., Bui-Tien, T., Nguyen-Quang, T., De Roeck, G., Eds.; ICSCE 2022. Lecture Notes in Civil Engineering; Springer: Singapore, 2024; Volume 344. [[CrossRef](#)]
30. Ribeiro, D.; Rakoczy, A.M.; Cabral, R.; Hoskere, V.; Narazaki, Y.; Santos, R.; Tondo, G.; Gonzalez, L.; Matos, J.C.; Massao Futai, M.; et al. Methodologies for Remote Bridge Inspection—Review. *Sensors* **2025**, *25*, 5708. [[CrossRef](#)] [[PubMed](#)]
31. Quyet, N.H.; Vu, L.V.; Hoa, T.N. Damage classification of steel frame using one-dimensional convolutional neural network and channel attention mechanism. *Transp. Commun. Sci. J.* **2024**, *75*, 2333–2344. [[CrossRef](#)]
32. Yessoufou, F.; Zhu, J. Classification and regression-based convolutional neural network and long short-term memory configuration for bridge damage identification using long-term monitoring vibration data. *Struct. Health Monit.* **2023**, *22*, 4027–4054. [[CrossRef](#)]
33. Le-Xuan, T.; Nguyen-Chi, T.; Bui-Tien, T.; Tran-Ngoc, H. ResUNet4T: A potential deep learning model for damage detection based on a numerical case study of a large-scale bridge using time-series data. *Eng. Struct.* **2025**, *340*, 120668. [[CrossRef](#)]
34. Son, H.; Pham, V.-T.; Jang, Y.; Kim, S.-E. Damage Localization and Severity Assessment of a Cable-Stayed Bridge Using a Message Passing Neural Network. *Sensors* **2021**, *21*, 3118. [[CrossRef](#)] [[PubMed](#)]
35. Mantawy, I.M.; Mantawy, M.O. Convolutional neural network based structural health monitoring for rocking bridge system by encoding time-series into images. *Struct. Control Health Monit.* **2021**, *29*, e2897. [[CrossRef](#)]
36. Azimi, M.; Pekcan, G. Structural health monitoring using extremely compressed data through deep learning. *Comput.-Aided Civ. Infrastruct. Eng.* **2020**, *35*, 597–614. [[CrossRef](#)]
37. Li, Z.; Li, D.; Sun, T. A Transformer-Based Bridge Structural Response Prediction Framework. *Sensors* **2022**, *22*, 3100. [[CrossRef](#)] [[PubMed](#)]

38. Liu, J.; Li, Q.; Li, L.; An, S. Structural damage detection and localization via an unsupervised anomaly detection method. *Reliab. Eng. Syst. Saf.* **2024**, *252*, 110465. [CrossRef]
39. Zhou, F.; Impraïmakis, M. Transformer self-attention encoder–decoder with multimodal deep learning for response time series forecasting and digital twin support in wind structural health monitoring. *Comput. Struct.* **2026**, *326*, 108216. [CrossRef]
40. Peeters, B.; De Roeck, G. One-year monitoring of the Z24-Bridge: Environmental effects versus damage events. *Earthq. Eng. Struct. Dyn.* **2000**, *30*, 149–171.
41. Minh, T.Q.; Matos, J.C.; Sousa, H.S.; Ngoc, S.D.; Van, T.N.; Nguyen, H.X.; Nguyễn, Q. Data reconstruction leverages one-dimensional Convolutional Neural Networks (1DCNN) combined with Long Short-Term Memory (LSTM) networks for Structural Health Monitoring (SHM). *Measurement* **2025**, *253*, 117810. [CrossRef]
42. Minh, T.Q.; Van, T.N.; Nguyen, H.X.; Nguyễn, Q. Enhancing the Structural Health Monitoring (SHM) through data reconstruction: Integrating 1D convolutional neural networks (1DCNN) with bidirectional long short-term memory networks (Bi-LSTM). *Eng. Struct.* **2025**, *340*, 120767. [CrossRef]
43. Das, A.; Kong, W.; Sen, R.; Zhou, Y. A decoder-only foundation model for time-series forecasting. *arXiv* **2024**. [CrossRef]
44. Rasul, K.; Seward, C.; Schuster, I.; Vollgraf, R. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. *arXiv* **2021**. [CrossRef]
45. GitHub—Thuml/Time-Series-Library A Library for Advanced Deep Time Series Models for General Time Series Analysis. 2026. Available online: <https://github.com/thuml/Time-Series-Library> (accessed on 31 March 2026).
46. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. *arXiv* **2023**. [CrossRef]
47. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *arXiv* **2021**. [CrossRef]
48. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *arXiv* **2022**. [CrossRef]
49. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. *arXiv* **2022**. [CrossRef]
50. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv* **2023**. [CrossRef]
51. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *arXiv* **2024**. [CrossRef]
52. Bagnall, A.; Dau, H.A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; Keogh, E. The UEA multivariate time series classification archive, 2018. *arXiv* **2018**. [CrossRef]
53. Ruiz, A.P.; Flynn, M.; Large, J.; Middlehurst, M.; Bagnall, A. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Disc* **2021**, *35*, 401–449. [CrossRef] [PubMed]
54. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
55. Zhang, J.; Zhang, J.; Wu, Z. Long-Short Term Memory Network-Based Monitoring Data Anomaly Detection of a Long-Span Suspension Bridge. *Sensors* **2022**, *22*, 6045. [CrossRef] [PubMed]
56. Alharthi, M.; Mahmood, A. xLSTMTIME: Long-Term Time Series Forecasting with xLSTM. *AI* **2024**, *5*, 1482–1495. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.