IISE Transactions

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie21

State-space modeling for degrading systems with stochastic neural networks and dynamic Bayesian layers

Md Tanzin Farhat & Ramin Moghaddass

To cite this article: Md Tanzin Farhat & Ramin Moghaddass (2023): State-space modeling for degrading systems with stochastic neural networks and dynamic Bayesian layers, IISE Transactions, DOI: 10.1080/24725854.2023.2185323

To link to this article: https://doi.org/10.1080/24725854.2023.2185323

IISE TRANSACTIONS
https://doi.org/10.1080/24725854.2023.2185323

State-space modeling for degrading systems with stochastic neural networks and dynamic Bayesian layers

Md Tanzin Farhat and Ramin Moghaddass

Department of Industrial and Systems Engineering, University of Miami, Coral Gables, FL, USA

ABSTRACT

To monitor the dynamic behavior of degrading systems over time, a flexible hierarchical discrete-time state-space model (SSM) is introduced that can mathematically characterize the stochastic evolution of the latent states (discrete, continuous, or hybrid) of degrading systems, dynamic measurements collected from condition monitoring sources (e.g., sensors with mixed-type outputs), and the failure process. This flexible SSM is inspired by Bayesian hierarchical modeling and recurrent neural networks without imposing prior knowledge regarding the stochastic structure of the system dynamics and its variables. The temporal behavior of degrading systems and the relationship between variables of the corresponding system dynamics are fully characterized by stochastic neural networks without having to define parametric relationships/distributions between deterministic and stochastic variables. A Bayesian filtering-based learning method is introduced to train the structure of the proposed framework with historical data. Also, the steps to utilize the proposed framework for inference and prediction of the latent states and sensor outputs are discussed. Numerical experiments are provided to demonstrate the application of the proposed framework for degradation system modeling and monitoring.

ARTICLE HISTORY
Received 29 July 2022
Accepted 9 February 2023

KEYWORDS
State-space models; Bayesian layers; recurrent neural networks; degradation monitoring

1. Introduction

The development of advanced sensors, smart devices, and wireless/remote sensing technologies has contributed significantly to the advancement of real-time monitoring and control of degrading systems. Most data acquisition frameworks in condition monitoring systems can routinely collect a large amount of the system's operating/environmental data, which can periodically deliver a potentially large volume of dynamic covariates (Meeker and Hong, 2014). The success of any data analytics approach for sensor-driven monitoring highly depends on the ability of the models employed to mimic the temporal and stochastic behavior of degrading systems. Although many mathematical frameworks have been developed for the modeling of system dynamics in degrading systems, no single model is known to be effective in all applications. State-Space Models (SSMs) are one of the most fundamental and widely used tools to model time-series data and dynamic systems (Zhang *et al.*, 2017). The popularity of SSMs stems from the fact that they are flexible and easily interpretable (Kantas *et al.*, 2015) and can represent the dynamic nature of many processes changing over time (Benidris *et al.*, 2015). Despite their popularity, current SSMs still have limitations that mainly stem from assumptions about their parametric forms, types of inputs and outputs, linearity, the Gaussian nature of system dynamics, and other mathematically convenient, but hard-to-justify, assumptions. This article aims to develop a flexible approach to extend the widely used SSMs and address some of their limitations through interpretable and dynamic hierarchical modeling and recurrent neural networks, so that the complex evolution of the system dynamics can be mathematically modeled and then large-scale sequential mixed-type sensor data can be utilized for real-time monitoring.

This article is motivated by degrading systems under dynamic operating and environmental conditions and sensor-based condition monitoring where accurate/sufficient knowledge with regard to the variables of the system dynamics (e.g., the parametric form and stochastic behavior of the degradation process and sensor outputs) is not available. Also, the framework is suitable for complex degrading systems where the parametric form assumed for the degradation process and the behavior of condition monitoring sensor outputs are hard to justify or simply made for mathematical convenience. This article aims at developing theoretically sound and fundamentally new methods that facilitate the implementation of large-scale, heterogeneous, and multi-source sensor data analytics for degradation monitoring. We should point out that the work in this article is not designed to replace current state-of-the-art SSMs for degradation monitoring. However, it is a new tool for nonlinear and non-Gaussian systems with potentially long-range time dependencies when no accurate/sufficient knowledge is available regarding the stochastic evolution of the system dynamics.

CONTACT Ramin Moghaddass ramin@miami.edu

Supplemental data for this article is available online at https://doi.org/10.1080/24725854.2023.2185323

The contributions made in the article can be summarized as follows: First, a flexible multilayer state-space framework is proposed for a partially observed hybrid degrading system. This framework can characterize the stochastic relationship between latent degradation states, condition monitoring sensor outputs, control variables, and failure processes with a dynamic Bayesian network. The proposed structure and all its functional relationships are characterized by neural networks and learned directly from past data; thus, no parametric distributions or prior assumptions are needed with regard to the stochastic behavior of the dynamic variables. The model has the flexibility to handle missing points and incorporate prior knowledge, when available (e.g., incorporating bounded system variables). Second, unlike most available SSMs for degrading systems, the proposed framework can handle mixed-type (e.g., binary, categorical, continuous, bounded) inputs/outputs, thus all types of sensor data and system dynamics' variables can be considered. Third, a Bayesian filtering-based Expectation-Maximization (EM) is proposed to fully train the structure of this framework with past data collected for observable variables. The article also discusses the steps to use the proposed structure for inference and prediction. Due to its generic structure, our framework can cover many available SSMs for degradation monitoring, such as hidden Markov, semi-Markov models, and proportional hazard models as special cases (see Section 3.1 for more details). Due to its generic structure and limited assumptions, our work can be applied in many settings where the evolution of the system dynamics can be reasonably represented by an SSM and when sufficient knowledge regarding the stochastic behavior of the system is not available. The key differences between this article and works on hybrid degrading systems (e.g., Long *et al.* (2022)) and fault detection methods driven by the hybrid data (e.g., Wei *et al.* (2009)) are the flexibility to define the relationship between discrete/latent states and sensor readings only by neural networks, accommodating mixed-type variables, and incorporating the failure process as part of the state-space system dynamics.

The rest of this article is organized as follows: In Section 2, we review similar works in the literature and discuss the limitations of available models. Section 3 discusses our proposed framework for data-driven system monitoring and its mathematical structure. Section 4 illustrates how to employ past data to fully train the structure of the proposed framework. In Section 5, we address two important problems of inference and prediction for the proposed framework. A comprehensive set of numerical examples is provided in Section 6. Finally, we conclude in Section 7 and discuss our future work.

2. Literature review

2.1. State-space modeling for degrading systems

SSMs are one of the most fundamental and prevalent tools in engineering that can represent the dynamic nature of many processes changing over time (Benidris *et al.*, 2015). Almost all discrete-time, continuous-state SSMs used for system health monitoring can be represented by the following form:

$$\boldsymbol{x}_{k+1} = f_{\boldsymbol{\Theta}_x}(\boldsymbol{x}_{1:k}, \boldsymbol{u}_{1:k}, \boldsymbol{v}_{1:k}), \quad \boldsymbol{y}_k = g_{\boldsymbol{\Theta}_y}(\boldsymbol{x}_{1:k}, \boldsymbol{u}_{1:k}, \boldsymbol{e}_{1:k}), \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the hidden system's state/health vector at time k, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ is the observable system outputs at time k, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the observable system input vector at time k, and $\boldsymbol{\Theta}_x \in \mathbb{R}^{n_{\Theta_x}}$ and $\boldsymbol{\Theta}_y \in \mathbb{R}^{n_{\Theta_y}}$ are the sets of model parameters used in mappings f and g, respectively. Here, $\boldsymbol{v}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{e}_k \in \mathbb{R}^{n_y}$ are often i.i.d vectors that are characterized by known probability distribution functions $p_v()$ and $p_e()$. In the case of a linear Gaussian SSM, f and g are linear mappings and $p_v()$ and $p_e()$ are Gaussian distributions (e.g., Rigamonti *et al.* (2016)). Despite its popularity, (1) has the following limitations/assumptions when used for degradation monitoring and control. First, prior/accurate knowledge regarding the parametric forms of mappings f and g and probability density functions (pdfs) $p_v()$ and $p_e()$ that govern the stochastic evolution of the system dynamics is not always available, particularly for non-Markovian, nonlinear, and non-Gaussian systems with long-range time dependencies. Second, more SSMs on degradation modeling have focused on a one-dimensional latent state and a continuous observation process. Thus, they are not designed to handle mixed-type inputs and outputs. For example, parametric forms of mappings f and g in (1) that can simultaneously characterize categorical, binary, and continuous variables do not exist. Third, most SSMs employed for degradation monitoring often assume that: (a) the system fails when the state process or a combination of state and observation processes reaches a predefined threshold (e.g., Hu *et al.* (2015)); and (b) the system is subject to only one failure mode (e.g., Wang *et al.* (2016)). This article proposes a framework that deals with the above-mentioned limitations in a flexible manner.

2.2. Bayesian filtering and dynamic Bayesian networks

Bayesian modeling is one of the most powerful and widely used frameworks for analyzing SSMs. The limitations of conventional Bayesian techniques and their inability to deal with high dimensional data of a mixed-type (continuous, binary, categorical) have made their implementation in degradation monitoring and control limited. A Bayesian Belief Network (BBN) is a graphical model that allows the correlation and dependencies between variables to be modeled in a causal manner (Krieg *et al.*, 2001). Given the generative structure of SSMs with multiple levels, a particular type of BBN referred to as Bayesian Hierarchical Modeling (BHM) is a natural fit to formalize the relationship between system variables while taking uncertainty, numerical instability, and overfitting into account. Despite their popularity and strong mathematical structures, BHM and BBN are subject to serious shortcomings, particularly for large-scale settings with many variables/levels, sensor measurements, and complex temporal structures with dependencies between variables. The hierarchical variant of Dynamic Bayesian Networks (DBNs), which allows for developing hierarchical models

with stochastic and deterministic nodes and temporal structures, inherently benefits from both DBN and BHM and can address their shortcomings to a great extent. Despite its strong and flexible mathematical structure, limited work has been reported on Hierarchical Dynamic Bayesian Networks (HDBNs) for monitoring and controlling the dynamics of degrading systems using sensor data. In this article, important layers of system dynamics in a degrading system are formulated through an HDBN under the umbrella of an SSM. Our work is different from Bayesian filtering-based methods, such as Blom and Bloem (2007) and Schön *et al.* (2011), because it does not require a predefined parametric structure for the layers of system dynamics and has flexible/general capabilities (e.g., mixed-type inputs/outputs and a layer for the failure process).

2.3. Application of neural networks in state-space modeling

When accurate knowledge to parametrically characterize SSMs is not available, neural network-based models are excellent tools to remedy this limitation. For example in Wang *et al.* (2021), a deep neural network structure was employed to represent nonlinear state transitions and an observation process, and a variational autoencoder model was introduced for learning and inference. In Bao *et al.* (2020), an estimation and inference algorithm for a linear parameter-varying SSM was developed where neural network structures were used to represent the state and observation processes. Reference Yang and Shi (2019) considered a linear SSM with a known structure, and then used a linear neural network for parameter and state estimation. A variational Bayesian inference Neural Network (BNN) to quantify uncertainties in matrix function estimation for a linear SSM was developed in Bao *et al.* (2021). Recurrent Neural Networks (RNNs) have also been applied for time series analysis in SSMs (Che *et al.*, 2018). Despite the power to model the time-behavior start of arbitrary dynamic systems, typical RNNs cannot handle uncertainty in inputs and outputs. RNNs also cannot be used directly to analyze hierarchical sequential data with both stochastic and deterministic variables. Although the utilization of neural networks for the identification of nonlinear systems is not new, our framework has unique capabilities that make it different from similar approaches. First, unlike most available SSMs for nonlinear systems that include latent/hidden states and observations, our framework can potentially accommodate other types of variables, such as the time to reach a certain event and the discrete status of the system representing a switching behavior SSM. In our article, RNN is not used to estimate the parameters of an SSM with a known structure. Instead, it characterizes the dynamics of the system and the relationships between variables all from past data. Our work can handle mixed-type deterministic or stochastic inputs and outputs as well as missing points and can incorporate prior knowledge (if available) about the dynamics of the system by incorporating deterministic or stochastic monotonicity and bounded variables. Although there are similarities between our work and the work of Xiuqin and Ying (2021) that can be applied to hybrid systems, our model is more flexible as it can include mixed-type inputs and outputs and consider separate RNNs for each layer of the system dynamics. Also, unlike reference Xiuqin and Ying (2021) that uses approximate variational inference for model estimation, our model develops an EM algorithm that can be operated according to Particle Filtering (PF). Our model has an advantage over variational inference-based models that marginalize the discrete latent variable using an approximate posterior.

