Mechanical Systems and Signal Processing 227 (2025) 112319

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

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

Invited for the Special Issue in Honor of Professor John Mottershead

Probabilistic machine learning pipeline using topological descriptors for real-time state estimation of high-rate dynamic systems

Yang Kang Chua [a], Daniel Coble [b,1], Arman Razmarashooli [c], Steve Paul [a], Daniel A. Salazar Martinez [c], Chao Hu [a,*], Austin R.J. Downey [b,d], Simon Laflamme [c,e]

[a] School of Mechanical, Aerospace, and Manufacturing Engineering, University of Connecticut, Storrs, CT, USA
[b] Department of Mechanical Engineering, University of South Carolina, Columbia, SC, USA
[c] Department of Civil, Construction, and Environmental Engineering, Iowa State University, Ames, IA, USA
[d] Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, USA
[e] Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA

ARTICLE INFO

Communicated by Z. Mao

Keywords:
Structural health monitoring
High-rate systems
Nonlinear time series
Topological data analysis
Probabilistic machine learning
Uncertainty quantification

ABSTRACT

High-rate systems are structures that undergo rapid changes, exhibiting dynamics that evolve over short durations, often less than 100 ms. In this study, we propose a probabilistic machine learning pipeline for estimating the state of a high-rate system. Our approach begins with the extraction of features using topological data analysis (TDA) that capture the underlying structure of datasets. We examine the design of probabilistic models for structural state estimation, emphasizing the importance of prediction intervals. Our method validation involves two datasets: a toy example of linear chirp signals and an experimental dataset from the Dynamic Reproduction of Projectiles in Ballistic Environments for Advanced Research (DROPBEAR) testbed. We use metrics such as mean absolute error (MAE) and time response assurance criterion (TRAC), along with uncertainty metrics such as negative log-likelihood (NLL), calibration curves, and expected calibration error (ECE), to evaluate model performance. The results indicate that the RNN–NNE model achieves the lowest MAE of 5.705 mm, the highest TRAC, and the lowest ECE of 7.335%, highlighting its superior predictive accuracy and robustness in handling uncertainty.

1. Introduction

High-rate systems undergo rapid and extreme changes within short timeframes, commonly encountered in aerospace, automotive safety, and impact engineering. These systems typically experience dynamic events with magnitudes surpassing 100 g_n and durations under 100 ms [1]. Examples include hypersonic vehicles, active blast mitigation, and ballistic packages [2–4]. Real-time feedback on structural integrity can significantly enhance survivability. However, the complexity of this research lies in large uncertainties in external loads, high non-stationarities, heavy disturbances, and unmodeled dynamics due to system configuration changes [5]. These challenges complicate real-time state estimation, which is critical in empowering timely decision-making and ensuring the safe operation and structural integrity of these systems.

* Corresponding author.
E-mail address: chao.hu@uconn.edu (C. Hu).
1 Present affiliation: Thomas Lord Department of Mechanical Engineering & Materials Science, Duke University, Durham, NC, USA.

https://doi.org/10.1016/j.ymssp.2025.112319
Received 3 October 2024; Received in revised form 20 December 2024; Accepted 3 January 2025
Available online 27 January 2025

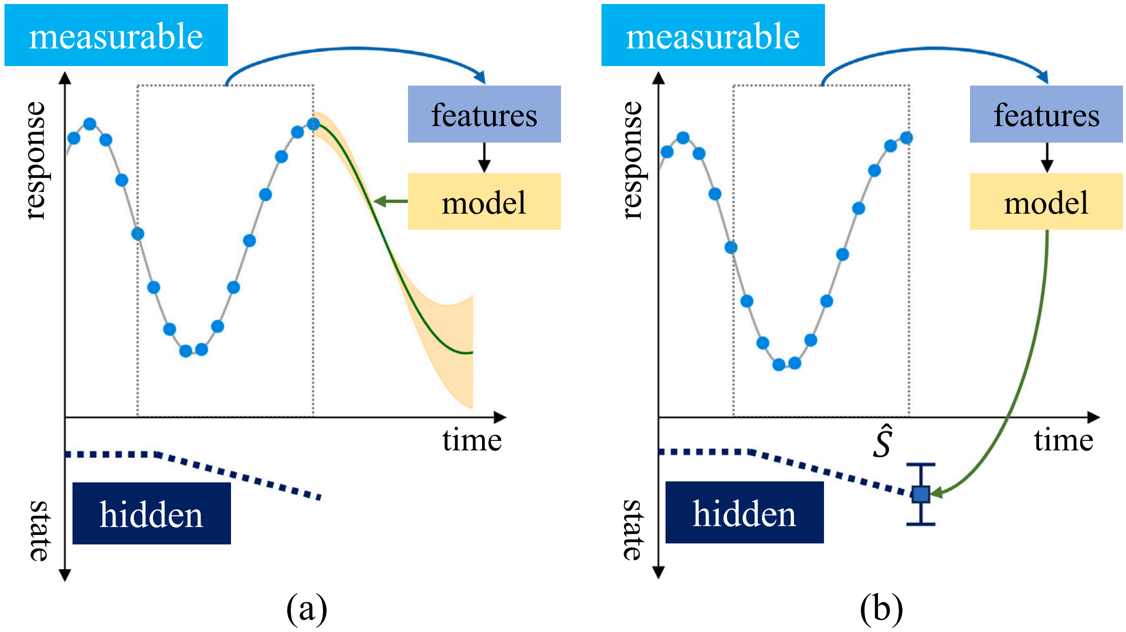

Fig. 1. Overview of the high-rate state estimation problem. (a) Forecasting for high-rate dynamic events. (b) State estimation for real-time structural health monitoring.

Various methods have been proposed to address these challenges, including both physics-based and data-driven techniques. Fig. 1 provides a broad overview of these approaches: (a) focuses on forecasting high-rate dynamic events, and (b) highlights state estimation methods used for real-time structural health monitoring. For instance, Joyce et al. [6] developed a sliding mode observer using experimental data from an accelerometer mounted on a beam that was subjected to a moving boundary condition. This observer tracks the position of a moving cart by identifying the fundamental frequency in real-time. Similarly, other physics-based models have demonstrated effectiveness in real-time model updating and achieving sub-millisecond computational speeds [7,8]. For example, Ogunniyi et al. [9] showed that seven finite element analysis models, each with 250 nodes, can be solved in parallel in under a millisecond, with a single 250-node model taking about 250 microseconds. However, while these techniques show promise for tracking and parameter estimation in controlled experimental setups, their effectiveness in more complex, real-world applications—where system dynamics are nonlinear or non-stationary—remains to be fully explored.

Topological data analysis (TDA) is a method that applies concepts from algebraic topology to study the shape of data using modern mathematical tools. By examining the topological features of datasets, TDA can reveal complex patterns and structures that are not immediately apparent through traditional methods. Recent advancements in TDA have showcased its power in feature extraction, particularly in the context of structural health monitoring [10] and time series analysis [11–13]. This approach has been effective in linking topological features to fundamental dynamic characteristics, such as the first fundamental frequency [14]. Utilizing TDA helps us gain deeper insights into the data structure and enhances our modeling and monitoring capabilities, especially in high-rate systems where dynamics are non-stationary.

In addition to TDA, data-driven techniques have gained prominence in structural health monitoring, particularly with the adoption of recurrent neural networks (RNNs), including long short-term memory (LSTM) networks. RNNs, and LSTM networks in particular, have shown effective in modeling complex, nonlinear time series data across various fields [15–17]. As a specific form of RNN, LSTMs are inherently suited for representing dynamic systems due to their ability to capture long-term temporal dependencies. This makes them particularly suitable for applications in high-rate environments. However, while these approaches address challenges like data scarcity and modeling complexity, they often fall short in providing effective uncertainty quantification (UQ). UQ is crucial for making informed decisions by assessing the confidence in predictions, especially in high-stakes scenarios [18].

UQ is increasingly recognized as a critical component in machine learning and deep learning applications. It manages uncertainties arising from various sources, such as mismatches between training and test data, noise, and model inaccuracies [19]. Evaluating the efficacy of artificial intelligence systems before deployment is crucial to ensure they handle uncertainties effectively [20]. For instance, the integration of UQ in machine learning models and its calibration plays a vital role in ensuring safety and reliability in artificial intelligence systems [21–24]. Addressing UQ effectively enhances the robustness and accuracy of predictive models, particularly in high-stakes environments.

This paper introduces **P**robabilistic **P**rediction for **S**tructural **H**ealth **M**onitoring (ProbPredict-SHM), a probabilistic machine learning pipeline designed for enabling real-decision systems for in high-rate dynamics. The ProbPredict-SHM pipeline integrates TDA for advanced feature extraction, enhancing the model's ability to interpret complex data patterns and improve estimation accuracy. To illustrate fundamental concepts and assess the pipeline's performance, we first apply it to a controlled toy example

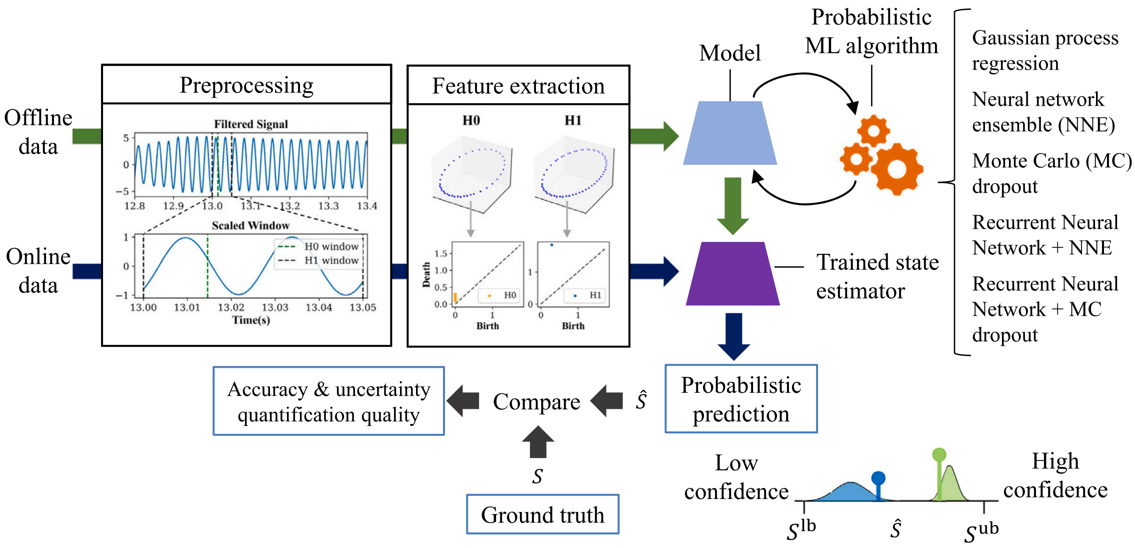

Fig. 2. Overview of the probabilistic machine learning pipeline using TDA features extraction. S^{lb} and S^{ub} represent the lower and upper bounds for state estimation, respectively.

involving linear chirp signals. This example serves as a preliminary assessment of ProbPredict-SHM's capabilities. Additionally, the pipeline's performance is rigorously evaluated using datasets from the Dynamic Reproduction of Projectiles in Ballistic Environments for Advanced Research (DROPBEAR) testbed, which simulates high-rate dynamic events. This comprehensive evaluation demonstrates the ProbPredict-SHM's capability to provide accurate real-time state estimates and effective UQ, contributing to more reliable structural health monitoring in dynamic environments. The primary contributions of this paper are: (1) the development of a real-time machine learning pipeline for structural health monitoring; (2) the application of TDA-based feature extraction; and (3) the implementation of comprehensive UQ.

The remainder of this paper is organized as follows: Section 2 provides an overview of the ProbPredict-SHM and its integration with TDA. Section 3 details the methodology employed for implementing the probabilistic machine learning framework. Sections 4 and 5 present and analyze two case studies: one utilizing synthetic data and the other evaluating the ProbPredict-SHM on the DROPBEAR testbed. Finally, Section 6 discusses the practical applicability of the ProbPredict-SHM and offers recommendations for future research and development.

2. Overview of probabilistic machine learning pipeline

State estimation is addressed from a machine learning perspective as a multivariate supervised regression problem. Our approach uses a pipeline-based methodology to engineer features from high-rate data for training both probabilistic and non-probabilistic models. This pipeline focuses on the offline stages of feature engineering, algorithm training, and uncertainty calibration to ensure robustness, accuracy, and reliability before any potential application in real-time scenarios. An overview of the ProbPredict-SHM pipeline is illustrated in Fig. 2.

Our methodology incorporates techniques from TDA, specifically persistent homology, to capture the topological characteristics of the data. The time series data is converted into delay vectors—representations of the original data that capture temporal relationships—using a technique called time delay embedding. This is followed by constructing sliding windows for feature extraction and applying persistent homology to derive 10 distinct features. Detailed explanations of these steps are provided in the Methodology section. The dataset, now enriched with TDA features, serves as input for various algorithms, including probabilistic models such as neural network ensembles (NNE) and Monte Carlo (MC) dropout, as well as advanced techniques that comprise ensemble-type RNNs and MC dropout-type RNNs. For hyperparameter tuning, all algorithms use a grid search approach to determine parameters such as the number of layers and ensemble size, which helps prevent overcomplicating the model. Adam optimization is used to train the weights of the neural network-based models. Techniques such as early stopping and regularization are applied where applicable to prevent overfitting.

The performance of the different machine learning models is assessed using a variety of metrics, including mean absolute error (MAE), time response assurance criterion (TRAC), and uncertainty estimation metrics such as negative log-likelihood (NLL) and expected calibration error (ECE). Detailed descriptions of these metrics are provided in Section 3.