Neural networks and degradation monitoring have also been studied in the literature through physics-informed neural network (PINN) models. In such models, some prior knowledge regarding the physics of the degradation/failure process or other physics principles are combined with a neural network structure. For example in Nascimento *et al.* (2021), a hybrid modeling approach that directly implements physics within deep neural networks was developed for Li-ion battery prognosis. In Nascimento and Viana (2020), a cumulative damage process was modeled through a physics-informed recurrent neural network for monitoring fatigue crack growth in a fleet of aircraft. In Dourado and Viana (2020), a PINN approach for missing physics estimation in modeling a cumulative damage process was developed. In their model, physics-informed layers are used to model relatively well-understood phenomena and data-driven layers account for hard-to-model physics. Despite similarities, our work is different, mainly because no prior knowledge regarding the physics of the degradation process is required. Also, our model considers a multi-layer state-space degradation process in a generative form through multi-dimensional discrete and continuous latent states. However, in Dourado and Viana (2020); Nascimento and Viana (2020); Nascimento *et al.* (2021), the degradation process was considered as a one-dimensional process with no condition monitored sensors. Also, our model considers stochastic neural network frameworks where uncertainty is handled in a structured manner. Compared with our model, PINN models can provide more justifiable/interpretable insights and require smaller datasets to train, which may lead to a potentially faster training phase.

2.4. Summary of the literature gap and the proposed work

The following summarizes the limitations of the current literature for degrading systems modeled by SSMs: (i) low flexibility and generality of available models to define the stochastic relationships between high-dimensional sensor data of a mixed-type (e.g., discrete and continuous) and other multilevel latent and observable variables, (ii) hard to justify/unrealistic parametric/distributional assumptions used in defining the evolution of nonlinear system dynamics and difficulty in mathematically defining the dynamic and stochastic relationships between system variables, and (iii) lack of generic learning models for SSM training and system characterization that can fully operate based on data from the observable processes without having to assume fixed

Table 1. The list of the main notation used in this article.

N	*number of time series in the train set*	M	*number of latent discrete states*
m_k	the discrete latent state (switching state) at time k	$\boldsymbol{x}_k$	the continuous latent state vector at time k
$\boldsymbol{u}_k$	the controllable inputs at time k	$\boldsymbol{z}_k$	the binary vector representing sensor availability at time k
$\boldsymbol{y}_k$	the observation outputs vector at time k	$\boldsymbol{o}_k$	the binary vector representing the failure status of the system
$\boldsymbol{\theta}_k$	the set of parameters generated in the stochastic layers of RNNs at time k	$\boldsymbol{h}_k$	the hidden states of RNNs at time k
$\phi_{\boldsymbol{h}}$	the activation function from the input layers to the hidden layers of RNNs	$\phi_{\boldsymbol{\theta}^y}$	the activation function from the hidden layers to the stochastic layers of RNNs
$\boldsymbol{W}_{\boldsymbol{h}^y}$	the RNN parameters from the input layer to the hidden layers	$\boldsymbol{W}_{\boldsymbol{\theta}^y}$	the RNN parameters from the hidden layers to the output layer

parametric distributions to characterize the behavior of system variables over time.

3. Generic stochastic SSM with RNNs

This section illustrates the structure of the proposed framework. The list of the notation is in Table 1. We use uppercase/lowercase italic for constants/variables, italic bold for vectors, and uppercase bold for matrices.

3.1. SSM with mixed-type sensor data

We introduce a generic SSM to model the dynamics of a hybrid degrading system under condition monitoring. A hybrid SSM, which is also referred to as a switching behavior SSM, is a system where the latent states are represented by both discrete and continuous states. For degrading systems, discrete states can represent system fault status (e.g., normal, faulty, and partially faulty) and the continuous state can represent the latent degradation dynamics of the system. The stochastic description of this hybrid system at time k can be theoretically defined as follows (fixed covariates are removed for notational convenience):

$$\text{discrete latent state:}\quad m_k \sim p^m_{\boldsymbol{\theta}_m}(m_k|\boldsymbol{x}_{1:k-1}, m_{1:k-1}, \boldsymbol{u}_{1:k}, \boldsymbol{e}_{1:k}), \tag{2}$$

$$\text{continuous latent state:}\quad \boldsymbol{x}_k \sim p^x_{\boldsymbol{\theta}_x}(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, m_{1:k-1}, \boldsymbol{u}_{1:k}, \boldsymbol{e}_{1:k}), \tag{3}$$

$$\text{observation outputs:}\quad \boldsymbol{y}_k \sim p^y_{\boldsymbol{\theta}_y}(\boldsymbol{y}_k|\boldsymbol{x}_{1:k}, m_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k}, \boldsymbol{e}_{1:k}), \tag{4}$$

$$\text{failure process:}\quad \boldsymbol{o}_k \sim p^o_{\boldsymbol{\theta}_o}(\boldsymbol{o}_k|\boldsymbol{o}_{1:k-1}, \boldsymbol{x}_{1:k}, m_{1:k}, \boldsymbol{u}_{1:k}, \boldsymbol{e}_{1:k}), \tag{5}$$

where $\boldsymbol{e}_k$ is the vector of uncontrollable, but observable, operating inputs or environmental factors (also called external factors), $\boldsymbol{u}_k$ is the vector of controllable inputs, $m_k \in \{1, ..., M\}$ is the discrete latent variable (health mode), and $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are the multidimensional latent state and sensor observation/measurement vectors, respectively. Here, variable $\boldsymbol{z}_k \in \{0,1\}^{n_y}$ is a known dynamic binary vector that denotes whether sensor outputs are available at time k (i.e., $z_{k,j} = 1$ means the jth observation output is available at time k). This variable has the critical role of mimicking the behavior of real systems where sensor data are subject to missing values. Here, observable vector $\boldsymbol{o}_k \in \{0,1\}^{n_o}$ represents the overall system health status with regards to n_o types of failures. In other words, instead of limiting the framework by defining somewhat unrealistic fixed and known regions based on latent state variables (e.g., $\boldsymbol{x}$) to characterize n_o failure modes, a stochastic process is defined where the elements of the system dynamics are stochastically related to the probability of each failure mode over time. Although connecting a system's variables to the probability of failure has been employed many times before (e.g., Markov failure time process in Banjevic and Jardine (2006)), SSMs do not explicitly use the evolution of the failure time in the dynamics of the system. With defining this stochastic process, we also allow users to incorporate any prior knowledge/assumption (if available) regarding the evolution of the failure process. For instance, one may impose a Cox proportional hazard model, a bathtub, or a monotonic shape for the hazard probability function. Defining this process helps in better identification of the latent degradation states. This is numerically shown in Section 6.1.C. The structure of the model and inclusion of latent variables allow for marginal dependencies between failure events (e.g., competing failures can be modeled together). Here, $p^m_{\boldsymbol{\theta}_m}$ represents a probability mass function (pmf) for the distribution of m_k given $\boldsymbol{x}_{1:k-1}$, $m_{1:k-1}$, $\boldsymbol{u}_{1:k}$, and $\boldsymbol{e}_{1:k}$. In the simplest case, $p^m_{\boldsymbol{\theta}_m}$ can be defined by a Markovian transition matrix $\mathbf{P}$ of size $M \times M$, where $\mathbf{P}_{ij} = p(m_k = j|m_{k-1} = i)$. The probability density function $p^x_{\boldsymbol{\theta}_x}$ describes the evolution of the continuous latent states. For instance, it can be represented by a n_x-dimensional multivariate Gaussian distribution for random vector $\boldsymbol{x}_k$ given $\boldsymbol{x}_{1:k-1}$, $m_{1:k-1}$, $\boldsymbol{u}_{1:k}$, and $\boldsymbol{e}_{1:k}$. The pmf $p^o_{\boldsymbol{\theta}_o}$ describes the probability distribution of failure modes given $\boldsymbol{o}_{1:k-1}$, $\boldsymbol{x}_{1:k-1}$, $m_{1:k-1}$, $\boldsymbol{u}_{1:k}$, and $\boldsymbol{e}_{1:k}$. Since the variables for the observation outputs can be of different types (binary, categorical, continuous), a combination of pmfs, pdfs, or mixed-type probability distribution functions may be needed to define $p^y_{\boldsymbol{\theta}_y}$. This will be further discussed in Section 3.2.1. The parameter sets $\boldsymbol{\theta}_m$, $\boldsymbol{\theta}_x$, $\boldsymbol{\theta}_y$, and $\boldsymbol{\theta}_o$ characterize these pdfs.

The ability to define the causal dependencies between system variables through the Bayesian network structure helps with the interpretability of the SSM structure used in this article. Due to the generative and flexible structure of the SSM framework used in the article, many SSM-based models in degradation modeling can be formulated as a special case of our framework. For instance, hidden Markov models, Cox-proportional hazard models, and linear state-observation models can be easily modeled with our framework. This is because the way these models define the relationships between the latent degradation state and the observation process falls under the

generic system of equations defined in (2)-(5). For instance, in Hidden Markov model-based degradation frameworks (e.g., Zhao *et al.* (2021)), the discrete latent degradation state and the multi-dimensional observation process can be defined with (2) and (4), respectively. Typical Cox-proportional hazard model for degradation modeling (e.g., Zheng *et al.* (2021)) can be formulated by (3) to represent the continuous degradation states or other covariates (environmental factors) and (5) to represent the hazard process or probability of failure at any time. Also, linear/nonlinear state-observations models used for degradation analysis (e.g., Zhang *et al.* (2022)) can simply be defined by (3) and (4) for the continuous degradation state and the observation process, respectively. Even more advanced nonlinear systems, such as switching Kalman filter-based models, can be formulated by (2) to represent the switching mode, (3) to represent the state vector, and (4) to represent the observation process. The probability distribution functions can be defined by any form of time-dependent functions (with or without the Markov property) to incorporate time dependency and uncertainty into any variable of interest. All variables can theoretically be multidimensional and of any type (e.g., binary, categorical). Inspired by the concept of hybrid systems (Tafazoli and Sun, 2006; Orchard and Vachtsevanos, 2009), a discrete mode variable is defined to represent discrete levels of hidden working conditions or faults (e.g., normal and faulty) at which the dynamics of the system may change. Such variables can also be responsible for incorporating switching behaviors of system dynamics within its life cycle (e.g., Peng *et al.* (2018)). For notational convenience, we removed the effect of observable vectors $\boldsymbol{u}_k$ and $\boldsymbol{e}_k$ from future equations. Despite its general form, the SSM equations discussed in (2)-(5) are useful only when it is mathematically possible to define $p^m_{\theta_m}, p^x_{\theta_x}, p^y_{\theta_y}$, and $p^o_{\theta_o}$ and prior knowledge regarding their parametric forms is available.

3.2. Utilizing neural networks for stochastic system dynamics

In many systems, accurate knowledge with regards to the stochastic behavior of the system dynamics is not available, and thus assuming predefined parametric forms (such as linear and Gaussian structures) for the pdfs in (2)-(5) may be impossible or somewhat unrealistic. In this subsection, we show how to utilize RNNs in a stochastic manner. The idea is discussed for $p^y_{\theta_y}$, that is, the pdf that models the temporal behavior of multidimensional sensor data. We first consider an RNN with an input layer including $[m_{1:k}, \boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k}, \boldsymbol{e}_{1:k}]$ and hidden/latent layer including neurons $\boldsymbol{h}^y_{1:k}$. It should be noted that the input variables selected to define $\boldsymbol{y}_k$ can be determined by the users. We made a reasonable assumption that latent states, inputs, and missing point indicators are sufficient to define the stochastic behavior of sensor outputs $\boldsymbol{y}_k$. Here, recurrent layer $\boldsymbol{h}^y_k$ plays the role of the memory of the network capturing information regarding previous time steps. To generate sensor measurements $\boldsymbol{y}_k$ in a stochastic manner, we include a new layer $\boldsymbol{\theta}^y_k$, which maps the neural network's hidden layer(s) to a probability distribution over possible outputs of $\boldsymbol{y}_k$ as $\boldsymbol{y}_k \sim \pi^y(\boldsymbol{y}_k; \boldsymbol{\theta}^y_k)$. Here π^y is a pdf that is characterized by parameters $\boldsymbol{\theta}^y_k$. This process can be summarized as

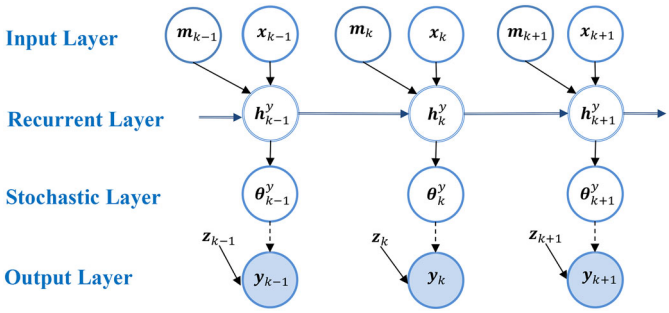

Figure 1. A simplified view of the relationship between inputs and outputs of the observable layer *y* layer at time *k* in a RNN form. Vertices represent stochastic variables; edges show the relationships between variables; filled circles are known/observable variables; unfilled circles are hidden variables; solid lines represent deterministic relationships; dashed lines represent stochastic relationships; and double-lined circles/edges denote recurrent variables. The parameters that characterize the uncertainty of the output variables are in the stochastic layer. Horizontal lines represent time dependencies.

$$\boldsymbol{h}^y_k = \phi_{\boldsymbol{h}^y}(m_k, \boldsymbol{x}_k, \boldsymbol{h}^y_{k-1} | \boldsymbol{W}_{\boldsymbol{h}^y}), \boldsymbol{\theta}^y_k = \phi_{\boldsymbol{\theta}^y}(\boldsymbol{h}^y_k | \boldsymbol{W}_{\boldsymbol{\theta}^y}), \tag{6}$$

$$\boldsymbol{y}_k \sim \pi^y(\boldsymbol{y}_k; \boldsymbol{\theta}^y_k, \boldsymbol{z}_k). \tag{7}$$

The pdf π^y would take an appropriate form depending on the type of variables in $\boldsymbol{y}$ (i.e., continuous, binary, categorical) and their stochastic behavior. It is clear that the choice of π determines the number of nodes for layer $\boldsymbol{\theta}^y_k$. The key point is that the probability distribution $\pi^y(\boldsymbol{y}_k; \boldsymbol{\theta}^y_k, \boldsymbol{z}_k)$ does not have any other parameters, and thus, is fully defined from previous layers of the network. The nonlinear deterministic mapping $\phi_{\boldsymbol{h}^y}$ (e.g., LSTM or GRU) and $\phi_{\boldsymbol{\theta}^y}$ (e.g., sigmoid or softmax) can be chosen by the users. In the recurrent layer of the RNN (e.g., LSTM), we can set the lookback (e.g., window size), which is the prior time step the model is looking back for learning the patterns and long-term dependencies. For a degrading system, a hypothesis could be that a longer lookback could capture more context for predictive patterns. However, as discussed in Section 3.3.2, activation functions $\phi_{\boldsymbol{\theta}^y}$ should be chosen so that parameter boundaries required in the parameter set $\boldsymbol{\theta}^y$ are not violated. For all other hidden layers, activation functions can be chosen from common and easy-to-use differentiable nonlinear activation functions, such as ReLu, Sigmoid, or Tanh. The parameters $\boldsymbol{W}_{\boldsymbol{h}^y}$ and $\boldsymbol{W}_{\boldsymbol{\theta}^y}$ denote the unknown weight and the bias vectors of the corresponding RNN. The graphical model to generate the sequence of sensor observations from the corresponding RNN is shown in Figure 1. It is interesting to note that despite the deterministic relationship between the input layer and the hidden layer of the RNN, all neurons in the hidden layer become random variables simply because the neurons in the input layers are random variables. For mathematical convenience, we refer to (6) as $\mathcal{RNN}^y(m_k, \boldsymbol{x}_k, \boldsymbol{z}_k, \boldsymbol{h}^y_k | \boldsymbol{W}_{\boldsymbol{h}^y}, \boldsymbol{W}_{\boldsymbol{\theta}^y})$. Based on the approach discussed above, the system dynamics equations can be represented by multilayer RNNs with an input layer, one or

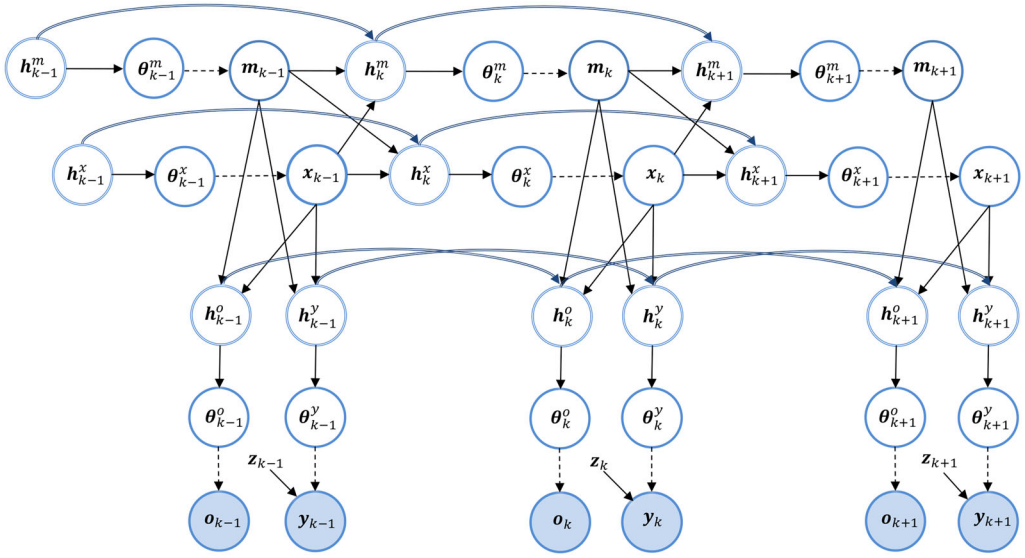

Figure 2. A system dynamics framework represented by a Bayesian network with four time-dependent layers of m, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{o}$ and the recurrent layer $\boldsymbol{h}$. The types of the nodes/edges are the same as Figure 1.

multiple hidden recurrent layers denoted by $\boldsymbol{h}$, the stochastic output layer denoted by $\boldsymbol{\theta}$, and the output layer. For example, the evolution of the hidden degradation state at the kth time interval is represented by $\mathcal{RNN}^x(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}_k^y | \boldsymbol{W}_{\boldsymbol{h}^x}, \boldsymbol{W}_{\boldsymbol{\theta}^x})$, where the structure of the $\mathcal{RNN}^x$ and latent state $\boldsymbol{\theta}_k^x$ is as follows:

$$\begin{aligned}\mathcal{RNN}^x(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}_k^y | \boldsymbol{W}_{\boldsymbol{h}^x}, \boldsymbol{W}_{\boldsymbol{\theta}^x}) \rightarrow \quad & \boldsymbol{h}_k^x \\ = \phi_{\boldsymbol{h}^x}(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}_{k-1}^x | \boldsymbol{W}_{\boldsymbol{h}^x}), \quad & \boldsymbol{\theta}_k^x = \phi_{\boldsymbol{\theta}^x}(\boldsymbol{h}_k^x | \boldsymbol{W}_{\boldsymbol{\theta}^x}),\end{aligned} \tag{8}$$

$$\boldsymbol{x}_k \sim \pi^x(\boldsymbol{x}_k; \boldsymbol{\theta}_k^x). \tag{9}$$

For the discrete-state variable m, we have

$$\begin{aligned}\mathcal{RNN}^m(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}_k^y | \boldsymbol{W}_{\boldsymbol{h}^m}, \boldsymbol{W}_{\boldsymbol{\theta}^m}) \rightarrow \quad & \boldsymbol{h}_k^m \\ = \phi_{\boldsymbol{h}^m}(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}_{k-1}^m | \boldsymbol{W}_{\boldsymbol{h}^m}), & \boldsymbol{\theta}_k^m = \phi_{\boldsymbol{\theta}^m}(\boldsymbol{h}_k^m | \boldsymbol{W}_{\boldsymbol{\theta}^m}),\end{aligned} \tag{10}$$

$$m_k \sim \pi^m(m_k; \boldsymbol{\theta}_k^m). \tag{11}$$

Finally, the stochastic behavior of the failure modes can be represented by

$$\begin{aligned}\mathcal{RNN}^o(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}_k^y | \boldsymbol{W}_{\boldsymbol{h}^o}, \boldsymbol{W}_{\boldsymbol{\theta}^o}) \rightarrow \quad & \boldsymbol{h}_k^o \\ = \phi_{\boldsymbol{h}^o}(m_k, \boldsymbol{x}_k, \boldsymbol{h}_{k-1}^o | \boldsymbol{W}_{\boldsymbol{h}^o}), & \boldsymbol{\theta}_k^o = \phi_{\boldsymbol{\theta}^o}(\boldsymbol{h}_k^o | \boldsymbol{W}_{\boldsymbol{\theta}^o}),\end{aligned} \tag{12}$$

$$\boldsymbol{o}_k \sim \pi^o(\boldsymbol{o}_k; \boldsymbol{\theta}_k^o). \tag{13}$$

To characterize this dynamic system (i.e., (6)-(13)), the structure of the RNNs defined by $\phi_{\boldsymbol{h}^m}$, $\phi_{\boldsymbol{\theta}^m}$, $\phi_{\boldsymbol{h}^x}$, $\phi_{\boldsymbol{\theta}^x}$, $\phi_{\boldsymbol{h}^y}$, $\phi_{\boldsymbol{\theta}^y}$, $\phi_{\boldsymbol{h}^o}$, and $\phi_{\boldsymbol{\theta}^o}$; the number of layers and neurons within each layer, pdfs $\pi_{\boldsymbol{\theta}_k^m}$, $\pi_{\boldsymbol{\theta}_k^x}$, $\pi_{\boldsymbol{\theta}_k^y}$, and $\pi_{\boldsymbol{\theta}_k^o}$; and weight (and bias) vectors of the RNN, denoted by $\boldsymbol{W} = \{\boldsymbol{W}_{\boldsymbol{h}^m}, \boldsymbol{W}_{\boldsymbol{h}^x}, \boldsymbol{W}_{\boldsymbol{h}^y}, \boldsymbol{W}_{\boldsymbol{h}^o}, \boldsymbol{W}_{\boldsymbol{\theta}^m}, \boldsymbol{W}_{\boldsymbol{\theta}^x}, \boldsymbol{W}_{\boldsymbol{\theta}^y}, \boldsymbol{W}_{\boldsymbol{\theta}^o}\}$, should be known. The structure of this dynamic system can be shown by a DBN in Figure 2, which has stochastic recurrent latent variables with deterministic and stochastic variables and causal relationships. We utilize Long Short Term Memory networks, referred to as LSTMs, which are a powerful form of RNNs that can deal with the well-known RNN issues of vanishing and exploding gradients. Using Bayesian networks to describe the relationship between variables not only helps with interpretability and understanding of system layers, but also introduces unique ways to formulate a physical system under monitoring through SSM and neural networks.