3. Methodology

We develop a probabilistic machine learning pipeline for structural state estimation of high-rate systems, called ProbPredict-SHM. This approach involves hierarchical steps, including preprocessing, feature engineering using TDA, and normalization before model fitting and tuning.

3.1. Preprocessing and feature engineering

ProbPredict-SHM begins with applying a low-pass filter to the raw data to enhance data quality and reduce noise. This filtering process eliminates high-frequency noise and captures the dominant signal components, mitigating the effects of non-stationarity in high-rate dynamic systems. The filtered acceleration data collected from sensors serves as the primary input for ProbPredict-SHM. Once the data is preprocessed, TDA is used for feature extraction. The collected signal data, initially a one-dimensional time series, is converted into higher-dimensional point clouds with a delay vector embedding, as shown in Eq. (1). The embedding theorem guarantees that, for a stationary system and with certain conditions on forcings, the full dynamics of the system can be recovered with an appropriate choice of embedding dimension d [25].

$$\chi(t) = [x(t), x(t-\tau), x(t-2\tau), \dots, x(t-(d-1)\tau)] \tag{1}$$

with τ representing the time delay. The embedding dimension d is typically determined using the false nearest neighbor test, while the time delay τ is selected based on mutual information [26,27]. The delay vector, if appropriately constructed, preserves the essential dynamics of the system. Under this hypothesis, TDA techniques can be employed to extract dynamically meaningful features from the resulting point clouds. Here, these features are used to build and refine machine learning models to estimate the state of the high-rate dynamic system.

To address nonstationarity due to moving boundary conditions, two sliding windows extract local TDA features, assuming stationarity within these windows. This assumption generally holds for our systems of interest, given that their dynamics return to a stationary behavior following a disturbance. Given the sampling frequency, maximum frequency ($f_{\max}$), and minimum frequency ($f_{\min}$) of the data, the two sliding windows (H_0 and H_1) are constructed with specific sizes [14]. The H_0 window captures TDA features related to connected components or zero-dimensional holes, while the H_1 window captures features related to loops or one-dimensional holes. The time delay τ is computed using:

$$\tau = \frac{0.25}{f_{\max}} \tag{2}$$

This strategy ensures that the embedded signal forms a circle at $f_{\max}$, distinguishing shapes at lower frequencies that resemble ellipses. The maximum allowable time delay τ is chosen to avoid folding the topological space onto itself, which would lead to information loss. It is recommended to choose a time delay τ lower than this maximum value to further mitigate the risk of information loss. For H_0 feature extraction, the window size is $\frac{1}{f_{\max}} + 2\tau$, and for H_1 feature extraction, it is $\frac{1}{f_{\min}} + 2\tau$. These windows are min–max scaled to standardize them, focusing on temporal aspects rather than amplitude. The scaled windows are converted into point clouds using the calculated time delays (τ).

To reduce computational demands, the H_1 window is moved during the sliding process with a step size s, while the H_0 window is constructed to keep its final point consistent with that of the H_1 window. This approach ensures that the collected data remain consistent. With an embedding dimension d set to 3, TDA techniques, specifically persistent homology [28], are used to extract 10 features (5 from H_0 and 5 from H_1) that numerically represent the topology of the point clouds. Finally, the feature data are normalized to ensure consistent scaling, which promotes stable training and improved convergence of the machine learning training process.

The effectiveness of this feature extraction method has been validated in [14], where it was demonstrated that the maximum persistence of the 0th and 1st-dimensional persistence homology groups, specifically H_0 and H_1, can provide stable estimations of system states, such as cart location, with reduced noise levels compared to other TDA features. While TDA remains a relatively new area of research, we are actively investigating the potential of TDA features to create more efficient representations of dynamic systems. Although TDA has shown considerable promise in our studies, we are also open to exploring alternative feature extraction methods. Nevertheless, the results thus far suggest that TDA features can serve as effective inputs for machine learning models, offering stable estimates of system states, such as cart location, with lower noise levels when compared to other feature extraction techniques. This reinforces our confidence in the utility of TDA for our system. Detailed descriptions of each feature are provided in Appendix A.

3.2. Model development

This section describes the background and methodology of the machine learning models used to evaluate the ProbPredict-SHM pipeline. The models include Gaussian process regression (GPR), neural network ensemble (NNE), Monte Carlo (MC) dropout, and recurrent neural network (RNN), each chosen for their unique strengths in handling high-rate data and providing UQ.

Gaussian process regression (GPR). GPR is a powerful non-parametric, probabilistic model that provides both predictions and uncertainty estimates, making it ideal for UQ. Unlike traditional parametric models, GPR defines a distribution over possible functions that fit the data, offering flexibility for complex datasets.

GPR's mathematically rigorous approach to predictive uncertainty and its inherent distance-aware property allow it to quantify uncertainty based on the proximity of test points to the training data. It produces low uncertainty for test points close to the training distribution and high uncertainty as points move farther away, making it particularly effective at detecting out-of-distribution (OOD) samples. This ability to distinguish between in-distribution and OOD points establishes GPR as a benchmark for reliable UQ and a gold standard for evaluating other probabilistic models.

GPR begins with a Gaussian process before the unknown function: $F(x) \sim \mathcal{GP}(m(x), k(x, x'))$ [29]. This Gaussian process prior is fully characterized by a mean function $m(x)$ and a covariance function $k(x, x')$. Here, $x \in \mathbb{R}^P$ is an arbitrary input variable, and the mean function $m(x)$ and covariance kernel $k(x, x')$ are defined as $m(x) = \mathbb{E}[F(x)]$ and $k(x, x') = \text{cov}(F(x), F(x'))$. The kernel function determines the properties of the function, such as smoothness and periodicity. For our implementation, we use the standard radial basis function (RBF) kernel and the Matérn kernel for hyperparameter tuning, as detailed in [29].

Neural network ensemble (NNE). NNE improves upon traditional neural networks by combining multiple models to capture both epistemic and aleatory uncertainty. It integrates multiple neural networks into a single meta-regressor, enhancing prediction accuracy, robustness, and the quality of uncertainty estimates compared to individual neural networks. NNE is also scalable to high-dimensional problems, making it a powerful tool for real-world applications.

Following the approach proposed in [30], our NNE comprises several deep neural networks, each with 5 hidden layers. Each network's output layer is designed to predict two components: the mean, $\mu_i(x)$, and the variance, $\sigma_i^2(x)$, where $\sigma_i^2(x) > 0$. This architecture allows the ensemble to generate predictions along with their associated uncertainty. The overall ensemble means $\mu(x)$ and variance $\sigma^2(x)$ are computed as follows:

$$\mu(x) = \frac{1}{M} \sum_{i=1}^{M} \mu_i(x) \tag{3}$$

$$\sigma^2(x) = \frac{1}{M} \sum_{i=1}^{M} \left(\sigma_i^2(x) + \mu_i^2(x)\right) - \mu^2(x) \tag{4}$$

where M is the number of models in the ensemble. Each neural network in the ensemble is trained using the negative log-likelihood (NLL) as the loss function, defined as:

$$Loss = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \log(\sigma_i^2(x)) + \frac{(y - \mu_i(x))^2}{2\sigma_i^2(x)} \right) \tag{5}$$

where N represents the number of data points, and y is the true value. This approach ensures that the models fit the data well while providing reliable uncertainty estimates. By combining the predictions from multiple neural networks, the ensemble method reduces the risk of overfitting and increases the model's robustness, leading to more accurate and reliable predictions.

Monte Carlo (MC) dropout. MC dropout, initially introduced as a regularization technique to mitigate overfitting in deep neural networks [31], has emerged as a promising method for approximating posterior predictive distributions in Bayesian neural networks [32].

In MC dropout, dropout layers are added after each fully connected layer of the DNN. During training, these dropout layers introduce randomness by stochastically dropping connections between neurons, creating a randomized sparse network. At test time, multiple forward passes through the model are conducted with different dropout patterns, resulting in an ensemble of predictions. This ensemble can then be leveraged to estimate prediction uncertainty.

One of the key advantages of MC dropout is its simplicity of implementation, requiring minimal modifications to existing DNN architectures. It exhibits low computational cost and scalability, making it applicable to various types of neural networks, including convolutional neural networks (CNNs) and RNNs [33,34].

Probabilistic RNN. RNN is a type of neural network model that handles sequential data [35]. It is mostly applied to time series prediction, video analysis, and musical information retrieval, where a model must learn from sequence inputs. RNN has a recurrent connection where the last hidden state is an input to the next state. The update of states can be described as follows:

$$h_t = \sigma(W x_t + U h_{t-1} + b) \tag{6}$$

where $x_t \in \mathbb{R}^M$ and $h_t \in \mathbb{R}^N$ are the input and hidden state at time step t, respectively. $W \in \mathbb{R}^{N\times M}$, $U \in \mathbb{R}^{N\times N}$, and $b \in \mathbb{R}^N$ are the weights for the current input and the recurrent input, and the bias of the neurons. σ is an element-wise activation function of the neurons, and N is the number of neurons in this RNN layer.

To convert an RNN into a probabilistic model, we can incorporate the methods described above—NNE and MC Dropout—into the RNN architecture. These approaches allow the model to not only make predictions but also quantify the uncertainty associated with those predictions, which is crucial for applications where understanding the confidence of the model's output is important.

Model selection rationale. While Markov chain Monte Carlo (MCMC) and variational inference provide high-quality uncertainty quantification, they suffer from significant computational costs, limited scalability, and implementation complexity. These methods require substantial resources for training and inference, particularly for high-dimensional data, which makes them impractical for real-time or large-scale applications. Furthermore, converting a deterministic model into a probabilistic one using MCMC or variational inference involves intricate setups, adding further implementation challenges. In contrast, MC Dropout offers a simpler, computationally efficient alternative that still delivers effective uncertainty quantification, making it a more practical choice for our study.

3.3. Metrics for performance evaluation

We evaluate the performance of our models using several metrics to ensure a robust and comprehensive assessment. These metrics include mean absolute error (MAE), time response assurance criterion (TRAC), negative log-likelihood (NLL), and expected calibration error (ECE).

Mean absolute error (MAE). MAE is a measure of errors between paired observations expressing the same phenomenon. It is calculated as:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{y}_i - y_i\right| \tag{7}$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and N is the total number of samples. MAE is simple to interpret and provides an absolute measure of the average error magnitude.

Time response assurance criterion (TRAC). TRAC is used to evaluate a system's dynamic response. It compares the predicted and actual responses over time [36]. The calculation of TRAC is given by:

$$\mathrm{TRAC} = \frac{(y^T\hat{y})^2}{(y^T y)(\hat{y}^T\hat{y})} \tag{8}$$

TRAC ranges from 0 to 1, with values closer to 1 indicating better model performance in capturing the system's dynamic behavior.

Negative log likelihood (NLL). NLL measures the fit of a probabilistic model to the validation or test data by evaluating the likelihood of observing the actual target values given the model's predictions [37]. An example of the NLL has been given in Eq. (5) as the loss function for training a probabilistic neural network in a NNE.

A lower NLL indicates a better fit of the model to the data, as it reflects higher likelihoods for the observed outcomes. NLL is an indirect indicator of model calibration and is frequently used with calibration metrics to evaluate the quality of predictive uncertainty.

Expected calibration error (ECE). ECE measures the discrepancy between the expected confidence of predictions and the observed confidence, providing a quantitative assessment of how well the predicted probabilities reflect actual outcomes [38]. To understand ECE, it is essential first to consider the concept of a calibration curve, which plots the predicted confidence levels against the observed frequency of correct predictions. A well-calibrated model will have a calibration curve close to the diagonal line, where predicted confidence matches observed accuracy. It is defined as:

$$\mathrm{ECE} = \sum_{j=1}^{K} w_j\left|\hat{c}_j - c_j\right| \tag{9}$$

where K represents the number of confidence bins equally spaced between 0 and 1, such that $0 \le c_1 < c_2 < \cdots < c_K \le 1$. The weight w_j can be either a constant $\frac{1}{K}$ or proportional to the number of samples falling into each bin. In our approach, we calculate $\hat{c}_j$ as the observed confidence level for the jth confidence bin. This is computed as:

$$\hat{c}_j = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i \text{ is in } PI_j) \tag{10}$$

Here, N is the total number of predictions, and $\mathbb{I}(\text{prop})$ is an indicator function that equals 1 if the true value y_i falls within the prediction interval PI_j and 0 otherwise. The prediction intervals PI_j for the jth bin are generated from a normal distribution using confidence bins c_j, the predicted mean $\hat{y}_i$, and variance $\hat{\sigma}^2$. The ECE provides a single-number summary of how well-calibrated the model is, where a lower ECE indicates better calibration.

Test time. Test time is an important metric for evaluating the efficiency of the ProbPredict-SHM pipeline. We calculate the test time as the sum of three main components: (1) signal preprocessing time, (2) feature extraction time (consumed by TDA), and (3) model prediction time (e.g., one or multiple forward passes for NN-based models). This evaluation helps determine the practicality of the pipeline for potential real-time applications. By analyzing feature extraction and prediction times, we aim to ensure that the ProbPredict-SHM pipeline is robust and feasible for real-time implementation.

These metrics provide a comprehensive evaluation of the models, capturing both the predictions' accuracy and uncertainty.