3.2.1. Defining the pdfs $\{\pi^m, \pi^x, \pi^y\}$

The vector-valued pdfs $\pi^{\cdot}$ have a critical role in the successful representation of the evolution of system dynamics. The key idea is to define these functions in a generic way while making sure that they are fully characterized through the neurons in layer $\boldsymbol{\theta}$. We first discuss the selection of function π^y for the observation process because it is the layer that is likely to have mixed-type variables. For example, for a continuous $\boldsymbol{y}$, the nodes in layer $\boldsymbol{\theta}_k^y$ should include the elements of the mean vector $\boldsymbol{\mu}_k^y$ and the covariance matrix $\boldsymbol{\Sigma}_k^y$ (i.e., $\boldsymbol{\theta}_k^y = \{\boldsymbol{\mu}_k^y, \boldsymbol{\Sigma}_k^y\}$) of a multivariate Gaussian distribution, so that

$$\boldsymbol{y}_k \sim \mathcal{N}(\boldsymbol{\mu}_k^y, \boldsymbol{\Sigma}_k^y). \tag{14}$$

In other words, the outcome of the $\mathcal{RNN}^y$ would give stochastic information regarding the value of sensor outputs represented by a mean vector and a covariance matrix. We should highlight the fact that we do not assume that $\boldsymbol{y}$ follows a multivariate Gaussian distribution with a fixed mean and covariance. Instead, we assumed observation outputs $\boldsymbol{y}_k$ follows a multivariate Gaussian distribution with a dynamic mean and covariance (that is $\boldsymbol{y}_k$ and $\boldsymbol{y}_{k+1}$ do not follow the same probability distribution). Similarly, $\boldsymbol{y}_k$ of two different systems follow different probability distribution that depends on the history of the system. It should be pointed out that other pdfs can be used for the stochastic layers; however, its performance particularly for the backpropagation step should be carefully investigated. For instance, a very generic choice for continuous observation outputs would be a Gaussian

Mixture Model (GMM), which is known to be a universal approximator of densities. Our framework can use a GMM by including mixture of components in the stochastic layers. From a feasibility point of view, one can choose any probability distribution/mass function that can be parameterized. For a binary variable in $\boldsymbol{y}$, we include a non-zero constant p_k^y in the stochastic layer $\boldsymbol{\theta}_k^y$, and then use the sigmoid activation function (which ranges between zero and one) to find the probability of y_k being one as:

$$\Pr(y_k = 1) = (1 + \exp(-p_k^y))^{-1}. \tag{15}$$

Here, we need a vector $\boldsymbol{p}_k^y$ (i.e., $\boldsymbol{\theta}_k^y = \{\boldsymbol{p}_k^y\}$) to accommodate multivariate binary vector $\boldsymbol{y}_k$. Finally, for a l-dimensional categorical variable y_k, we should include neurons $\{a_1^y, a_2^y, ..., a_l^y\}$ in the stochastic layer $\boldsymbol{\theta}_k^y$ and then use the softmax activation function to get the probability of each class as follows:

$$\Pr(\boldsymbol{y}_k = f) = \exp(a_{k,f}^y)\left(\sum_{f'=1}^{l} \exp(a_{k,f'}^y)\right)^{-1}, \; f \in \{1, ..., l\}. \tag{16}$$

Now for a system with $n_y = n_c + n_b + n_a$ variables in the observation process, where n_c, n_b, and n_a are respectively the number of continuous, binary, and categorical variables, the stochastic layer $\boldsymbol{\theta}_k^y$ would need to include a n_c-vector for the mean, a $n_c \times n_c$ matrix for the covariance matrix, n_b neurons to cover binary variables, and $\sum_{f=1}^{n_a} l_f$ neurons to cover categorical variables. In other words, the $\boldsymbol{\theta}_k^y$ layer of $\mathcal{RNN}^y$ should include all $n_c + n_c \times n_c + n_b + n\sum_{f=1}^{n_a} l_f$ neurons that characterize the stochastic behavior of sensor outputs. We should point out that the framework discussed above does not assume that output variables are independent. In fact, since all output variables are at the same level and dependent on previous layers, they are marginally dependent (or only conditionally independent given neurons in layer $\boldsymbol{\theta}$). If prior knowledge is available with regards to the causal dependencies between sensor outputs, one can incorporate such dependencies by changing the inputs and outputs in (6)-(7). The second way to incorporate dependency is from the probability distribution used in the output layer that provides a joint distribution of output variables. For instance, when multivariate Gaussian distribution is used, then the correlation between continuous sensor outputs may be captured with the non-diagonal elements of the covariance matrix. For categorical distribution, the SoftMax function provides a joint probability distribution where the sum of probabilities of all categories is one. Our framework can handle two important special cases with respect to the observation process, namely, missing data and bounded data. We have shown in Appendices 1 and 2 how these concepts can be taken into account by modifying the probability distribution functions. The ability to handle these two concepts helps with numerical stability and extends its application to more complex degrading systems.

3.3. RNN training with stochastic variables in the output layer

Although the parameters and the elements of the stochastic RNN structures in each layer of the Bayesian network may be different, the process to train these networks is the same. Below, we discuss how a generic loss function can be used as a criterion for training $\mathcal{RNN}^y$. In Section 4, we discuss how to conduct model training when only data for observable layers are available. Results presented here can be extended to $\mathcal{RNN}^m$, $\mathcal{RNN}^x$, and $\mathcal{RNN}^o$ in a similar manner. Typical neural networks are often trained by defining appropriate loss functions, which are chosen based on the expected target values and how they can be compared to true values. Typically, loss functions in deterministic neural networks are structured in a way that the true value and the estimated value are used to find the error or the loss. However, in the structure used in this article, the loss function should handle the stochastic nature of three types of variables (i.e., continuous, binary, and categorical). This makes the selection of the appropriate loss function very important. Due to the stochastic nature of the variables in the output layers, the negative log of the probability of output variables is considered as the loss function for model training. This is very common across stochastic neural network models. Unlike available deterministic and stochastic neural networks, the stochastic recurrent neural network in this article has a multiterm loss function due to the nature of the output layer that includes multidimensional and mixed-type variables. Such loss functions offer the flexibility of utilizing separate loss functions for separate neurons in the output layer. The first loss function is the binary cross-entropy (BCE), a simple and commonly used loss function originally defined for binary classification. This loss function can be used to predict the probability values for all binary variables in the system, denoted by S_{bin},

$$\begin{aligned}\mathcal{L}_{\text{bin}}(\boldsymbol{W}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{T_n}\sum_{j\in S_{bin}} y_{k,j}^n . \log(\sigma(z_{k,j}^n y_{k,j}^n)) \\ + (1 - y_{k,j}^n). \log(1 - \sigma(z_{k,j}^n y_{k,j}^n)), \text{ where} \\ \sigma(z_{k,j}^n y_{k,j}) = \frac{1}{1 + z_{k,j}\exp(-p_k^y)},\end{aligned}$$

$y_{k,j}^n$ is the true value of the jth binary variable (from set S_{bin}) for time series n, N is the number of time series in the train set, and T_n is the length of the nth time series. For categorical variables (set S_{cat}), we treat the estimation problem as a multiclass classification problem with the cross-entropy loss function as

$$\begin{aligned}\mathcal{L}_{\text{cat}}(\boldsymbol{W}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{T_n}\sum_{j\in S_{cat}} \mathbb{1}\{z_{k,j}^n = 0\}\log(1 - z_{k,j}) \\ + \sum_{c=1}^{l_j} \mathbb{1}\{y_{k,j}^n \\ = c, \; \{z_{k,j}^n = 1\}\}. \log\left(\frac{z_{k,j}\exp(a_{k,c}^y)}{(\sum_{f'=1}^{l_j}\exp(a_{k,f'}^y))}\right).\end{aligned}$$

Here, l_j is the number of categories for the jth categorical variable. For both of the above loss functions, when an observation is missing (i.e., when $z_{k,j} = 0$), then the corresponding value in the loss function is zero. This ensures that missing points do not impact the training process. The parameters of the continuous variables are the mean and covariance components that are defined in the output layer of the RNN. Assuming $\boldsymbol{\mu}_k^y$ as the mean vector and $\boldsymbol{\Sigma}_k^y$ as the covariance matrix with $\boldsymbol{y}_k^n$ as the true observation vector, the loss function for all continuous variables can be computed as follows:

$$\mathcal{L}_{\text{con}}(\boldsymbol{W}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{T_n}\log\left(\Pr(\boldsymbol{y}_k^n|\boldsymbol{\mu}_k^n, \boldsymbol{\Sigma}_k^n, \boldsymbol{z}_k^n)\right)$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{T_n}\frac{1}{2}\log|\boldsymbol{\Sigma}_{\cdot k}^{\cdot n}| + \frac{1}{2}(\boldsymbol{y}_k^n - \boldsymbol{z}_k^n\boldsymbol{\mu}_k^n)^T\left[\boldsymbol{\Sigma}_{\cdot k}^{\cdot n}\right]^{-1}(\boldsymbol{y}_k^n - \boldsymbol{z}_k^n\boldsymbol{\mu}_k^n).$$

Here $|\ |$ is the determinant operator and $\boldsymbol{\Sigma}_{\cdot k}^{\cdot n}$ is the covariance matrix after the effect of vector $\boldsymbol{z}_k^n$, that is,

$$\begin{aligned}\boldsymbol{\Sigma}_{\cdot k}^{\cdot n} &= \boldsymbol{\Sigma}_k^n \odot \text{diag}(\boldsymbol{z}_k^n) + (\boldsymbol{I}_{n_{con}} - \text{diag}(\boldsymbol{z}_k^n))\\ &= \text{diag}(\boldsymbol{z}_k^n \odot \boldsymbol{\Sigma}_k^n + ([1, \ldots, 1]_{1\times n_{con}} - \boldsymbol{z}_k^n)),\end{aligned}$$

where $\odot$ is the Hadamard product (element-wise product), $\boldsymbol{I}_{n_{con}}$ is an identity matrix of size $n_{con} \times n_{con}$, and $\text{diag}(\boldsymbol{z}_k^n)$ is a diagonal matrix with diagonal elements $\boldsymbol{z}_k^n$. It can be seen that all dimensions with missing points ($z = 0$) will be ignored, that is, the corresponding elements in the loss functions are all zero. The loss function can be easily extended to the truncated normal distribution by dividing the loss function into three elements as discussed in Appendix 2. When using multiple loss functions in neural networks, each loss function requires a weight parameter that is used while optimizing the targets with multiple outputs and separate loss functions. The loss weights can be considered as hyperparameters that need to be tuned in by cross-validation. The overall loss function can now be represented as

$$\mathcal{L}_{\text{con}}(\boldsymbol{W}) = \gamma_1\mathcal{L}_{\text{bin}}(\boldsymbol{W}) + \gamma_2\mathcal{L}_{\text{cat}}(\boldsymbol{W}) + \gamma_3\mathcal{L}_{\text{con}}(\boldsymbol{W}).$$

The hyperparameters γ_1, γ_2, and γ_3 are tuned using a grid search across a validation set. Standard backpropagation utilizes a gradient descent algorithm in which the weights are updated along with the negative of the gradient of the loss function and a combination of weights that minimizes the predefined loss function.

3.3.1. The loss function for the failure process

The loss function discussed earlier can only handle regular binary, categorical, and continuous variables and cannot be used for variables that represent events with a monotonic rate of occurrence. Due to the page limit, we discuss the structure of the loss function for the failure process in Appendix 3.

3.3.2. Appropriate stochastic layer's activation functions

Activation functions have a very important role in the design and performance of the proposed framework, particularly in the control of the learning process using the training set. In the proposed framework, the activation function used in the hidden stochastic layers (i.e., $\phi_{\boldsymbol{\theta}}$) should be carefully chosen to make sure binary, categorical, continuous (regular and truncated), and failure process variables are generated in their appropriate domains. We added a discussion in Appendix 4 to discuss possible activation functions for different layers of the system dynamics. In practice, we can conduct cross-validation to fine-tune the best activation functions from a set of feasible ones, usually through a grid search process.

4. Learning with data from the observable layers

The loss functions in Section 3 can only be evaluated if all variables/parameters in the inputs/output layers are known. However, the structure of the proposed SSM is generative with multiple latent/hidden and observable layers and variables where both inputs and outputs can be stochastic. Thus, unlike loss functions in well-known deterministic and stochastic neural networks, the direct evaluation of the discussed loss functions for the purpose of model training is not possible. Furthermore, available methods with algorithms with negative log-likelihood as the loss function can only deal with continuous or discrete variables at a time (thus mixed-type variables cannot be handled directly). That is why a new EM-based training algorithm is defined for model learning. In this section, we develop an optimization approach based on the well-known EM method that shows how the entire structure shown in Section 3 can be characterized with historical data obtained only from observable layers. We first assume that there are N sequences of training data (time series) that include sensor data $\boldsymbol{y}_{1:T_n}^n$; sensor data availability vectors $\boldsymbol{z}_{1:T_n}^n$; and failure data $\boldsymbol{o}_{1:T_n}^n$, where T_n is the length of the nth time series. All other system dynamics elements, including $m_{1:T_n}$, $\boldsymbol{x}_{1:T_n}$, and hidden variables in the corresponding Bayesian network are unknown. The only unknown parameters of the proposed framework are the weight and bias vectors of the corresponding RNNs denoted by $\boldsymbol{W}$. Given historical observations, the log of the likelihood function for observed data is

$$\begin{aligned}\log\mathcal{L}(\boldsymbol{W}) = \sum_{n=1}^{N}\Pr(\boldsymbol{y}_{1:T_n}^n, \boldsymbol{o}_{1:T_n}^n|\boldsymbol{W}) \rightarrow\\ \boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} \quad \log\mathcal{L}(\boldsymbol{W}).\end{aligned} \tag{17}$$

The direct evaluation of the log-likelihood in terms of $\boldsymbol{W}$ is not possible because of latent variables and deterministic and stochastic relationships among all variables. In this article, we design an EM-based model that creates an approximation $\mathcal{Q}$-function of the likelihood that is tractable based on the structure of the proposed framework. The $\mathcal{Q}$-function is represented as follows:

$$\begin{aligned}\mathcal{Q}(\boldsymbol{W}, \boldsymbol{W}^{old}) = \sum_{n=1}^{N}\int \log\Pr(\boldsymbol{a}_{1:T_n}^n, \boldsymbol{y}_{1:T_n}^n, \boldsymbol{o}_{1:T_n}^n|\boldsymbol{W})\\ \Pr(\boldsymbol{a}_{1:T_n}^n|\boldsymbol{y}_{1:T_n}^n, \boldsymbol{z}_{1:T_n}^n, \boldsymbol{o}_{1:T_n}^n, \boldsymbol{W}^{old})\text{d}\boldsymbol{a}_{1:T_n}^n,\end{aligned} \tag{18}$$

where $\boldsymbol{a}^n_{1:T_n}$ represents all hidden variables in the model for the nth time series with length T_n. It is known that maximizing $\mathcal{Q}$ iteratively leads to increasing the likelihood function. Before we discuss the structure of the EM algorithm, we show how to evaluate the $\mathcal{Q}$-function with available historical data. From the structure of the proposed Bayesian network, we can show that for a known set of initial values, the following holds:

$$\begin{aligned}&\log \Pr(\boldsymbol{a}_{1:T_n}, \boldsymbol{y}_{1:T_n}, \boldsymbol{z}_{1:T_n}, \boldsymbol{o}_{1:T_n}|\boldsymbol{W})\\&= \sum_{k=1}^{T_n} \log \Pr(m_k|m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}^m_k)\\&+ \log \Pr(\boldsymbol{x}_k|m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}^x_k)\\&+ \log \Pr(\boldsymbol{y}_k|m_k, \boldsymbol{x}_k, \boldsymbol{h}^y_k) + \log \Pr(\boldsymbol{o}_k|m_k, \boldsymbol{x}_k, \boldsymbol{h}^o_k).\end{aligned} \tag{19}$$

Although using the above decomposition can significantly lower the complexity of computing the $\mathcal{Q}$-function, the elements in this equation contain hidden variables and still cannot be evaluated. We will show below that the causal dependencies between Bayesian network variables can help with evaluating the complex integrals in the $\mathcal{Q}$-function. We first show how each term in (19) can be simplified in the $\mathcal{Q}$-function. For notational convenience, we use $\boldsymbol{e}_k$ to refer to all observable variables at time k, that is, $\boldsymbol{e}_k = [\boldsymbol{y}_k, \boldsymbol{z}_k, \boldsymbol{o}_k]$ and discuss the model training for just one time series with length K. We first start with the evaluation of the elements of the discrete mode variables as follows:

$$\begin{aligned}&\int \log \Pr(m_k|m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}^m_k) \Pr(\boldsymbol{a}_{1:K}|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old}) \mathrm{d}\boldsymbol{a}_{1:K}\\&= \int\int\int \log \Pr(m_k|m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}^m_k)\\&\Pr(\boldsymbol{m}_k, m_{k-1}, \boldsymbol{x}_{k-1}|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old}) \mathrm{d}\boldsymbol{x}_{k-1} \mathrm{d}m_{k-1} \mathrm{d}m_k.\end{aligned} \tag{20}$$

For the elements of the latent state $\boldsymbol{x}$, we have

$$\begin{aligned}&\int\int\int \log \Pr(\boldsymbol{x}_k|m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{h}^x_k)\\&\Pr(m_{k-1}, \boldsymbol{x}_{k-1}, \boldsymbol{x}_k|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old}) \ \mathrm{d}\boldsymbol{x}_k \mathrm{d}\boldsymbol{x}_{k-1} \mathrm{d}m_{k-1}.\end{aligned} \tag{21}$$

For the elements of sensor data $\boldsymbol{y}$, we have

$$\int\int \log \Pr(\boldsymbol{y}_k|m_k, \boldsymbol{x}_k, \boldsymbol{h}^y_k) \Pr(m_k, \boldsymbol{x}_k|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old}) \ \mathrm{d}\boldsymbol{x}_k \mathrm{d}m_k \tag{22}$$

Finally, for the elements of $\boldsymbol{o}$, we have

$$\int\int \log \Pr(\boldsymbol{o}_k|m_k, \boldsymbol{x}_k, \boldsymbol{h}^o_k) \Pr(m_k, \boldsymbol{x}_k|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old}) \ \mathrm{d}\boldsymbol{x}_k \mathrm{d}m_k. \tag{23}$$

It should be noted that the integral terms should be converted to summation for discrete variables. Although the integrals in (20)-(23) are less complicated than their original forms, numerically evaluating them is still intractable, due to the existence of integration on latent variables over a potentially large range of K. In this article, we utilize a PF approach to approximate the $\mathcal{Q}$-function numerically through discretization together with sequential importance resampling. Based on the PF approach, the posterior distribution of the latent variables can be computed by a weighted sum of P samples (particles) drawn from the posterior distribution. Let us denote $m^{(p)}_k, \boldsymbol{x}^{(p)}_k$ as the pth sample (particle) drawn from the posterior distribution. For instance, we can approximate $\Pr(\boldsymbol{x}_k, m_k|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old})$ as

$$\Pr(m_k, \boldsymbol{x}_k|\boldsymbol{e}_{1:k}, \boldsymbol{W}^{old}) \approx \sum_{p=1}^{P} w^{(p)}_k \delta\Big(\{m_k, \boldsymbol{x}_k\}, \{\boldsymbol{x}^{(p)}_k, m^{(p)}_k\}\Big), \tag{24}$$

where $w^{(p)}_k$ is the cumulative importance weights associated with particle series $\{m^{(p)}_k, \boldsymbol{x}^{(p)}_k\}$ and δ is the Dirac delta function. To avoid weight degeneracy, we can apply the resampling step at the end of each step. During the resampling step at iteration $k-1$, a new set of particles is generated by resampling with replacement P times. As a result, the particle importance weights become $1/P$ after resampling. By using the transition prior density as the proposal distribution and applying resampling at each step, we have

$$\begin{aligned}w^{(p)}_k &= \Pr(\boldsymbol{y}_k, \boldsymbol{o}_k|m^{(p)}_k, \boldsymbol{x}^{(p)}_k, \boldsymbol{W}^{old})\\&= \frac{\pi^y(\boldsymbol{y}^n_k; \boldsymbol{\theta}^{y(p)}_k) \quad \pi^o(\boldsymbol{o}^n_k; \boldsymbol{\theta}^{o(p)}_k)}{\sum_{p=1}^{P} \pi^y(\boldsymbol{y}^n_k; \boldsymbol{\theta}^{y(p)}_k) \quad \pi^o(\boldsymbol{o}^n_k; \boldsymbol{\theta}^{o(p)}_k)},\end{aligned} \tag{25}$$

where $\boldsymbol{\theta}^{y(p)}_k$ and $\boldsymbol{\theta}^{o(p)}_k$ are calculated from the updated $\mathcal{RNN}^y$ and $\mathcal{RNN}^o$ in a forward manner. Since resampling makes the resampled particles equally important, the $\mathcal{Q}$-function can be approximated as follows:

$$\begin{aligned}&\mathcal{Q}(\boldsymbol{W}, \boldsymbol{W}^{old})\\&\approx \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{q=1}^{P} w^{(pqm)}_k \log \Pr(m^{(p)}_k|m^{(q)}_{k-1}, \boldsymbol{x}^{(q)}_{k-1}, \boldsymbol{h}^m_k, \boldsymbol{W}_{\boldsymbol{h}^m}, \boldsymbol{W}_{\boldsymbol{\theta}^m})\\&+ \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{q=1}^{Q} w^{(pqx)}_k \log \Pr(\boldsymbol{x}^{(p)}_k|m^{(q)}_{k-1}, \boldsymbol{x}^{(q)}_{k-1}, \boldsymbol{h}^x_k, \boldsymbol{W}_{\boldsymbol{h}^x}, \boldsymbol{W}_{\boldsymbol{\theta}^x})\\&+ \sum_{k=1}^{K}\sum_{p=1}^{P} w^{(p)}_k \log \Pr(\boldsymbol{y}^{(p)}_k|m^{(p)}_k, \boldsymbol{x}^{(p)}_k, \boldsymbol{h}^y_k, \boldsymbol{W}_{\boldsymbol{h}^y}, \boldsymbol{W}_{\boldsymbol{\theta}^y})\\&+ \sum_{k=1}^{K}\sum_{p=1}^{P} w^{(p)}_k \log \Pr(\boldsymbol{o}^{(p)}_k|m^{(p)}_k, \boldsymbol{x}^{(p)}_k, \boldsymbol{h}^o_k, \boldsymbol{W}_{\boldsymbol{h}^o}, \boldsymbol{W}_{\boldsymbol{\theta}^o}).\end{aligned} \tag{26}$$

where, $w^{(pqm)}_k$ and $w^{(pqx)}_k$ can be approximated as

$$\begin{aligned}w^{(pqm)}_k &= \frac{w^{(p)}_k w^{(q)}_{k-1} \Pr(m^{(p)}_k|m^{(q)}_{k-1}, \boldsymbol{x}^{(q)}_{k-1}, \boldsymbol{h}^m_k)}{\sum_{g=1}^{P} w^{(g)}_{k-1} \Pr(m^{(p)}_k|m^{(g)}_{k-1}, \boldsymbol{x}^{(g)}_{k-1}, \boldsymbol{h}^m_k)},\\w^{(pqx)}_k &= \frac{w^{(p)}_k w^{(q)}_{k-1} \Pr(\boldsymbol{x}^{(p)}_k|m^{(q)}_{k-1}, \boldsymbol{x}^{(q)}_{k-1}, \boldsymbol{h}^x_k)}{\sum_{g=1}^{P} w^{(g)}_{k-1} \Pr(\boldsymbol{x}^{(p)}_k|m^{(g)}_{k-1}, \boldsymbol{x}^{(g)}_{k-1}, \boldsymbol{h}^x_k)}.\end{aligned} \tag{27}$$

All particle samples are estimated based on $\boldsymbol{W}^{old}$ and after the resampling step. Based on the above steps, the calculations needed for the expectation step can be computed based on the developed PF approach. The maximization steps require finding the optimal values of $\boldsymbol{W}$ that maximize the function $\mathcal{Q}(\boldsymbol{W}, \boldsymbol{W}^{old})$. Since the terms for $(\boldsymbol{W}_{\boldsymbol{h}^m}, \boldsymbol{W}_{\boldsymbol{\theta}^m})$, $(\boldsymbol{W}_{\boldsymbol{h}^x}, \boldsymbol{W}_{\boldsymbol{\theta}^x})$, $(\boldsymbol{W}_{\boldsymbol{h}^y}, \boldsymbol{W}_{\boldsymbol{\theta}^y})$, and $(\boldsymbol{W}_{\boldsymbol{h}^o}, \boldsymbol{W}_{\boldsymbol{\theta}^o})$ are independent, we can find the optimal values of these parameter vectors independently. It is clear that optimizing all RNN weight and bias vectors is