4. Case study 1: Linear chirp signals

In this case study, we introduce a synthetic dataset designed to emulate the DROPBEAR testbed dynamics (Case Study 2) and thus provide a first level assessment of the ProbPredict-SHM pipeline. The dataset utilizes linear chirp signals, representing dynamic systems with varying frequencies over time, providing a controlled setting to evaluate ProbPredict-SHM's performance and reliability. The linear chirp signal is mathematically defined as:

$$x(t) = \cos\left(2\pi\left(\frac{(f_1 - f_0)}{2T}t^2 + f_0 t\right)\right), \tag{11}$$

where f_0 denotes the initial frequency, f_1 represents the final frequency, and T is the total duration of the signal. The quadratic term t^2 indicates a linear change in frequency over time, simulating dynamic frequency variations in the system.

Fig. 3 illustrates the generated signals for the training phase. The training signal is constructed with frequencies ranging from 10 Hz to 15 Hz. Specifically, the signal maintains 10 Hz for the first two seconds (0–2 s), increases linearly to 15 Hz over the next two seconds (2–4 s), remains at 15 Hz for an additional two seconds (4–6 s), then decreases back to 10 Hz over the subsequent two seconds (6–8 s), and finally stays at 10 Hz for the last two seconds (8–10 s).

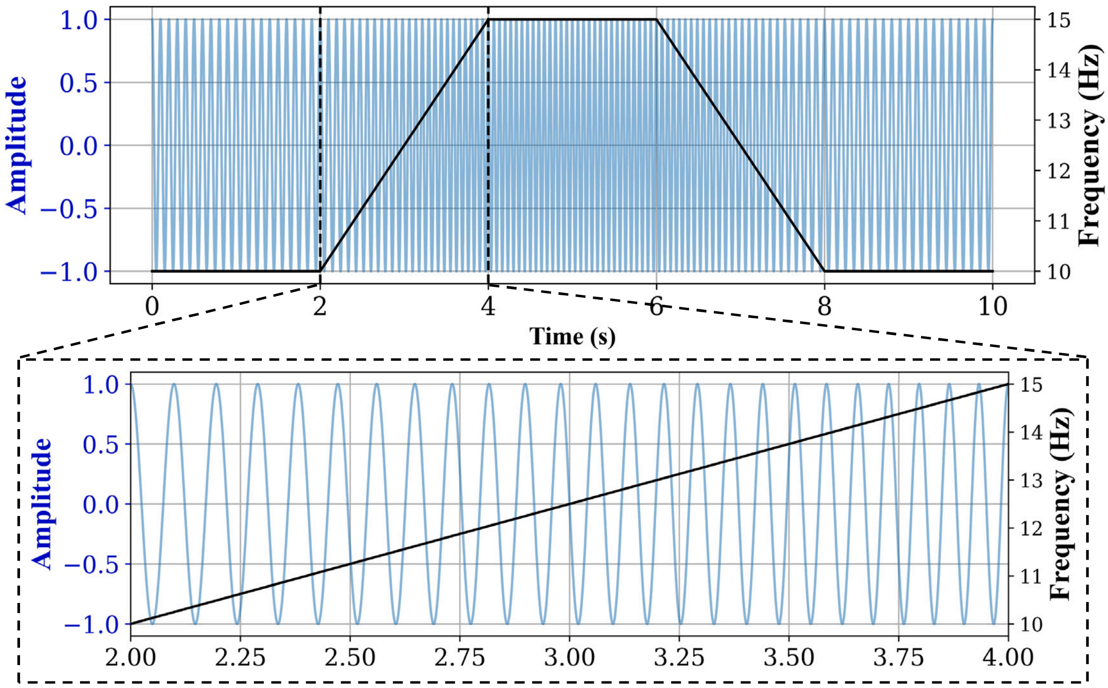

Fig. 3. Synthetic signal generated for training.

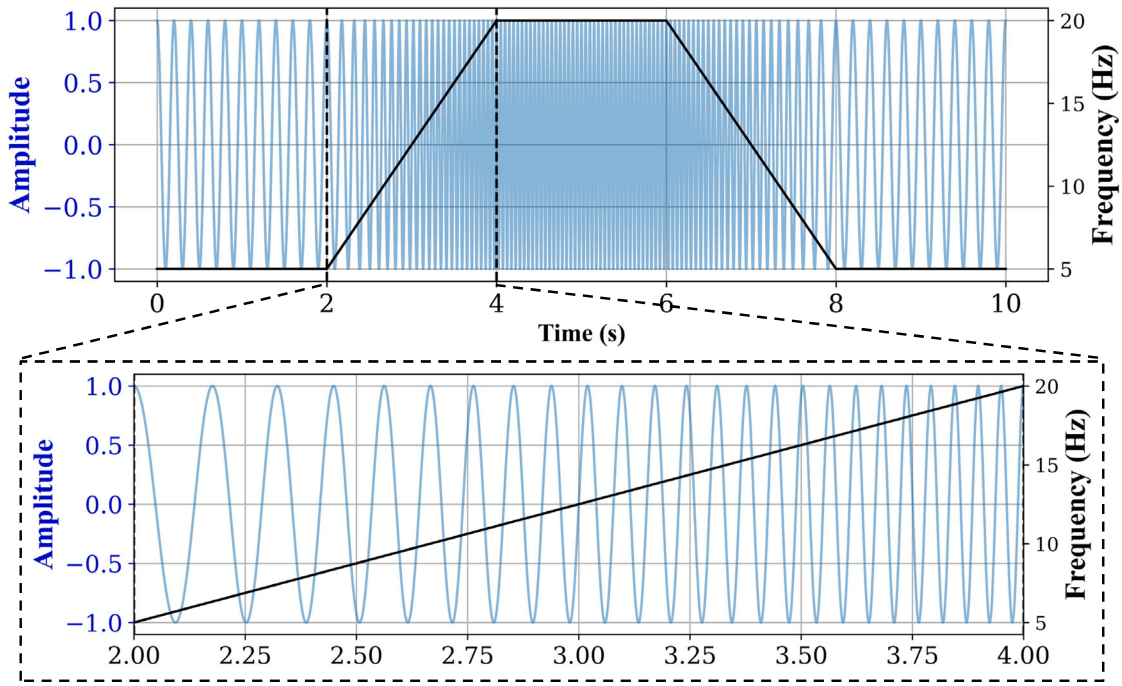

Fig. 4. Synthetic signal generated for testing.

To evaluate model performance with out-of-domain frequencies, we introduce a testing signal that extends beyond the training domain, spanning frequencies from 5 Hz to 20 Hz. This signal maintains a similar pattern to the training signal but introduces new frequency ranges. Fig. 4 highlights the new frequency ranges introduced for this evaluation, providing a robust test for the model's ability to handle such OOD frequencies. Both the training and testing signals underwent feature extraction using the TDA parameters

Table 1
Parameters for TDA feature extraction for Case Study 1. The chosen time delay τ is 0.0083 s, half of the maximum allowable value of 0.016 s, computed using Eq. (2), to prevent folding of the topological space.

Parameter	Value (unit)
H_0 Window Size	0.1 (s)
H_1 Window Size	0.1333 (s)
Time Delay (τ)	0.0083 (s)
Embedding Dimension (d)	3
Window Step Size	1 (step)

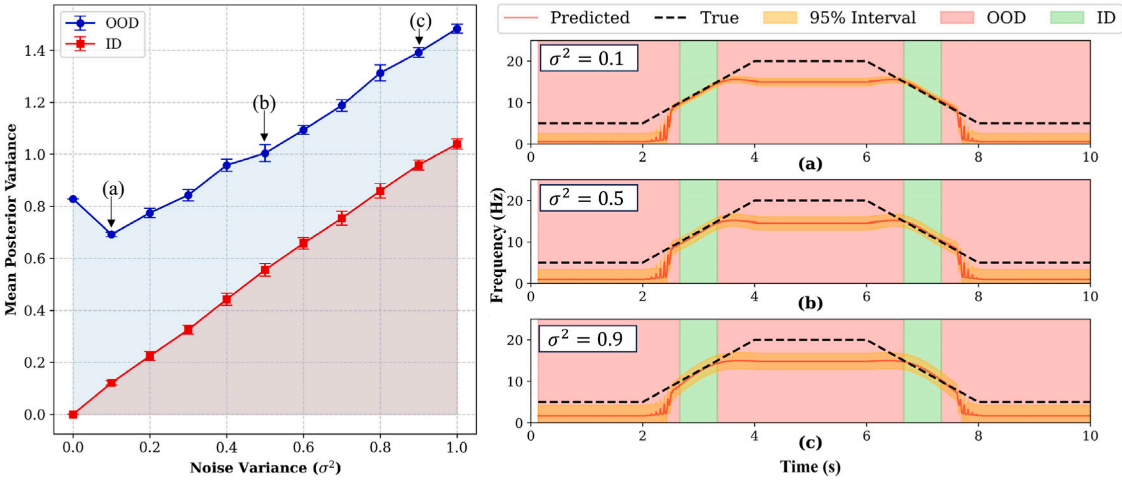

Fig. 5. Results of a parametric study on the total predictive uncertainty vs. measurement noise for Case Study 1. The left plot shows mean posterior variance vs. noise variance for in-distribution (ID) and out-of-distribution (OOD) test samples. The mean posterior variance is calculated as the average of GPR-produced posterior variances across all ID or OOD test samples. The right plot consists of three sub-plots that show the mean predictions, prediction intervals, and ground truth for three noise variance values: (a) 0.1, (b) 0.5, and (c) 0.9.

outlined in Table 1. These parameters are calculated based on the training frequency range and applied consistently to the testing signal. This approach simulates a real-world scenario where we expect that incoming signals will not deviate significantly from the training bounds.

Preprocessing steps, including low-pass filtering and scaling, are part of the pipeline; however, they are not applied in this instance. This choice is based on the chirp signal's inherent characteristics: it is free of noise and maintains a constant amplitude, which allows for a direct and unaltered assessment of the pipeline's performance. Additionally, for this study, we utilize only two features—H_1 and H_0 maximum persistence—extracted from the signal as inputs for our model.

4.1. Uncertainty evaluation on linear chirp signals

In this section, we analyze how aleatory and epistemic uncertainties evolve with varying noise levels. Aleatory uncertainty represents the inherent variability or noise within the data, while epistemic uncertainty reflects the model's lack of knowledge or uncertainty about the data, especially in regions not covered by the training set. To illustrate these uncertainties, we apply GPR to linear chirp signals. Fig. 5 shows how the uncertainties are affected by different noise levels.

A notable reduction in uncertainty is observed when the noise variance is incrementally increased from 0 to 0.1. Initially, with no added noise, the model exhibits heightened sensitivity to OOD points, resulting in elevated uncertainties. This heightened sensitivity arises because the model tends to overfit the training data in the absence of noise, making it less robust and overly reactive to deviations or new patterns. Consequently, uncertainties for OOD points can be significantly elevated. Conversely, introducing a slight increase in noise variance reduces the model's sensitivity to OOD points, leading to a substantial decrease in uncertainty. This initial addition of noise aids in regularizing the model, thereby enhancing its stability and reducing sensitivity to variations in OOD points that are near the training data.

As shown in Fig. 5, as noise variance increases, the ID posterior variance also increases, representing aleatory uncertainty. This rise in ID posterior variance indicates that aleatory uncertainty grows with higher noise levels. However, it is observed that the OOD posterior variance remains consistently higher than the ID posterior variance. This higher OOD posterior variance is due to aleatory noise exacerbating the uncertainty associated with OOD points. The consistent gap between ID and OOD posterior variances suggests that the excess uncertainty observed in the OOD predictions is predominantly epistemic. This implies that, while aleatory uncertainty increases with noise, the larger gap highlights an additional layer of uncertainty arising from the model's lack of knowledge about

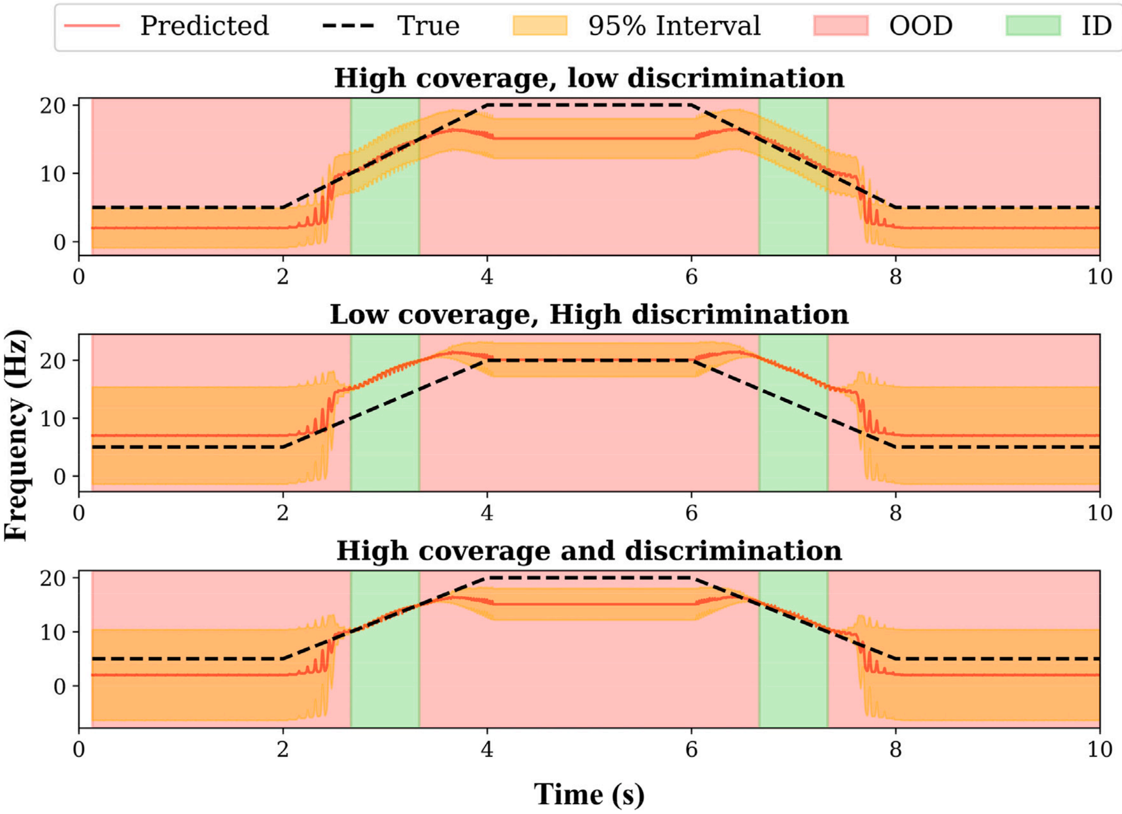

Fig. 6. Visual representations of coverage and discrimination across three different scenarios. The top plot displays intervals that accurately cover the true values The top plot demonstrates high coverage with low discrimination, where the 95% prediction intervals (±2 standard deviations) encompass the true values but fail to discriminate between high- and low-confidence predictions. The middle plot illustrates prediction intervals that fail to encompass the true values, indicating poor coverage, with no accurate predictions in the in-domain segment, but show effective discrimination. The bottom plot shows intervals that both accurately cover the true values and effectively discriminate between high- and low-confidence predictions.

the OOD data, which is characteristic of epistemic uncertainty.

Following this evaluation, we emphasize the importance of coverage and discrimination in probabilistic models. These requirements are crucial for ensuring that uncertainty estimates are both actionable and reliable. Accurate coverage indicates that the model's uncertainty estimates reliably encompass the true values of predictions with high probability, which is crucial in applications where avoiding false negatives is essential. Conversely, accurate discrimination refers to the model's ability to effectively differentiate between high-confidence and low-confidence predictions. Effective discrimination enables the model to identify uncertain instances, guiding further data collection or adjustments. Fig. 6 visually represents these concepts, illustrating how probabilistic models should achieve both accurate coverage and discrimination to ensure reliable predictions.

In addition to evaluating the general performance of the model, we investigate the relationship between uncertainty estimates and prediction error. Using our toy example, we derive uncertainty estimates from the GPR model, which is evaluated on test samples. We then assess whether these estimates are correlated with the actual prediction error, which remains unknown during the model training phase. As illustrated in Fig. 7, we observe that higher uncertainty estimates positively correlate with greater prediction error, indicating that the model's uncertainty estimates can effectively reflect the level of prediction error.

4.2. Algorithm performance

In this section, we utilize the GPR model trained in the previous section and compare its performance with other probabilistic models, such as MC Dropout and NNE. By comparing these models, we aim to assess how well each approach balances accurate prediction with reliable uncertainty estimates. Fig. 8 provides a visual comparison of the predictive performance and UQ for GPR, MC Dropout, and NNE. Each subfigure contrasts the predicted frequency (Hz) over time (s) with the true frequency (black dashed line), displaying the model's prediction (red line) and the corresponding uncertainty interval (yellow shaded region).

All models—GPR, MC Dropout, and NNE—demonstrated effective coverage and discrimination in their predictive uncertainty estimates, though notable differences are observed in their performance when generalizing beyond the training domain. NNE and MC Dropout exhibited superior behavior compared to GPR when predicting frequencies above the training domain, indicating that these

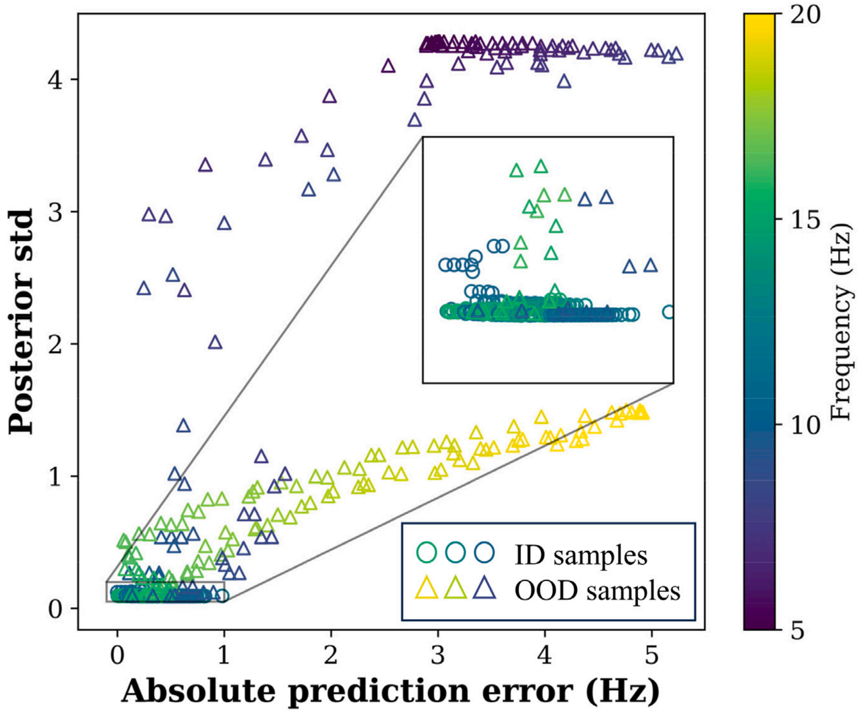

Fig. 7. Scatter plot showing the relationship between prediction error and uncertainty estimate for in-domain (ID) and out-of-domain (OOD) samples. The inset plot highlights a zoomed-in region to visualize better the correlation between prediction error and uncertainty estimate for ID samples in this area.

models are better equipped to handle OOD scenarios. This highlights their strength in managing unseen signals more effectively. However, despite achieving strong coverage and discrimination, all models displayed high prediction error and uncertainty at frequencies below the training domain. This increased uncertainty likely stems from limitations in the feature extraction process, as features extracted in this range fail to capture meaningful patterns, resulting in inaccurate predictions. GPR, while maintaining lower uncertainty when predicting frequencies above the training domain, also followed this trend, with increased uncertainty and error at lower frequencies. Overall, this analysis provides valuable insight into the limitations of both model performance and the feature extraction process, with NNE and MC Dropout proving more robust in handling OOD predictions compared to GPR.

5. Case study 2: ProbPredict-SHM on DROPBEAR testbed

The DROPBEAR experimental testbed, shown in Fig. 9, was developed to study high-rate dynamic systems, with a focus on simulating damage or detachment in such systems [6]. It comprises of a 51 × 6 × 350 mm cantilever beam equipped with a single accelerometer (model 393B04 by PCB Piezotronics) mounted at the beam's free edge. The testbed incorporates a movable roller support system that allows controlled variation in boundary conditions during experiments. By adjusting the roller's position, the system induces repeatable, controllable changes in system dynamics, simulating damage in real-time without requiring permanent alterations to the beam. This ability to simulate dynamic damage during system response, without needing to change configurations between tests, enables a more realistic and controlled study of system behavior under high-rate conditions.

The roller follows a predefined profile ranging from 48 mm to 175 mm, initiating vibrations in the beam. Experimental tests involved various input profiles, designed to elicit specific structural responses. Data acquisition during experiments was conducted using a 14-bit ADC for the linear transducer (SPS-L225-HALS by Honeywell) and a 24-bit IEPE ADC for acceleration data (NI-9234). These measurements provide insights into the dynamic behavior of structures under ballistic environments, facilitating the development and validation of advanced state estimation techniques. The latest dataset used in this work is made available through a public repository [39]. The latest dataset extends the experimental test by running the test with multiple trials and configurations, enabling a more comprehensive analysis of the system's behavior under varied conditions. This updated dataset includes new movement profiles and additional trial repetitions to improve the robustness of the experimental observations. Specifically, the dataset encompasses three primary movement sets: Standard, Stepwise, and Random.

In the **Standard Movement** set, data was collected using six distinct input profiles: square waves, sinusoidal waves, and impulse inputs, each designed to represent common excitation forces applied to the beam. This set includes 20 trials to capture the beam's response under these predictable conditions. The **Random Movement** set consists of 10 different random movement profiles, each with 10 trials, designed to capture the beam's response under unpredictable or less structured conditions, providing a broader view of the system's dynamic behavior.

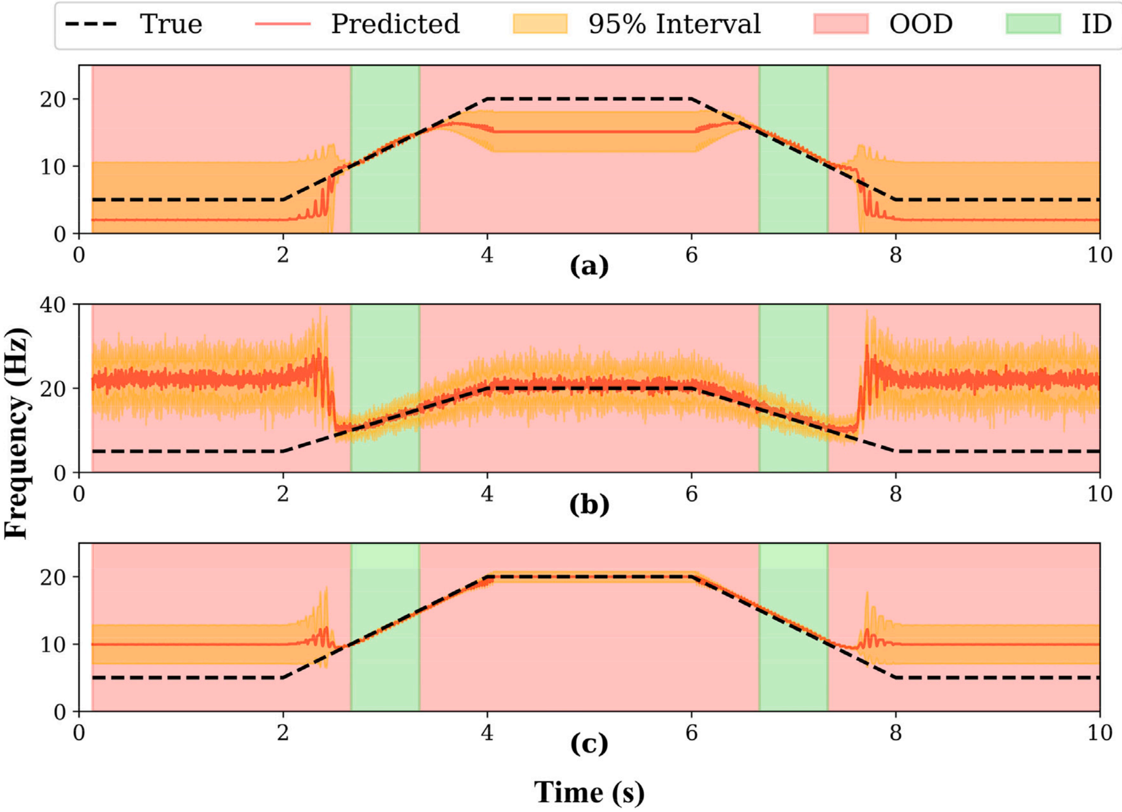

Fig. 8. Prediction results using the ProbPredict-SHM pipeline. From top to bottom: (a) Gaussian process regression (GPR), (b) Monte Carlo (MC) dropout, and (c) Neural network ensemble (NNE). Each subplot shows the predicted frequency (Hz) over time (s) with the true frequency indicated by the black dashed line. The red line represents the model's predictions, and the yellow shaded region denotes the prediction interval (±2 standard deviations). The shaded areas in light green and light coral highlight in-domain (ID) and out-of-domain (OOD) regions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

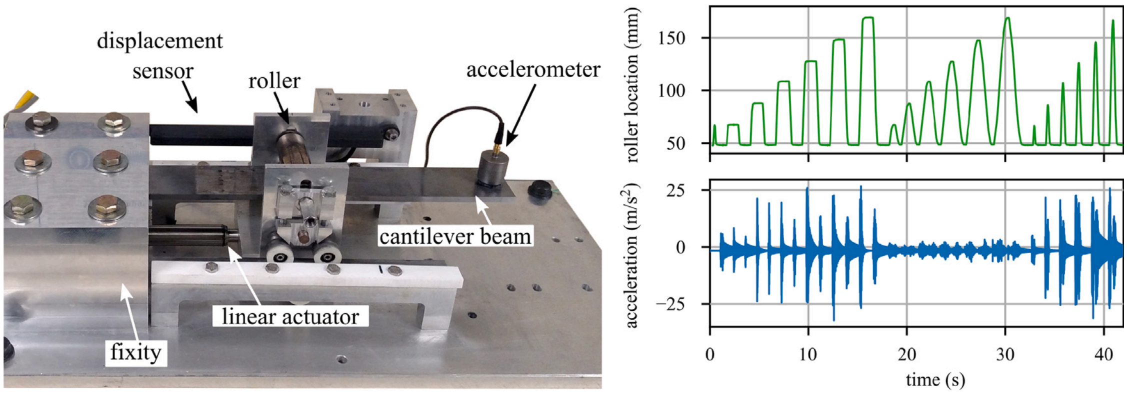

Fig. 9. DROPBEAR experimental setup along with the displacement and acceleration signals [39].