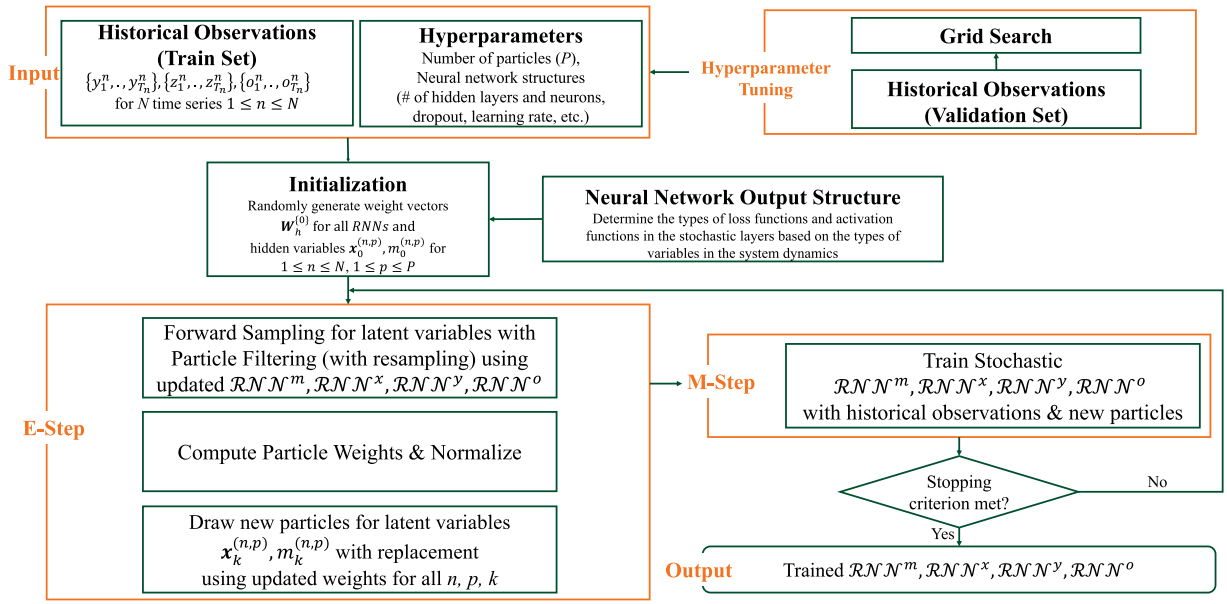

Figure 3. An overview of the main steps for structure learning.

equivalent to training the corresponding RNNs with the stochastic loss functions discussed earlier. The overall structure of the proposed EM algorithm can be summarized as follows: The algorithm starts with some initial values of the RNN weight and bias vectors depending on the structure of the RNNs selected for each layer. All hidden elements of the Bayesian networks are initialized first and then calculated according to a forward process using the RNNs parameterized by the most updated values of the weight vectors and PF samples. Then a set of particles is generated to estimate the Q-function. Since all particles have the same weights after resampling, they can be used as inputs/outputs to characterize the neural networks and update weight vectors. This process continues until a stopping criterion is met. In this article, we used the relative percentage decrease of the moving average of the total loss after the maximization step falling below a predefined threshold as the stopping criterion. The details of the training algorithm are discussed in Algorithm 1 in Appendix 5. Although the training process is an iterative process with potentially many steps (depending on the data size and RNN structures), it is done offline. Thus, CPU time for training is not a serious concern in this work. Figure 3 shows an overview of all the training steps.

5. Inference from the developed structure

In this section, we discuss how to use real-time sensor data to generate insights for the system's current state (filtering problem) and the future behavior of sensor outputs (prediction problem).

5.1. The filtering problem

To monitor the status of the latent variables over time, we need to be able to derive their stochastic distributions over time given the most updated set of observable data (e.g., sensor outputs). Given the set of observations up to time k, represented by $\boldsymbol{e}_{1:k} = [\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}]$, the probability distribution of hidden variables can be computed as follows:

$$\Pr(\boldsymbol{a}_k|\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}, \boldsymbol{W}) = \frac{\Pr(\boldsymbol{a}_k, \boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}, \boldsymbol{W})}{\Pr(\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}|\boldsymbol{W})}, \quad \text{where}$$
$$\Pr(\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}|\boldsymbol{W}) = \int \Pr(\boldsymbol{a}_{1:k}, \boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}|\boldsymbol{W}) \mathrm{d}\boldsymbol{a}_{1:k}. \tag{28}$$

Directly evaluating the above integral over all latent vectors is numerically impossible. Instead, we use a Bayesian filtering approximation of the latent variables' probability distribution functions. To do so, we can generate P particle sets $\boldsymbol{a}^{(p)} = (m^{(p)}, \boldsymbol{x}^{(p)})$ $(1 \le p \le P)$ in a forward manner based on the E-step in Algorithm 1. Now, the joint and marginal probability distributions of m_k and $\boldsymbol{x}_k$ can be estimated as

$$\Pr(m_k = m, \boldsymbol{x}_k = \boldsymbol{x}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) \approx \sum_{p=1}^{P} \frac{\delta(\{m, \boldsymbol{x}\}, \{\boldsymbol{x}_k^{(p)}, m_k^{(p)}\})}{P} \quad \forall (m, \boldsymbol{x}),$$
$$\Pr(m_k = m|\boldsymbol{e}_{1:k}, \boldsymbol{W}) \approx \sum_{p=1}^{P} \frac{1}{P} \delta(m, m_k^{(p)}),$$
$$\Pr(\boldsymbol{x}_k = \boldsymbol{x}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) \approx \sum_{p=1}^{P} \frac{1}{P} \delta(\boldsymbol{x}, \boldsymbol{x}_k^{(p)}) \quad \forall (m, \boldsymbol{x}).$$

The empirical distribution of particles at any time point can be used to derive the probability distribution of latent states. For the most likely estimates of $\boldsymbol{a}_k$ one may use the mean, mode, or median of the updated set of particles. Also, bootstrap methods may be used to find a prediction interval of the latent state estimates. The particle size plays an important role in the performance of the model and can directly impact modeling/estimation accuracy and convergence speed. One common approach to find the best particle size is to use a grid search from a range of possible particle sizes and then choose the one with the best performance on a

validation set. A larger number of particles often leads to more accurate estimates of the state of the system and a better modeling of the complexity of the system. However, increasing the number of particles leads to increased computation time, since more particles need to be processed.

5.2. The prediction problem

A very typical problem in SSMs is the prediction problem, which in the context of our framework can be represented by finding the following two important measures of interest:

$$\Pr(\boldsymbol{y}_{k+1}|\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}, \boldsymbol{W}) \quad \text{and} \quad \Pr(\boldsymbol{o}_{k+1}|\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}, \boldsymbol{W}).$$

The idea behind these two measures is to find the conditional probability of the next observations given the observable data received up to the current time. We discuss how to estimate the first measure; the second measure can be computed in a similar manner. We will first show that the prediction problem forms a hierarchical Bayesian framework and then propose a PF-based approximation trick utilizing a non-Bayesian Gaussian mixture problem. By defining $\boldsymbol{a}_{k+1} = \{\boldsymbol{x}_{k+1}, m_{k+1}\}$ as the unknown status of the hidden variables at time epoch $k+1$, the following holds true:

$$\Pr(\boldsymbol{y}_{k+1}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) = \int_{\boldsymbol{a}_{k+1}} \Pr(\boldsymbol{y}_{k+1}|\boldsymbol{a}_{k+1}, \boldsymbol{e}_{1:k}, \boldsymbol{W}) \Pr(\boldsymbol{a}_{k+1}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) \mathrm{d}\boldsymbol{a}_{k+1}.$$

Since the above integral has no closed-form solution, we can approximate it with discrete values of $\dot{\boldsymbol{a}}_{k+1}^{(p)}$ for $p \in \{1, ..., P\}$, which are the particles drawn using the observations up to time k. Now the prediction measure can be rewritten as $\Pr(\boldsymbol{y}_{k+1}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) \approx \sum_{p=1}^{P} \Pr(\boldsymbol{y}_{k+1}|\dot{\boldsymbol{a}}_{k+1}^{(p)}, \boldsymbol{e}_{1:k}, \boldsymbol{W}) \Pr(\dot{\boldsymbol{a}}_{k+1}^{(p)}|\boldsymbol{e}_{1:k}, \boldsymbol{W})$.

Since vector $\dot{\boldsymbol{a}}_{k+1}^{(p)}$ should be sampled without $\boldsymbol{e}_{k+1}$, we implement the following steps to generate it:

Forward Sampling for particle p :

- Generate ${\boldsymbol{\theta}^m_{k+1}}^{(p)}$ from $\mathcal{RNN}^m(m_k^{(p)}, \boldsymbol{x}_k^{(p)}, \boldsymbol{h}_k^m | \boldsymbol{W}_{\boldsymbol{h}^m}, \boldsymbol{W}_{\boldsymbol{\theta}^m})$.
- Sample $m_{k+1}^{(p)}$ from probability distribution $\pi^m(\boldsymbol{\theta}^m_{k+1}{}^{(p)})$.
- Generate ${\boldsymbol{\theta}^x_{k+1}}^{(p)}$ from $\mathcal{RNN}^x(m_k^{(p)}, \boldsymbol{x}_k^{(p)}, \boldsymbol{h}_k^x | \boldsymbol{W}_{\boldsymbol{h}^x}, \boldsymbol{W}_{\boldsymbol{\theta}^x})$.
- Sample $\boldsymbol{x}_{k+1}^{(p)}$ from probability distribution $\pi^x(\boldsymbol{\theta}^x_{k+1}{}^{(p)})$.
- Generate ${\boldsymbol{\theta}^y_{k+1}}^{(p)}$ from $\mathcal{RNN}^y(m_{k+1}^{(p)}, \boldsymbol{x}_{k+1}^{(p)}, z_{k+1}^n, \boldsymbol{h}_k^y | \boldsymbol{W}_{\boldsymbol{h}^y}, \boldsymbol{W}_{\boldsymbol{\theta}^y})$.

Now the term $\Pr(\dot{\boldsymbol{a}}_{k+1}^{(p)}|\boldsymbol{e}_{1:k}, \boldsymbol{W})$ can be calculated as

$$\Pr(\dot{\boldsymbol{a}}_{k+1}^{(p)}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) = \pi^m(\dot{m}_{k+1}^{(p)}; \boldsymbol{\theta}^m_{k+1}{}^{(p)}) \times \pi^x(\dot{\boldsymbol{x}}_{k+1}^{(p)}; \boldsymbol{\theta}^x_{k+1}{}^{(p)}).$$

Given the conditional independence of $\dot{\boldsymbol{a}}_{k+1}^{(p)}$ and $\boldsymbol{e}_{1:k}$, the prediction measure can be simplified to

$$\Pr(\boldsymbol{y}_{k+1}|\boldsymbol{e}_{1:k}, \boldsymbol{W}) \approx \sum_{p=1}^{P} \pi^y\left(\boldsymbol{y}_{k+1}; \boldsymbol{\theta}^y_{k+1}{}^{(p)}\right) \pi^m\left(\dot{m}_{k+1}^{(p)}; \boldsymbol{\theta}^m_{k+1}{}^{(p)}\right) \times \pi^x\left(\dot{\boldsymbol{x}}_{k+1}^{(p)}; \boldsymbol{\theta}^x_{k+1}{}^{(p)}\right).$$

For a continuous feature j, the conditional variable $y_{k+1,j}$ follows a Gaussian mixture with P components

$$y_{k+1,j}|\boldsymbol{e}_{1:k} \sim \mathcal{N}({\mu^y_{k+1,j}}^{(l)}, {\sigma^y_{k+1,j}}^{(l)}), \quad \text{and} \quad l \sim \mathrm{Cat}(\phi_1, ..., \phi_P), \text{ where } \phi_p = \frac{\pi^m(\dot{m}_{k+1}^{(p)}; \boldsymbol{\theta}^m_{k+1}{}^{(p)}) \times \pi^x(\dot{\boldsymbol{x}}_{k+1}^{(p)}; \boldsymbol{\theta}^x_{k+1}{}^{(p)})}{\sum_{p=1}^{P} \pi^m(\dot{m}_{k+1}^{(p)}; \boldsymbol{\theta}^m_{k+1}{}^{(p)}) \times \pi^x(\dot{\boldsymbol{x}}_{k+1}^{(p)}; \boldsymbol{\theta}^x_{k+1}{}^{(p)})}.$$