5.1. Algorithm performance

In this section, we implement the proposed pipeline by first training the model on the Standard Movement dataset and then evaluating its generalization ability on the Random Movement dataset. This strategy allows us to assess the model's robustness across diverse input profiles and scenarios, providing insights into its adaptability. Both the training and testing signals are processed using feature extraction techniques, employing the TDA parameters specified in Table 2. These parameters were optimized based on the training frequency range and consistently applied to the testing signal to ensure a fair and comparable evaluation. The pipeline was

Table 2
Parameters for TDA feature extraction for Case Study 2. The time delay τ is selected as 2.976 ms, half of the maximum allowable value of 5.952 ms, computed using Eq. (2), to prevent folding of the topological space.

Parameter	Value (unit)
H_0 Window Size	35.71 (ms)
H_1 Window Size	50.37 (ms)
Time Delay (τ)	2.976 (ms)
Embedding Dimension (d)	3
Window Step Size	40 (steps)

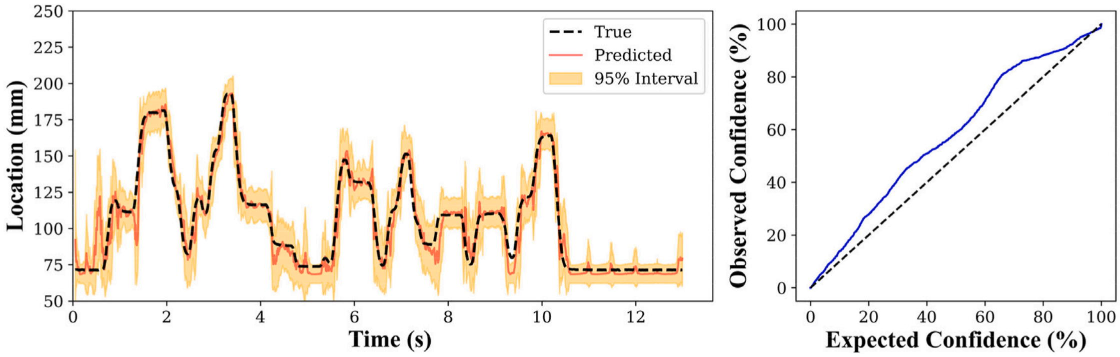

Fig. 10. Prediction results with RNN–NNE for sequence number 5. The left figure visualizes the prediction results using RNN–NNE, while the right figure shows the calibration curve for RNN–NNE.

Table 3
Model performance metrics for sequence number 5. The computation time per sample is computed as the sum of the times consumed by signal preprocessing, TDA-based feature extraction, and model prediction.

Model	MAE (mm)	TRAC	NLL	ECE (%)	Computation time per sample (ms)
NN	6.856	0.991	–	–	26.896
RNN	5.901	0.994	–	–	26.961
GPR	6.508	0.991	−2.339	10.625	27.107
MC Dropout	8.475	0.990	3.282	22.264	26.932
NNE	6.309	0.992	−4.304	13.621	27.220
RNN-MC	7.524	0.992	−2.613	16.001	27.156
RNN–NNE	5.030	0.995	−2.498	7.817	27.158

Table 4
Breakdown of computation time per sample for models on sequence number 5. The computation time per sample is computed as the sum of the times consumed by signal preprocessing (Prep. time), TDA-based feature extraction (TDA time), and model prediction (Pred. time).

Model	Prep. time (ms)	TDA time (ms)	Pred. time (ms)
NN			0.003
RNN			0.068
GPR			0.213
MC Dropout	0.0369	26.856	0.039
NNE			0.326
RNN-MC			0.263
RNN–NNE			0.265

evaluated on a high-performance computing system equipped with (1) a 12th Gen Intel® Core™ i7-12700K 3.60 GHz processor with 12 cores and 20 threads and (2) 32.0 GB RAM.

The results of ProbPredict-SHM with the RNN–NNE model are illustrated in Fig. 10, where sequence number 5 is selected due to its lowest MAE among all models. Table 3 summarizes the algorithm's performance on this sequence. The RNN–NNE model achieves the lowest MAE of 5.030 mm, indicating superior prediction accuracy. This suggests that the combination of RNN and NNE is particularly effective at capturing the underlying patterns in the data, as evidenced by its highest TRAC and the lowest ECE among the probabilistic models, with an ECE of 7.817%. Regarding computation time, while the RNN–NNE model takes 0.175 ms longer to predict compared to the standard RNN, most of the time is spent on TDA-based feature extraction. The TDA time, which consistently accounts for around 26.7 ms across all models, dominates the total computation time. Table 4 breaks down the computation time

Table 5
Average results over Random Movement dataset. The computation time per sample is computed as the sum of the times consumed by signal preprocessing, TDA-based feature extraction, and model prediction.

Model	MAE (mm)	TRAC	NLL	ECE (%)	Computation time per sample (ms)
NN	7.806	0.988	–	–	26.806
RNN	7.171	0.992	–	–	26.865
GPR	7.573	0.988	−2.327	9.120	27.016
MC Dropout	10.074	0.985	7.330	25.263	26.836
NNE	7.346	0.989	−4.202	11.353	27.110
RNN-MC	8.197	0.990	−2.598	15.876	27.043
RNN–NNE	5.705	0.993	−2.481	7.335	27.055

Table 6
Breakdown of average computation time per sample for models over Random Movement dataset. The computation time per sample is computed as the sum of the times consumed by signal preprocessing (Prep. time), TDA-based feature extraction (TDA time), and model prediction (Pred. time).

Model	Prep. time (ms)	TDA time (ms)	Pred. time (ms)
NN			0.0039
RNN			0.0628
GPR			0.2137
MC Dropout	0.0361	26.766	0.0339
NNE			0.3078
RNN-MC			0.2415
RNN–NNE			0.2527

for each model into signal preprocessing (Prep. time), TDA-based feature extraction (TDA time), and model prediction (Pred. time). Nonetheless, this indicates that incorporating a probabilistic model does not significantly increase computation time. All models' prediction durations remain well below 100 ms, ensuring real-time predictions are feasible. Additional figures showing results for other models and sequence number 5 can be found in the Appendix C.

Several key insights emerge from the average results across all Random Movement sequences presented in Table 5. The RNN–NNE model excels with the lowest MAE of 5.705 mm and the highest TRAC, demonstrating superior predictive accuracy and robustness. It also achieves strong probabilistic performance with a low ECE of 7.335%, indicating well-calibrated uncertainty estimates. In contrast, the MC Dropout model, while fast, shows poor calibration with a high ECE of 25.263%. Table 6 provides a breakdown of the average computation time per sample for models tested on the Random Movement dataset. Despite its slightly longer processing time, the RNN–NNE model offers an excellent balance between high accuracy and reliable UQ, making it particularly well-suited for real-time applications where both performance and calibration are crucial.

6. Discussion on practical applicability of ProbPredict-SHM

6.1. Practical applicability of ProbPredict-SHM

The proposed ProbPredict-SHM pipeline advances structural state estimation by offering a sophisticated alternative to traditional methods. Its performance across various datasets, including Standard and Random Movements, demonstrates strong generalization and adaptability to different conditions. A key strength of ProbPredict-SHM is its dual capability to provide state estimates alongside UQ, offering a clear measure of predictive reliability. This feature ensures that predictions are both accurate and trustworthy. With predictions delivered in under 100 ms, ProbPredict-SHM is ideal for real-time applications such as industrial monitoring and autonomous systems, where rapid and reliable state estimation is essential.

Looking ahead, several avenues could enhance ProbPredict-SHM's applicability. Expanding the pipeline to include additional metrics for predictive uncertainty could provide deeper insights into prediction reliability, particularly in complex scenarios. Applying ProbPredict-SHM to diverse high-rate datasets would also test its robustness across various domains. Furthermore, incorporating a proxy for error and anomaly detection could significantly enhance the pipeline's functionality. A proxy for error refers to a surrogate measure that approximates prediction errors when the true errors are not directly available. This approach helps assess model performance and reliability in real-time applications. Anomaly detection would enable ProbPredict-SHM to identify and respond to unexpected data deviations, improving its ability to handle irregularities.

Fig. 11 illustrates both use cases. The proxy for error is demonstrated by comparing predicted uncertainty with actual deviations in the signal. It shows that regions with high uncertainty also exhibit high absolute error. Meanwhile, the anomaly detection mechanism identifies significant outliers or abnormal patterns in the data stream. This combination provides real-time feedback, facilitating more adaptive and accurate predictions. Overall, ProbPredict-SHM's strengths and potential for future enhancements make it a powerful tool for real-time state estimation in the high-rate domain.

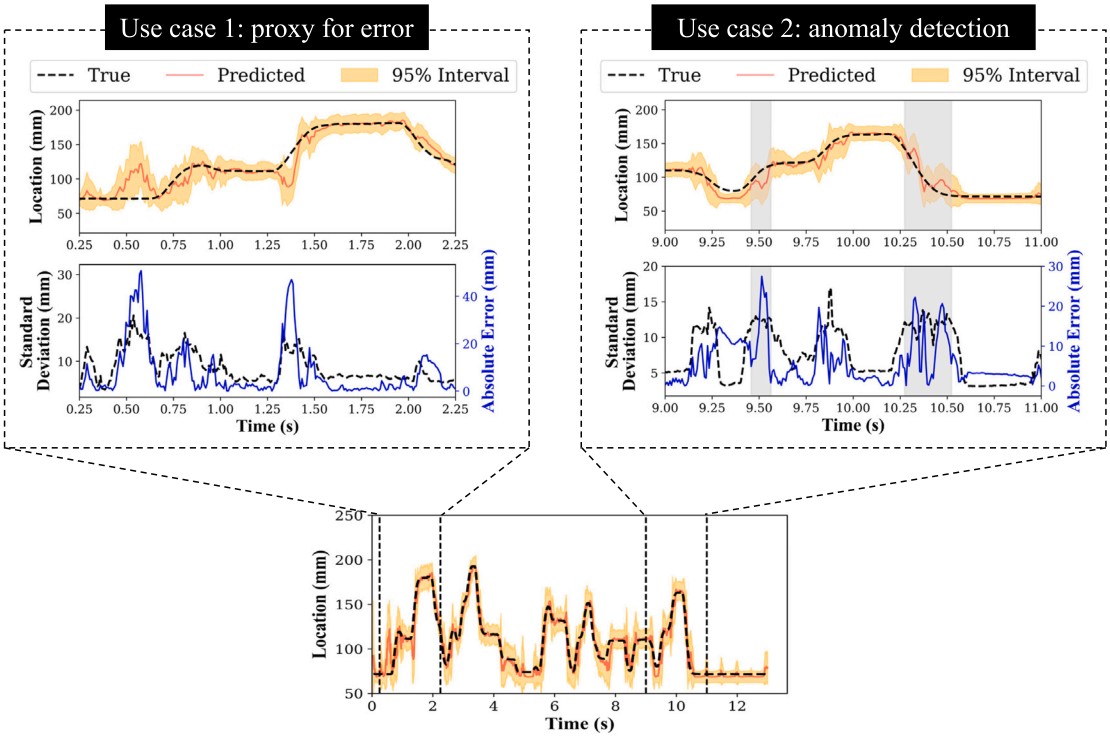

Fig. 11. Two potential use cases of ProbPredict-SHM. The top left figure illustrates the proxy for error within the pipeline by comparing predicted uncertainty with actual deviations, showing that regions of high uncertainty correspond to high absolute error. The top right figure demonstrates anomaly detection within the pipeline by identifying significant outliers or abnormal patterns in the data stream.

6.2. Limitations and potential opportunities

Despite achieving prediction times of less than 100 ms, the current implementation of ProbPredict-SHM faces several limitations. The primary bottleneck is the TDA feature extraction process, which restricts further reductions in computational time. Moreover, the purely data-driven nature of the pipeline results in a lack of interpretability for the extracted topological features, as they do not have a direct connection to the physical system behavior. Additionally, the feature extraction process relies on fixed window sizes for H_0 and H_1. If new incoming signals fall outside these predetermined bounds, prediction accuracy may degrade.

To address these limitations, several opportunities for enhancement can be pursued. One key opportunity is to develop a fast TDA approximation method that bypasses full persistent homology calculations, reducing computational costs while still retaining essential topological features. Integrating hybrid physics-informed approaches alongside TDA features could also improve the interpretability of the predictions, aligning them more closely with the underlying physical system dynamics. Finally, implementing adaptive learning techniques would enable the pipeline to update on-the-fly, capturing new incoming signals that fall outside the established bounds, and ensuring that the system remains accurate and adaptable in dynamic real-time environments.

7. Conclusion

The two widely adopted modeling techniques for high-rate structural health monitoring are physics-based and data-based. However, when deployed online, these models often lack prediction confidence, providing only the predicted results without an accompanying measure of certainty. This paper introduces an alternative, machine learning-based solution called the **P**robabilistic **P**rediction for **S**tructural **H**ealth **M**onitoring (ProbPredict-SHM). The pipeline offers several advantages over traditional approaches, including the ability to quantify uncertainty in predictions. This dual capability ensures more reliable decision-making in high-rate structural health monitoring, where understanding the uncertainty of predictions is crucial for safety and performance.

The paper explores four algorithms: Gaussian process regression (GPR), Monte Carlo (MC) dropout, neural network ensembles (NNE), and recurrent neural networks (RNN) with probabilistic capabilities. Each algorithm is evaluated based on error metrics and its ability to quantify uncertainty. The results indicate that the RNN–NNE model stands out with the lowest MAE of 5.705 mm, the highest TRAC, and the lowest ECE of 7.335%. These metrics underscore the RNN–NNE model's superior predictive accuracy and robustness, particularly in its effective UQ compared to the other models.

Future work will focus on automating the model selection process to enhance the pipeline's usability. By evaluating multiple models during training based on metrics like MAE, TRAC, NLL, and ECE, the pipeline will automatically identify and deploy the best-performing model, ensuring efficient and reliable real-world applications. Overall, our work underscores the value of integrating machine learning tools for high-rate structural health monitoring. Despite achieving low errors, we also emphasize that any algorithm must undergo UQ checks before deployment in the field. Finally, we demonstrate how the use of machine learning pipelines can provide a computationally efficient and accurate solution for structural state estimation. We envision machine learning pipelines becoming a standard technique in the field of structural health monitoring, driving advancements in both accuracy and reliability across a wide range of high-rate dynamic systems.

CRediT authorship contribution statement

Yang Kang Chua: Writing – original draft, Software, Methodology, Investigation, Formal analysis. **Daniel Coble:** Writing – review & editing, Software, Methodology, Investigation, Formal analysis. **Arman Razmarashooli:** Writing – review & editing, Methodology, Investigation. **Steve Paul:** Writing – review & editing, Methodology. **Daniel A. Salazar Martinez:** Writing – review & editing, Methodology. **Chao Hu:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Conceptualization. **Austin R.J. Downey:** Writing – review & editing, Supervision, Project administration, Conceptualization. **Simon Laflamme:** Writing – review & editing, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to acknowledge the financial support from the Defense Established Program to Stimulate Competitive Research (DEPSCoR) award number FA9550-22-1-0303, the Air Force Office of Scientific Research (AFOSR) award numbers FA9550-23-1-0033 and FA9550-21-1-0083, and the National Science Foundation, United States awards numbers CCF-1937460, CCF-2234919, and CPS-2237696. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the sponsors' views.

Appendix A. TDA features

This section describes the features extracted using TDA. We will first provide an overview of the TDA process and then explain each feature extracted through TDA.

A.1. Introduction to TDA

TDA is a powerful tool that analyzes data using techniques from algebraic topology [40]. TDA methods are especially useful for capturing complex patterns that are not easily identified through traditional statistical approaches. One of these approaches is called persistent homology [41].

The TDA process begins with data preprocessing to ensure its suitability for topological analysis. This preparation may involve converting the data into a point cloud. Next, a simplicial complex (e.g., Vietoris–Rips or Čech complex) is constructed to represent the data's structure [42]. Persistence diagrams (PD) or barcodes are then computed to encapsulate the topological features across various scales [43]. Points in a PD are organized by topological feature dimensions, each dimension i containing its own set of points. Specifically, PD_i denotes the set of points for the ith feature group. Points in PD_i are denoted as $(b_{j,i}, d_{j,i})$ for $j = 1, 2, \ldots, n_i$, where $b_{j,i}$ represents the birth time and $d_{j,i}$ represents the death time of the jth point within the ith feature group. Essential features are extracted from these persistence diagrams for further analysis. Additional details on TDA can be found in [44].

A.2. Extracted features

The features extracted using TDA provide valuable insights into the underlying structure of the data. Implemented in tools such as giotto-tda [45], these features include Betti numbers, persistence entropy, and landscape functions, which facilitate detailed characterization of the data's topological properties.

Maximum persistence. Persistence measures the duration of a topological feature, calculated as the difference between its death and birth times. For the ith topological feature group, the maximum persistence is:

$$p_{\max,i} = \max_j (d_{j,i} - b_{j,i}) \tag{A.1}$$

where $d_{j,i}$ and $b_{j,i}$ denote the death and birth times of the jth point within the ith feature. The feature can identify significant and stable topological features in the data [46].

Wasserstein amplitude. The Wasserstein distance captures overall variation in a persistence diagram, unlike the bottleneck distance [47]. For a persistence diagram P compared with the trivial diagram Q, the Wasserstein amplitude is given by:

$$\|P\|_{W,i} = \frac{1}{2}\left(\sum_j \left(d_{j,i} - b_{j,i}\right)^p\right)^{1/p} \tag{A.2}$$

where p is the norm used (typically $p = 2$ for the Euclidean norm), and $\frac{1}{2}\left(d_{j,i} - b_{j,i}\right)$ represents the height of features in P.

Landscape amplitude. The persistence landscape maps persistence diagrams into a vectorized function space, facilitating analysis and comparison with statistical and machine learning methods [48]. It is represented as a sequence of functions $\{\lambda_k\}_{k\in\mathbb{N}}$, where each $\lambda_k(t)$ is the kth largest value of $\{\Lambda_i(t)\}_{i\in I}$, defined by:

$$\Lambda_i(t_m) = \max\left(\min_j(t_m - b_{j,i}, d_{j,i} - t_m), 0\right) \tag{A.3}$$

Here, t_m denotes evenly spaced sample points ranging from $t_0 = b_{\min}$ to $t_{M-1} = d_{\max}$, with m from 0 to $M-1$. The values $b_{\min}$ and $d_{\max}$ are the minimum birth time and maximum death time across all persistence diagrams, respectively. To compute the landscape amplitude for comparison with the trivial diagram Q, we use:

$$\|P\|_{Ls,i} = \left(\sum_{k=1}^{N}\sum_{m=0}^{M-1}(\lambda_k(t_m))^p\right)^{1/p} \cdot \Delta t^{1/p} \tag{A.4}$$

where Δt is the spacing between sample points t_m. We use $N = 1$, $M = 100$, and $p = 2$ to represent the Euclidean norm. For cases beyond the finite setting, refer to [44].

Betti amplitude. The Betti curve, or persistence indicator function (PIF), measures the topological activity of a persistence diagram (PD) by counting features at various threshold levels [49]. The Betti curve $\beta_i(t_m)$ is defined as:

$$\beta_i(t_m) = \sum_j \mathbb{I}(b_{j,i} \le t_m < d_{j,i}) \tag{A.5}$$

where $\mathbb{I}$ is the indicator function. To compute the Betti amplitude for comparison with the trivial diagram Q, we use:

$$\|P\|_{Bt,i} = \left(\sum_{m=0}^{M-1}(\beta_i(t_m))^p\right)^{1/p} \cdot \Delta t^{1/p} \tag{A.6}$$

where $M = 100$ and $p = 2$, representing the Euclidean norm. The spacing between sample points t_m is Δt.

Silhouette amplitude. The persistence silhouette highlights significant topological features from a persistence diagram, aiding data analysis and interpretation [50]. It is computed as a weighted average of the landscape functions:

$$\phi_i(t_m) = \frac{\sum_{j\in I} w_{j,i}\Lambda_i(t_m)}{\sum_{j\in I} w_{j,i}} \tag{A.7}$$

where $w_{j,i} = |d_{j,i} - b_{j,i}|^p$. For $0 < p \le \infty$, ϕ is referred to as the *p-power-weighted silhouette* of P. To compute the silhouette amplitude for comparison with the trivial diagram Q, we use:

$$\|P\|_{Sil,i} = \left(\sum_{m=0}^{M-1}(\phi_i(t_m))^p\right)^{1/p} \cdot \Delta t^{1/p} \tag{A.8}$$

with $M = 100$ and $p = 2$, representing the Euclidean norm.

Appendix B. Hyperparameter tuning

In this section, we describe the hyperparameter tuning process for each model utilized in the ProbPredict-SHM pipeline. We outline the specific hyperparameters adjusted, the methodologies used for tuning, and the results obtained from the tuning process. This includes a discussion of the grid search approach for optimizing model settings, the evaluation criteria for selecting the best parameters, and the impact of these hyperparameters on model performance and reliability.

B.1. Hyperparameter tuning for Gaussian process regression

In this section, we detail the hyperparameter tuning process for the Gaussian process regression (GPR) model, focusing on the kernel function selection, the noise level estimation, and the optimization of hyperparameters.

Table B.7
Hyperparameter tuning results for different GPR kernels, including the Mean Absolute Error (MAE) for each kernel.

Kernel type	σ_f	Length scale	Noise level	MAE (mm)
RBF	0.005154	1.774090	8.83×10^{-5}	6.9595
Matérn 1.5	0.003215	1.230911	1.45×10^{-5}	7.0650
Matérn 2.5	0.003695	1.967375	6.75×10^{-5}	6.7461

Kernel selection. The first step in tuning the GPR model involves selecting an appropriate kernel function. We considered three kernels for this purpose:

- **Radial basis function (RBF) kernel**: The RBF kernel, also known as the Gaussian kernel, is defined as:
$$k(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{\|x_i - x_j\|^2}{\ell^2}\right) \tag{B.1}$$
Here, σ_f^2 is a constant multiplier that sets the upper limit of the prior variance and covariance, while ℓ is the length scale parameter that controls the smoothness of the function.
- **Matérn kernel**: The Matérn kernel is given by:
$$k(x_i, x_j) = \sigma_f^2 \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j)\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j)\right) \tag{B.2}$$
where $\Gamma(\cdot)$ is the Gamma function, $d(x_i, x_j)$ is the Euclidean distance between points x_i and x_j, and K_ν is the modified Bessel function of the second kind and order ν [29]. A larger value of ν results in a smoother approximated function. We will test this kernel with $\nu = 1.5$ and $\nu = 2.5$.

For our experiments, we set $\sigma_f^2 = 1.0$ and $\ell = 1.0$, with both parameters being optimized automatically by the GPR algorithm using the default settings in the `scikit-learn` library.

Noise estimation with white kernel. To accurately estimate the noise level in the data, we introduced a white kernel into the GPR model. The white kernel allows the model to account for measurement noise, essential for preventing overfitting and improving the model's generalization ability. The white kernel is combined with the chosen kernel as follows:

$$k(x_i, x_j) = k_{\text{chosen}}(x_i, x_j) + \sigma_n^2 \delta(x_i, x_j) \tag{B.3}$$

where k_{chosen} represents the selected kernel function (RBF or Matern), σ_n^2 is the noise variance, and $\delta(x_i, x_j)$ is the Kronecker delta function.

To find the optimal noise variance σ_n^2, we performed a grid search over a range of values: $\sigma_n^2 \in \{1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}, 1.0, 10.0, 100.0\}$. The grid search is conducted using a 5-fold cross-validation approach, with the performance of each parameter combination assessed using the negative mean squared error (MSE) as the scoring metric. Choosing specific noise values helps the model find the optimal noise value by the GPR algorithm in the `scikit-learn` library for each kernel. We will use the estimated noise value through this process as the noise variance in the GPR algorithm.

Results of hyperparameter tuning for GPR. Using the optimized parameters for each kernel identified through the grid search, we validated the results with unseen data. We used Mean Absolute Error (MAE) as the scoring metric to determine the best kernel for the GPR model. Table B.7 summarizes the results of our hyperparameter tuning. Among the tested kernels, Matérn 2.5 demonstrated the best performance with the lowest MAE value of 6.75 mm and a well-calibrated noise level of 6.75×10^{-5}.

B.2. Hyperparameter tuning for neural network ensemble (NNE) and Monte Carlo (MC) dropout

This section details the hyperparameter tuning for NNE and MC Dropout, focusing on the number of hidden units, layers, and ensemble size.

Number of layers and hidden units. For both NNE and MC Dropout, we began by fine-tuning the number of layers and hidden units for the baseline neural network (NN). We experimented with hidden unit sizes [8, 16, 32, 64, 128] and layer counts from 1 to 7, evaluating each configuration over 10 runs using the ReLU activation function for all hidden layers. Models are trained with early stopping to prevent overfitting, using a patience of 15 epochs. The Adam optimizer with a mean squared error loss function is employed, and the neural network is trained for up to 1000 epochs. MAE is computed on the validation data to assess performance. The results are aggregated to determine the optimal configuration of layers and hidden units, identifying that 5 hidden layers and 64 hidden units provided the best trade-off between model complexity and performance, as shown in Fig. B.12. This configuration minimized unnecessary complexity while maintaining strong performance.

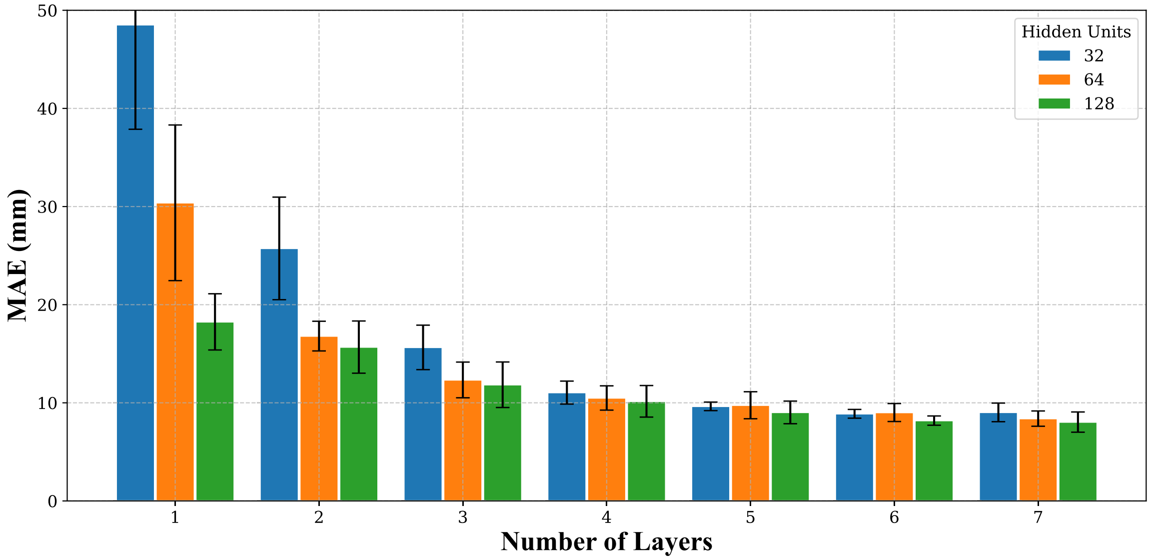

Fig. B.12. Tuning results for the number of layers and hidden units in the neural network. The error bars reflect the run-to-run variation. Results for hidden units 8 and 16 are not shown as they consistently produced higher errors than the other configurations.