Each particle p is a component of the mixture with mean ${\mu^y_{k+1,j}}^{(p)}$ and variance ${\sigma^y_{k+1,j}}^{(p)}$, which are the jth element and the jth diagonal element of $\boldsymbol{\mu}_k^y$ and $\boldsymbol{\Sigma}_k^y$ obtained for particle p. Now the marginal expected predicted value of $y_{k+1,j}$ can be found as follows: $\mathrm{E}(y_{k+1,j}|\boldsymbol{e}_{1:k}) = \sum_{p=1}^{P} {\mu^y_{k+1,j}}^{(p)} \phi_p$. The above Gaussian mixture can be used for the uncertainty evaluation of any variable estimate. For instance, we can also draw bootstrap samples from the corresponding Gaussian mixture to find a conditional prediction interval for any desired confidence level. Similar steps can be followed to obtain non-Gaussian mixture models for binary and categorical observations ((15) and (16)). A similar approach can be used for $t > 1$. In summary, once the structure of the stochastic neural networks is trained from the algorithm discussed in Section 5, one can track two types of key measures at any time point k, given the set of observations up to time k ($[\boldsymbol{y}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{o}_{1:k}]$). The first measure is the probability distribution of latent states at any time k ($\boldsymbol{x}_k, m_k$). The second measure is the t – *step* ahead prediction of the variables in the observable layers (e.g., $\boldsymbol{y}_{k+t}$). To find these measures, we need to utilize PF and forward sampling in the E-step to find the most likely sets of latent states' particles from time 1 to time k (and time k to time $k+t$ for prediction). These particles are used in Sections 5.1 and 5.2 to calculate the two types of measures discussed above.

6. Numerical experiments

In this section, we provide a set of numerical experiments using different datasets to discuss the application of the proposed framework in (i) modeling state-space frameworks without having prior knowledge regarding the stochastic behavior of the system and (ii) modeling the stochastic evolution of a degrading system and generating diagnostic and prediction insights. The RNN structures are discussed in Appendix 6.

A. Performance measures

The performance of our framework is analyzed for filtering and prediction tasks. For continuous variables, we use Root Mean Squared Error (RMSE) and the % coverage of the prediction interval. Accuracy is used for binary and categorical variables. For model training, we analyze the convergence of the loss function for each layer and discuss relevant insights for the learning phase for each dataset. These measures are very common in evaluating SSMs and degradation models. In all experiments and for all tasks, the only available data are the outputs of the observable layers. For the model training, we should make sure the structure of the neural networks and hyperparameters are fine-tuned before the steps in the EM algorithm are followed. Once the model is

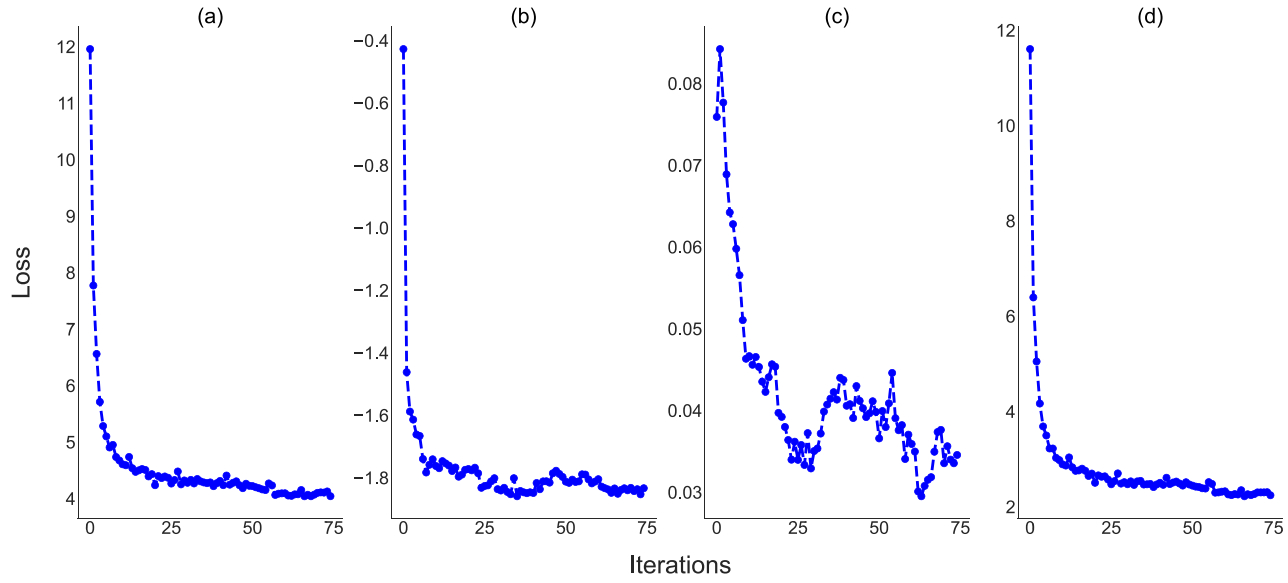

Figure 4. The improvement of the loss functions over iterations of the EM algorithm. (a)-(c) are for the loss functions for $\mathcal{RNN}^x$, $\mathcal{RNN}^y$, $\mathcal{RNN}^o$, and (d) is the direct sum of (a)-(c).

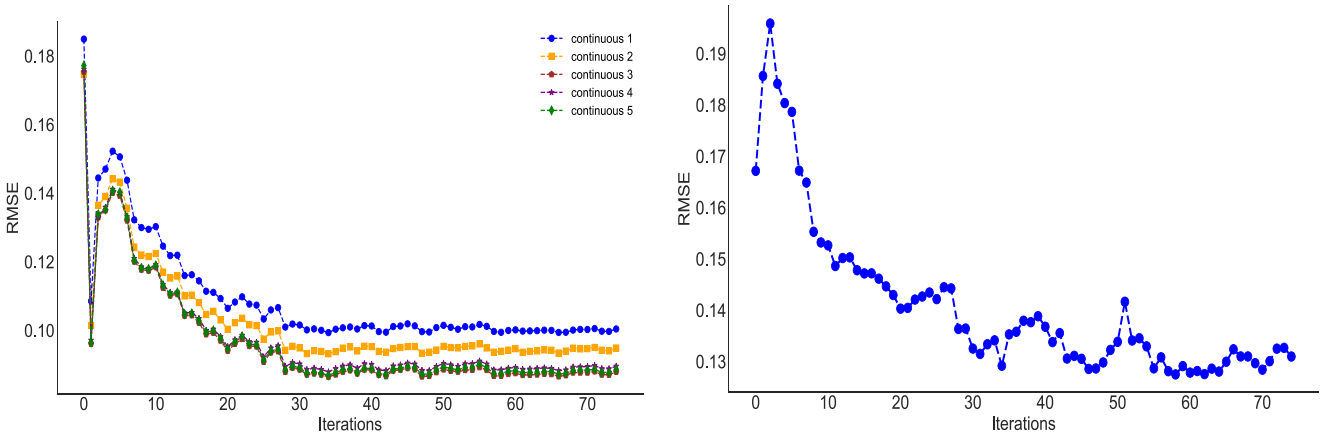

Figure 5. The improvement of the RMSE for (a) five continuous sensor outputs and (b) latent state *x*.

trained, we can use the results provided in Section 5 and the trained network for inference and prediction.

6.1. A simulated hybrid degrading system (dataset 1)

We consider a hybrid SSM to simulate multiple runs to failure time series for a degrading system. The stochastic layers at cycle k include a latent continuous degradation state process x_k, a latent binary discrete-state process m_k (i.e., faulty or normal), a latent hazard process λ_k, a 14-dimensional observation process $\boldsymbol{y}_k$ representing mixed-type sensor data (five dependent continuous sensors, four binary sensors, and five categorical sensors), and the binary working status process o_k representing the overall system's working conditions (i.e., working or failure). The structure of this system is summarized in Appendix 7.

A. Model training

At each step of the learning algorithm, the training model tries to optimize the model structures $\mathcal{RNN}^{m,x,y,o}$. To show how efficient the algorithm improves each model, the loss function values over EM iterations, and their direct sum are shown in Figure 4. It can be seen that while each model's loss value has some fluctuations, the overall loss has a decreasing trend and converges around 75 iterations. To better demonstrate the efficiency of the training algorithm, we also recorded the RMSE between the true and estimated continuous sensor values and the latent x values over the iterations of the EM algorithm. As shown in Figure 5, the RMSEs tend to decrease for all cases, which verifies that model structures are better trained after each iteration of the EM algorithm. For categorical and binary attributes, we recorded the accuracy of the sensor outputs after each iteration of the algorithm. Results in Figure 6 show that the overall accuracy of sensor outputs improves iteration by iteration. The relatively low accuracy for categorical sensor outputs is due to the nature of categorical sensor outputs that have multiple discrete outcomes. In summary, we can verify that the proposed framework is able to characterize the stochastic behavior of the system without using prior knowledge or parametric distribution after a reasonable number of iterations.

B. Filtering and prediction power

To show how well the trained framework can represent the evolution of the system over time, we evaluated its performance in terms of predicting sensor values and estimating latent state variables in the test set. In Figure 7, we show the estimated/true sensor values for five continuous sensor outputs and the corresponding 95% prediction intervals for a sample times series in the test set. Results in Figure 7 show that the estimated values are very close to true values and the prediction intervals cover the estimated values pretty well. We prepared a similar plot for the latent state x in Figure 8 for six test time series. We can observe from this figure that although the coverage for this latent state is relatively lower compared to the cases for sensor outputs, the trends of the estimated and true x are very similar and the differences become smaller as we approach the end of the lifetimes. Since x values are normalized in the simulation phase and are hidden in the training phase, there is no guarantee that latent state x is identifiable. Because of this, we only expect the estimates to be a reasonable representation of the degradation process. For the discrete latent state m, we plotted the estimated/true m for six sample time series in Figure 9. Results indicate the model's reasonable performance in detecting the true latent state m and the transition time from a normal state to

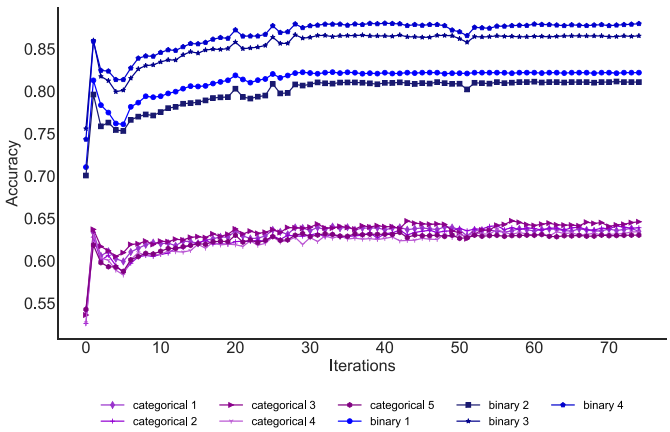

Figure 6. The improvement of accuracy for four binary and five categorical sensor outputs over EM iterations.

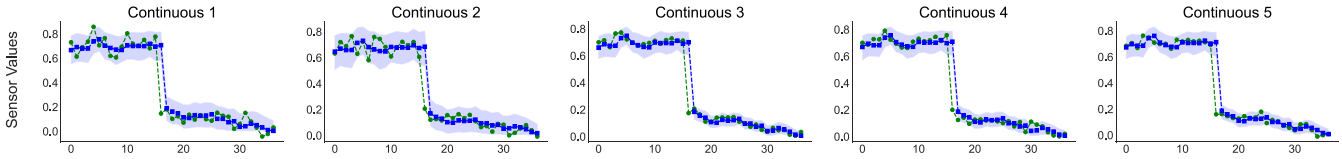

Figure 7. Comparison between true and estimated sensor values and the 95% prediction intervals.