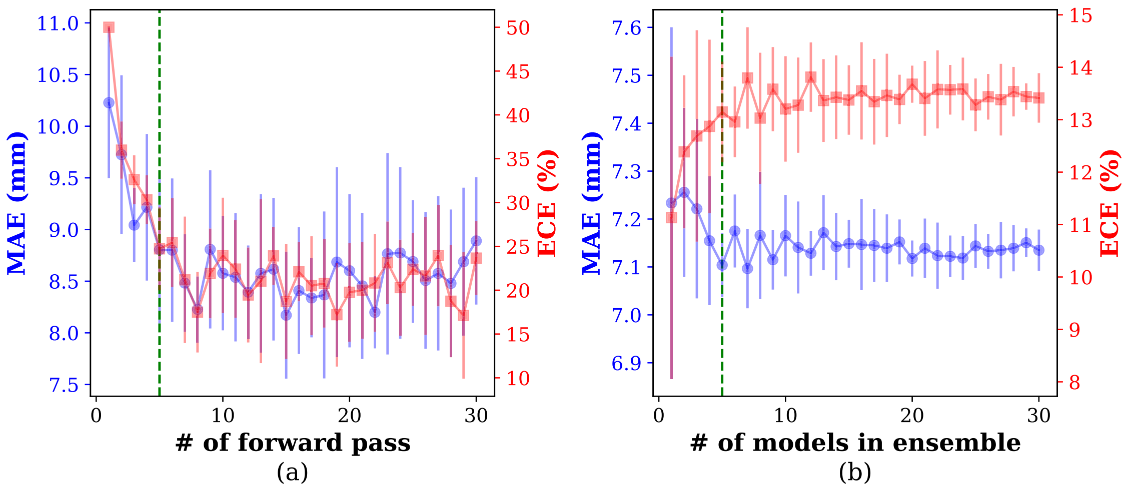

Fig. B.13. Tuning results for the (a) MC Dropout and (b) Neural Network Ensemble (NNE). The selected ensemble size for this case study is determined by the green vertical line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Forward pass for MC dropout. For MC Dropout tuning, we used one trial's data for training and another for validation, iterating through five model initializations. Each model is trained on the training data, and the model with the lowest MSE is selected. Uncertainty estimates are evaluated by performing multiple forward passes through the best model, capturing predictions for ensemble sizes from 1 to the maximum. For each ensemble size, we computed the mean and standard deviation of random subsets of predictions, with accuracy and calibration assessed using MAE and expected calibration error (ECE). This process is repeated 10 times per dataset to ensure robust evaluation, and the results—including MAE and ECE for various ensemble sizes are saved for further analysis and final model evaluation.

Ensemble size for neural network ensemble (NNE). For NNE tuning, one trial's data is used for training and another for validation. We trained 80 NNE models, tracking progress and calculating each model's MSE. The 60 models with the lowest MSEs are selected for further evaluation. We assessed ensemble sizes ranging from 1 to the maximum over 10 runs. For each size, we computed the mean and variance of predictions from random subsets of selected models and evaluated accuracy and calibration using MAE and ECE. Results for different ensemble sizes are aggregated and stored for each dataset type. This evaluation determined the optimal ensemble size and fine-tuned the hyperparameters effectively.

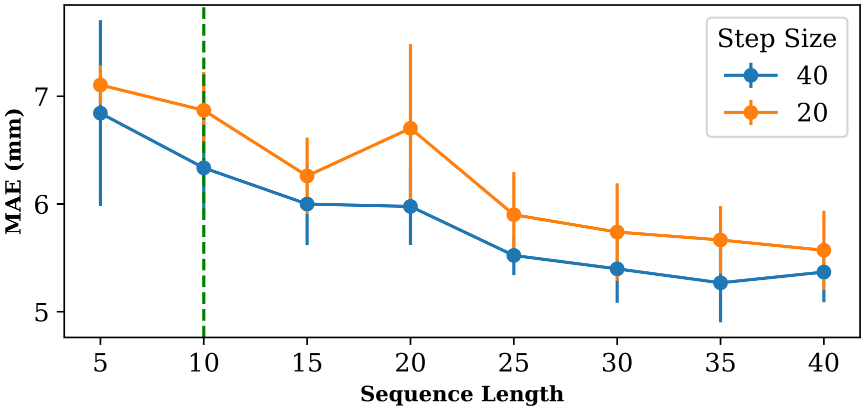

Fig. B.14. Tuning results for different segment lengths in RNN models. The plot shows the mean absolute error (MAE) for various segment lengths, with error bars representing the standard deviation across 10 runs. The green vertical line indicates the selected sequence length.

Results of hyperparameter tuning for MC dropout and NNE. Fig. B.13 presents the results of hyperparameter tuning for the number of forward passes and the number of ensembles for MC Dropout and NNE. MAE and ECE is chosen to balance accuracy and UQ capabilities. The error bars represent the variation across different runs. Based on this study, an ensemble size of 5 is selected for both MC Dropout and the neural network ensemble.

B.3. Hyperparameter tuning for RNN

Hyperparameter tuning plays a crucial role in optimizing the performance of RNNs. In this study, we maintain consistency by using the same number of layers and hidden units as selected for the neural network (NN) models. However, we refine the tuning process by focusing on the sequence length and adjusting it to optimize the model's predictive accuracy and performance. This approach allows for targeted fine-tuning while preserving the overall architecture, ensuring a balance between model complexity and computational efficiency.

This tuning method evaluates the RNN using different sequence lengths through a sliding window approach on time series data. The training and validation sets are transformed into sliding windows for each segment length to capture temporal patterns. The model is trained and evaluated on these windows. Ten runs are performed for each sequence length to ensure robust results, and the MAE is calculated for each run. The final results, including the mean and standard deviation of the MAE across different sequence lengths, are plotted to provide insights into the optimal sequence length for this task. Fig. B.14 illustrates the results of this tuning process, highlighting the performance across various sequence lengths. The process is also repeated with a smaller sliding window step size for further analysis.

Results of hyperparameter tuning for RNN. As shown in Fig. B.14, the model's performance improves with increasing sequence length, as indicated by the reduction in MAE. This trend suggests that longer sequence lengths allow the RNN to better capture the temporal dependencies in the time series data, leading to more accurate predictions. However, it is important to note that increasing the sequence length also increases computational cost and may lead to diminishing returns in predictive accuracy beyond a certain point.

The tuning process also reveals that the error bars, representing the standard deviation of MAE across 10 runs, tend to shrink as the sequence length increases. This reduction in variability indicates that the model becomes more stable and less sensitive to initializations with longer sequence lengths. The green vertical line in Fig. B.14 highlights the selected optimal sequence length, which strikes a balance between performance and computational efficiency.

In addition to adjusting the sequence length, we also evaluated the model's performance using a smaller sliding window step size. While reducing the step size provides finer granularity in capturing the temporal dynamics, it led to an increase in errors. This increase occurs because smaller steps fail to introduce significant new information, resulting in redundancy in the model's input. Consequently, the past values remain too similar, offering little improvement in predictive power. Despite the more frequent updates to the input data, the marginal improvement does not outweigh the higher computational cost, highlighting the trade-off between step size and model performance.

Overall, this hyperparameter tuning process for the RNN models demonstrated the importance of optimizing sequence length to enhance predictive accuracy while maintaining manageable computational demands.

Appendix C. Additional figures for sequence number 5

In this section, we present additional figures related to sequence number 5. These figures provide further insights into the performance and behavior of our models under various conditions and parameter settings. Each figure is designed to complement the main results presented in the paper by offering detailed visualizations that reveal deeper aspects of the data and model performance.

See Figs. C.15–C.20.

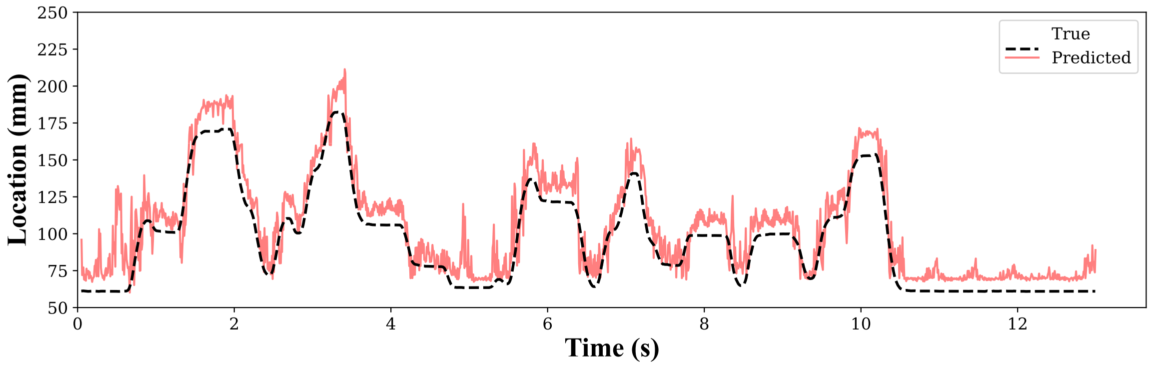

Fig. C.15. Prediction results produced by NN for sequence number 5.

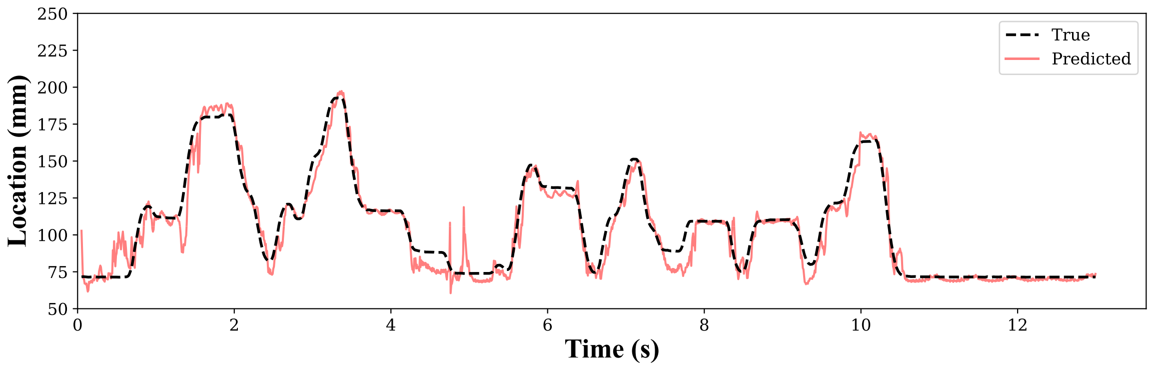

Fig. C.16. Prediction results produced by RNN for sequence number 5.

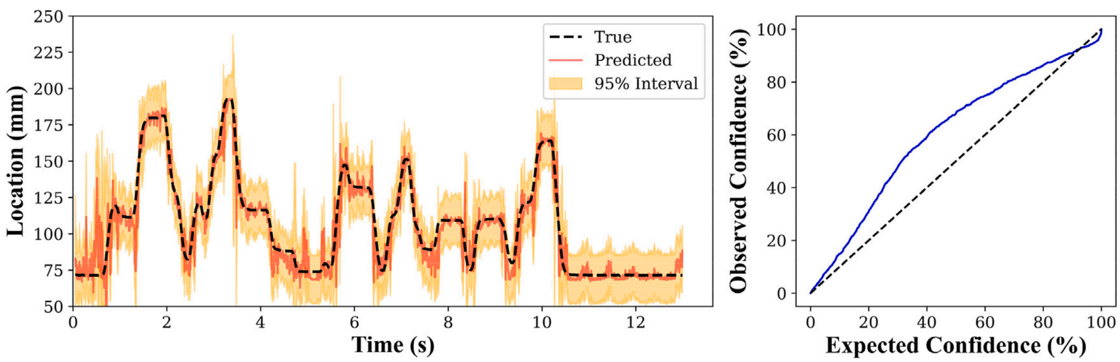

Fig. C.17. Prediction results produced by GPR for sequence number 5.

Data availability

The DROPBEAR testbed dataset used in this work has already been open sourced by Dr. Austin Downey's research group at the University of South Carolina. This dataset can be accessed at the following URL: https://github.com/High-Rate-SHM-Working-Group/Dataset-8-DROPBEAR-Acceleration-vs-Roller-Displacement.

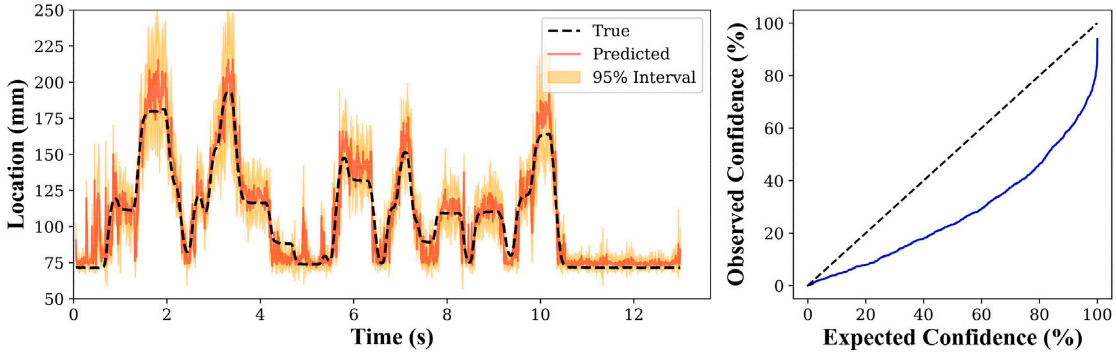

Fig. C.18. Prediction result produced by MC dropout for sequence number 5.

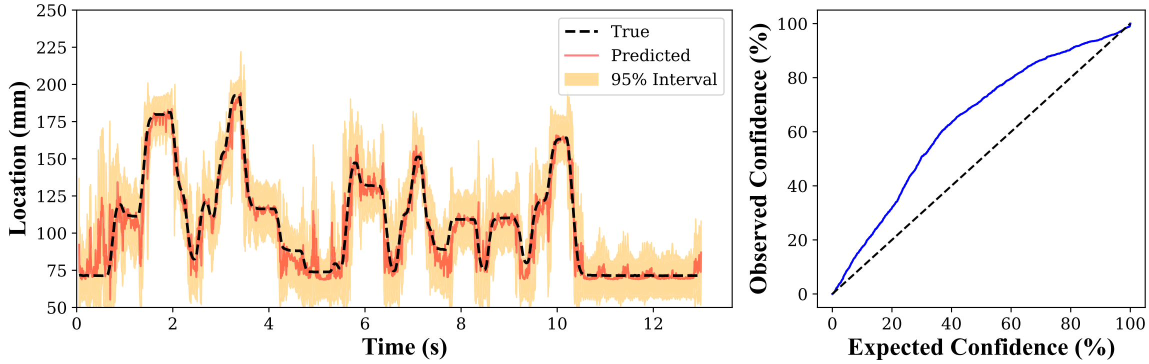

Fig. C.19. Prediction results produced by NNE for sequence number 5.

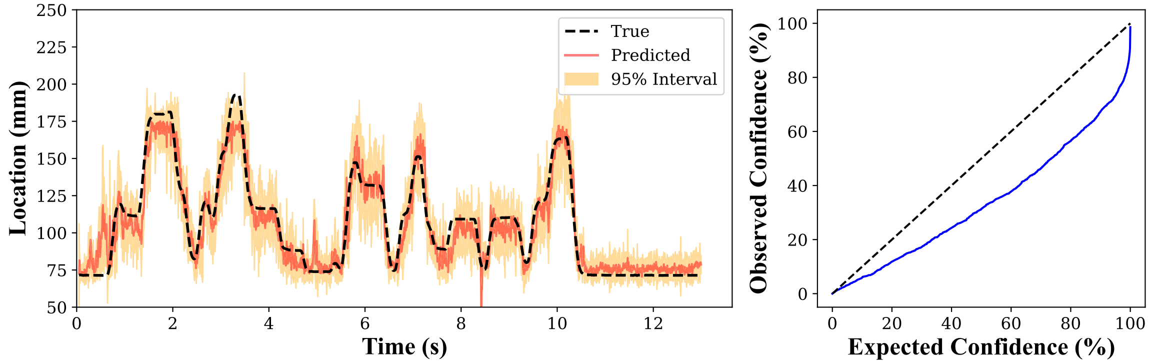

Fig. C.20. Prediction result with RNN combined with Monte Carlo Dropout (RNN MC) for sequence number 5.

References

[1] J. Dodson, A. Downey, S. Laflamme, M.D. Todd, A.G. Moura, Y. Wang, Z. Mao, P. Avitabile, E. Blasch, High-rate structural health monitoring and prognostics: An overview, in: R. Madarshahian, F. Hemez (Eds.), in: Data Science in Engineering, vol. 9, Springer International Publishing, Cham, 2022, pp. 213–217.

[2] C. Stein, R. Roybal, P. Tlomak, W. Wilson, A review of hypervelocity debris testing at the air force research laboratory, Space Debris 2 (4) (2000) 331–356, http://dx.doi.org/10.1023/b:sdeb.0000030024.23336.f5.

[3] R.P. Hallion, C.M. Bedke, M.V. Schanz, Hypersonic Weapons and US National Security: A 21st Century Breakthrough, Mitchell Institute for Aerospace Studies, 2016.

[4] H. Wadley, K. Dharmasena, M. He, R. McMeeking, A. Evans, T. Bui-Thanh, R. Radovitzky, An active concept for limiting injuries caused by air blasts, Int. J. Impact Eng. 37 (3) (2010) 317–323, http://dx.doi.org/10.1016/j.ijimpeng.2009.06.006.

[5] J. Hong, S. Laflamme, J. Dodson, B. Joyce, Introduction to state estimation of high-rate system dynamics, Sensors 18 (2) (2018) 217, http://dx.doi.org/10.3390/s18010217.

[6] B. Joyce, J. Dodson, S. Laflamme, J. Hong, An experimental test bed for developing high-rate structural health monitoring methods, Shock Vib. 2018 (2018) 1–10, http://dx.doi.org/10.1155/2018/3827463.
[7] A. Downey, J. Hong, J. Dodson, M. Carroll, J. Scheppegrell, Millisecond model updating for structures experiencing unmodeled high-rate dynamic events, Mech. Syst. Signal Process. 138 (2020) 106551, http://dx.doi.org/10.1016/j.ymssp.2019.106551.
[8] J. Yan, S. Laflamme, J. Hong, J. Dodson, Online parameter estimation under non-persistent excitations for high-rate dynamic systems, Mech. Syst. Signal Process. 161 (2021) 107960.
[9] E.A. Ogunniyi, C. Drnek, S.H. Hong, A.R. Downey, Y. Wang, J.D. Bakos, P. Avitabile, J. Dodson, Real-time structural model updating using local eigenvalue modification procedure for applications in high-rate dynamic events, Mech. Syst. Signal Process. 195 (2023) 110318.
[10] T. Gowdridge, N. Dervilis, K. Worden, On topological data analysis for structural dynamics: an introduction to persistent homology, ASME Open J. Eng. 1 (2022).
[11] Y. Umeda, Time series classification via topological data analysis, Inf. Media Technol. 12 (2017) 228–239.
[12] M. Byers, L.B. Hinkle, V. Metsis, Topological data analysis of time-series as an input embedding for deep learning models, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2022, pp. 402–413.
[13] N. Ravishanker, R. Chen, Topological Data Analysis (TDA) for time series, 2019, arXiv preprint arXiv:1909.10604.
[14] A. Razmarashooli, Y.K. Chua, V. Barzegar, D. Salazar, S. Laflamme, C. Hu, A.R. Downey, J. Dodson, P.T. Schrader, Real-time state estimation of nonstationary systems through dominant fundamental frequency using topological data analysis features, Mech. Syst. Signal Process. 224 (2025) 112048, http://dx.doi.org/10.1016/j.ymssp.2024.112048.
[15] V. Barzegar, S. Laflamme, C. Hu, J. Dodson, Ensemble of recurrent neural networks with long short-term memory cells for high-rate structural health monitoring, Mech. Syst. Signal Process. 164 (2022) 108201, http://dx.doi.org/10.1016/j.ymssp.2021.108201.
[16] D. Coble, J. Satme, E. Kabir, A. Downey, J. Bakos, D. Andrews, M. Huang, A. Moura, J. Dodson, Towards online structural state-estimation with sub-millisecond latency, 2023.
[17] A. Razmarashooli, D.A.S. Martinez, Y.K. Chua, S. Laflamme, C. Hu, Real-time state estimation using recurrent neural network and topological data analysis, in: Nondestructive Characterization and Monitoring of Advanced Materials, Aerospace, Civil Infrastructure, and Transportation XVIII, vol. 12950, SPIE, 2024, pp. 106–114.
[18] V. Nemani, L. Biggio, X. Huan, Z. Hu, O. Fink, A. Tran, Y. Wang, X. Zhang, C. Hu, Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial, 2023, http://dx.doi.org/10.48550/ARXIV.2305.04933, arXiv.
[19] B.T. Phan, Bayesian Deep Learning and Uncertainty in Computer Vision (Master's thesis), University of Waterloo, 2019.
[20] H. Jiang, B. Kim, M. Guan, M. Gupta, To trust or not to trust a classifier, Adv. Neural Inf. Process. Syst. 31 (2018).
[21] J. Jiménez-Luna, F. Grisoni, G. Schneider, Drug discovery with explainable artificial intelligence, Nat. Mach. Intell. 2 (10) (2020) 573–584.
[22] S. Guo, H. Ding, Y. Li, H. Feng, X. Xiong, Z. Su, W. Feng, A hierarchical deep convolutional regression framework with sensor network fail-safe adaptation for acoustic-emission-based structural health monitoring, Mech. Syst. Signal Process. 181 (2022) 109508.
[23] S. Khan, T. Yairi, A review on the application of deep learning in system health management, Mech. Syst. Signal Process. 107 (2018) 241–265.
[24] A. Thelen, X. Zhang, O. Fink, Y. Lu, S. Ghosh, B.D. Youn, M.D. Todd, S. Mahadevan, C. Hu, Z. Hu, A comprehensive review of digital twin—part 1: modeling and twinning enabling technologies, Struct. Multidiscip. Optim. 65 (12) (2022) 354.
[25] E.R. Deyle, G. Sugihara, Generalized theorems for nonlinear state space reconstruction, PLoS One 6 (3) (2011) e18295.
[26] A.M. Fraser, H.L. Swinney, Independent coordinates for strange attractors from mutual information, Phys. Rev. A 33 (2) (1986) 1134.
[27] M.B. Kennel, R. Brown, H.D.I. Abarbanel, Determining embedding dimension for phase-space reconstruction using a geometrical construction, Phys. Rev. A 45 (6) (1992) 3403–3411, http://dx.doi.org/10.1103/physreva.45.3403.
[28] H. Edelsbrunner, J. Harer, et al., Persistent homology-a survey, Contemp. Math. 453 (26) (2008) 257–282.
[29] C.E. Rasmussen, Gaussian processes in machine learning, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 63–71, http://dx.doi.org/10.1007/978-3-540-28650-9_4.
[30] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, 2016, http://dx.doi.org/10.48550/ARXIV.1612.01474, arXiv.
[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958, URL http://jmlr.org/papers/v15/srivastava14a.html.
[32] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, 2015, http://dx.doi.org/10.48550/ARXIV.1506.02142, arXiv.
[33] Y. Gal, Z. Ghahramani, A theoretically grounded application of dropout in recurrent neural networks, 2015, http://dx.doi.org/10.48550/ARXIV.1512.05287, arXiv.
[34] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with Bernoulli approximate variational inference, 2015, http://dx.doi.org/10.48550/ARXIV.1506.02158, arXiv.
[35] M.I. Jordan, Serial order: A parallel distributed processing approach, in: Advances in Psychology, vol. 121, Elsevier, 1997, pp. 471–495.
[36] P. Avitabile, P. Pingle, Prediction of full field dynamic strain from limited sets of measured data, Shock Vib. 19 (5) (2012) 765–785, http://dx.doi.org/10.1155/2012/408919.
[37] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2, Springer, 2009.
[38] V. Kuleshov, N. Fenner, S. Ermon, Accurate uncertainties for deep learning using calibrated regression, 2018, http://dx.doi.org/10.48550/ARXIV.1807.00263, arXiv.
[39] A. Vereen, A. Downey, J. Dodson, A.G. Moura, Dataset-8-DROPBEAR-acceleration-vs-roller-displacement, 2023, URL https://github.com/High-Rate-SHM-Working-Group/Dataset-8-DROPBEAR-Acceleration-vs-Roller-Displacement.
[40] A. Zomorodian, Topological data analysis, Adv. Appl. Comput. Topol. 70 (2012) 1–39.
[41] N. Otter, M.A. Porter, U. Tillmann, P. Grindrod, H.A. Harrington, A roadmap for the computation of persistent homology, EPJ Data Sci. 6 (2017) 1–38.
[42] G. Carlsson, M. Vejdemo-Johansson, Topological Data Analysis with Applications, Cambridge University Press, 2021.
[43] H. Edelsbrunner, D. Letscher, A. Zomorodian, Topological persistence and simplification, Discrete Comput. Geom. 28 (2002) 511–533.
[44] P.T. Schrader, Topological multimodal sensor data analytics for target recognition and information exploitation in contested environments, in: Signal Processing, Sensor/Information Fusion, and Target Recognition XXXII, vol. 12547, SPIE, 2023, pp. 114–143.
[45] G. Tauzin, U. Lupo, L. Tunstall, J.B. Pérez, M. Caorsi, A.M. Medina-Mardones, A. Dassatti, K. Hess, giotto-tda: A topological data analysis toolkit for machine learning and data exploration, J. Mach. Learn. Res. 22 (39) (2021) 1–6, URL http://jmlr.org/papers/v22/20-325.html.
[46] F. Chazal, B. Michel, An introduction to topological data analysis: fundamental and practical aspects for data scientists, Front. Artif. Intell. 4 (2021) 667963.
[47] H. Edelsbrunner, J.L. Harer, Computational Topology: An Introduction, American Mathematical Society, 2022.
[48] P. Bubenik, P. Dłotko, A persistence landscapes toolbox for topological statistics, J. Symbolic Comput. 78 (2017) 91–114.
[49] B. Rieck, F. Sadlo, H. Leitte, Topological machine learning with persistence indicator functions, in: Topological Methods in Data Analysis and Visualization V: Theory, Algorithms, and Applications 7, Springer, 2020, pp. 87–101.
[50] F. Chazal, B.T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, Stochastic convergence of persistence landscapes and silhouettes, in: Proceedings of the Thirtieth Annual Symposium on Computational Geometry, 2014, pp. 474–483.