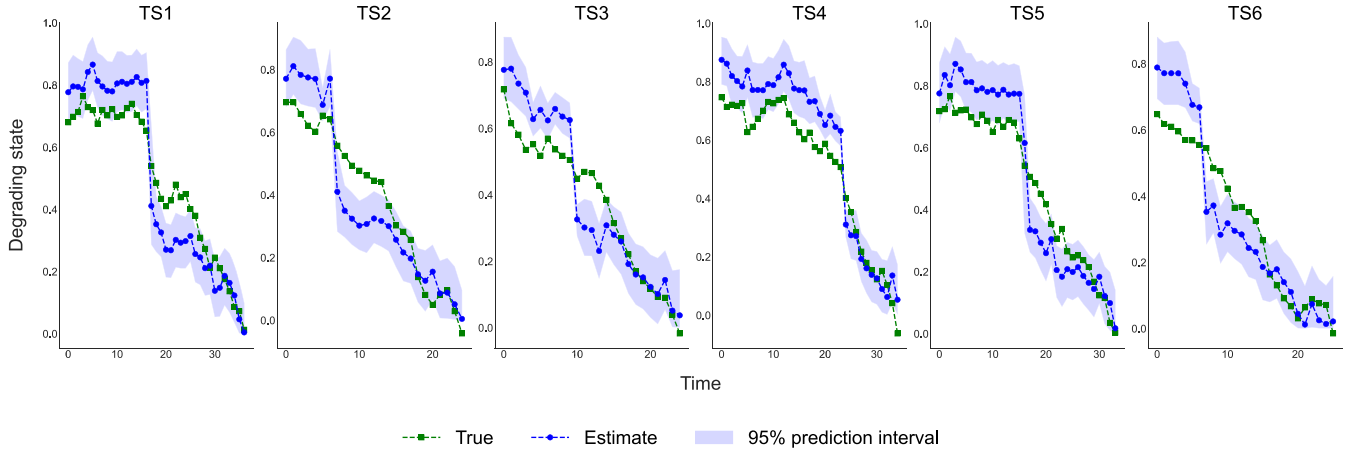

Figure 8. Comparison between true and estimated values of latent state x and the 95% prediction intervals.

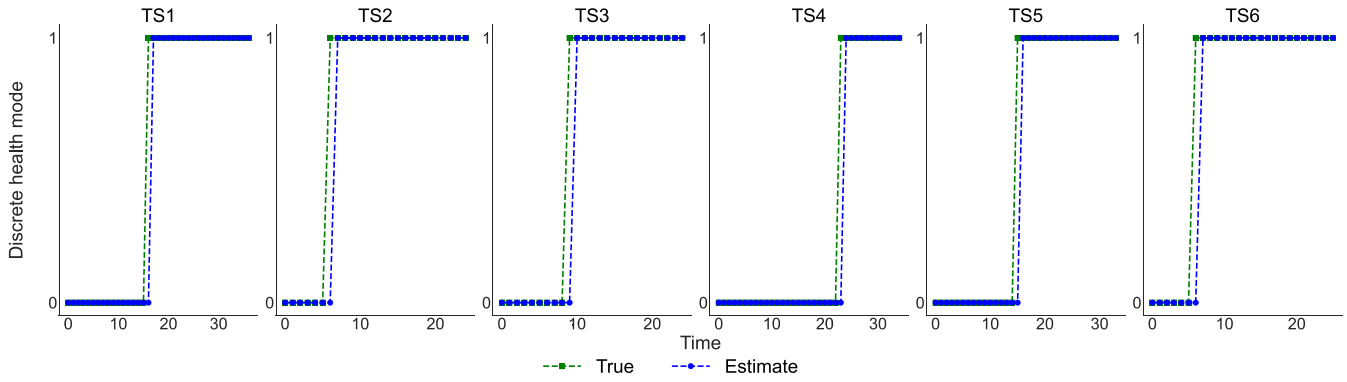

Figure 9. Comparison between the true and estimated values for the latent state m for six time series.

Table 2. The overall performance of the proposed model for different types of variables.

	RMSE - Continuous Sensors					*Accuracy (%) - Binary Sensors*			
Set	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
(Train,Test)	(0.1,0.109)	(0.095,0.1)	(0.088,0.094)	(0.09,0.096)	(0.089,0.096)	(82.2,83.1)	(81.1,82.4)	(86.5,86.3)	(88,87.6)
	Accuracy (%) - Categorical Sensors					*RMSE*		*Accuracy (%)*	
Set	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	latent state x		latent state m	
(Train,Test)	(63.5,65.2)	(63.9,63.5)	(64.6,61.8)	(63.4,61.3)	(63.0,60.8)	(0.131,0.128)		(94.1,95.5)	

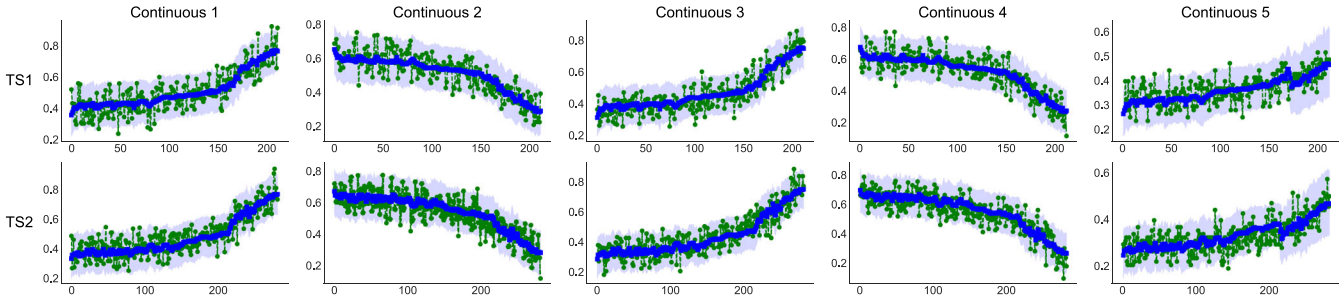

Figure 10. Sample CMAPSS true and estimated sensor values and their 95% prediction intervals in FD001.

a faulty state. A summary of the performance measures calculated for the entire test set is shown in Table 2. From the results, we can confirm the reasonable performance of the proposed framework in terms of predicting mixed-type sensor values, latent states x, and latent state m.

C. The effect of including the failure process

To show how including the failure process o can impact the performance of our model, we conducted the same experiment discussed earlier but excluded layer o from the system dynamics. We first calculated the errors of estimation for sensor values in the test set (RMSE for continuous outputs and accuracy for discrete outputs). Due to the page limit, we discuss the results in detail in Appendix 8. In summary, including model o helps with a better overall fit, particularly a better estimation of latent states and preventing overfitting. Without process o, there is no data in the model that can encourage a monotonic trend for the latent degradation state x. Including process o can also help with predictive maintenance tasks, such as the remaining useful life prediction (out of the scope of this paper).

6.2. The CMAPSS turbofan engine degradation dataset (Dataset 2)

The experiments in this section are based on a widely used CMAPSS turbofan engine degradation dataset provided by the NASA Ames Research Center (Saxena and Goebel, 2008). This dataset consists of multiple multivariate time series from a fleet of engines of the same type. Each time series starts with an unknown level of initial wear and manufacturing variation implying the randomness of the engine's degradation. In all engines, a fault initiates at some unknown point and grows until system failure. There are four datasets, referred to as FD001-FD004, with different numbers of operational settings and fault modes and 21 sensor measurements. In our experiments, we focused only on 14 non-constant sensor outputs for process $\boldsymbol{y_k}$. Datasets FD001-FD004 have 1, 6, 1, and 6 operational settings (i.e., $\boldsymbol{u_k} \in \{1, ..., 6\}$), respectively. Datasets FD001 and FD002 have one fault mode (HPC Degradation) whereas FD003 and FD004 have two modes (HPC Degradation and Fan Degradation). These fault modes are treated as part of the discrete latent state m_k (i.e., $m_k \in \{\text{Normal, HPC Degradation, Fan Degradation}\}$). We have considered 50 time series for training and another 50 time series for testing. Similar to the simulated dataset, we evaluate the performance of the model in terms of model training, filtering, and prediction. We have assumed a dynamic system based on Figure 2 with multiple layers of stochastic variables, that is, a latent discrete state (m), a latent continuous state (x), a multidimensional observation process (y), and a failure process (o). We also included operational settings as inputs for all neural networks. We conducted hyperparameter-tuning with grid search. Due to the page limit, we present some results in the Appendices. All categorical features (discrete latent states and discrete sensor outputs) are processed through one-hot encoding when used as inputs in the neural network models.

A. Model training

To check whether the proposed framework can train the structure of the engine degradation systems using historical data, we monitored the change in (i) the loss functions for $\mathcal{RNN}^x$, $\mathcal{RNN}^y$, $\mathcal{RNN}^o$, and total loss and (ii) the change in the RMSEs of estimated sensor values. Due to the similarity of the results of our experiments to the ones presented in Section 6.1.A and also due to the page limit, we discussed the results for the model training in Appendix 9. We should point out the results for FD001-FD004 were very similar in terms of model training convergence.

B. Prediction power for sensor observations

Since for this dataset, latent states are not known, we discuss the prediction power of the model only for observable

Table 3. Estimation error and prediction interval coverage for sensor outputs in datasets FD001-FD004.

Dataset	*Measure*	*Sensor Measurements*													
		y_2	y_3	y_4	y_7	y_8	y_9	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{17}	y_{20}	y_{21}
FD001	RMSE (Train)	0.092	0.088	0.069	0.067	0.050	0.056	0.061	0.066	0.049	0.057	0.078	0.081	0.079	0.083
	RMSE (Test)	0.092	0.089	0.071	0.068	0.052	0.059	0.064	0.067	0.051	0.059	0.079	0.081	0.080	0.085
	95% PI Coverage (Train)	94.8	95.0	95.0	94.7	95.1	95.1	95.0	95.3	95.0	95.2	94.8	94.5	95.0	95.1
	95% PI Coverage (Test)	94.7	94.6	94.4	95.4	94.2	94.3	94.3	94.7	84.6	94.4	95.4	94.2	94.3	94.3
FD002	RMSE (Train)	0.0029	0.0111	0.0100	0.0010	0.0005	0.0127	0.0088	0.0008	0.0382	0.0339	0.0079	0.0103	0.0035	0.0035
	RMSE (Test)	0.0029	0.0112	0.0102	0.0010	0.0005	0.0114	0.0089	0.0008	0.0390	0.0301	0.0080	0.0103	0.0035	0.0035
	95% PI Coverage (Train)	95.4	95.0	94.6	94.2	90.8	96.1	94.6	93.8	80.5	95.9	94.4	95.0	94.9	94.7
	95% PI Coverage (Test)	95.1	94.7	94.2	94.2	88.6	95.8	94.6	93.4	81.2	96.6	94.3	94.8	94.8	94.8
FD003	RMSE (Train)	0.0720	0.0811	0.0684	0.0316	0.0326	0.0492	0.0663	0.0292	0.0332	0.0477	0.0524	0.0907	0.0650	0.0589
	RMSE (Test)	0.0742	0.0817	0.0683	0.0300	0.0318	0.0548	0.0679	0.0273	0.0321	0.0538	0.0528	0.0898	0.0640	0.0602
	95% PI Coverage (Train)	94.9	95.1	94.8	94.6	94.9	95.7	94.6	94.5	94.9	95.3	94.6	94.9	95.0	94.8
	95% PI Coverage (Test)	94.2	94.8	94.7	93.5	93.3	94.0	94.1	93.3	94.1	93.8	94.5	95.3	95.4	94.2
FD004	RMSE (Train)	0.0031	0.0115	0.0111	0.0033	0.0009	0.0111	0.0103	0.0033	0.0005	0.0312	0.0175	0.0107	0.0048	0.0047
	RMSE (Test)	0.0032	0.0116	0.0114	0.0031	0.0009	0.0102	0.0106	0.0032	0.0004	0.0282	0.0175	0.0108	0.0048	0.0047
	95% PI Coverage (Train)	93.5	93.9	94.6	95.5	38.6	96.0	95.0	95.4	95.6	96.9	95.9	93.9	95.2	95.5
	95% PI Coverage (Test)	92.5	93.7	93.9	96.1	41.7	96.2	94.0	95.9	96.3	97.0	96.5	93.1	95.0	95.5

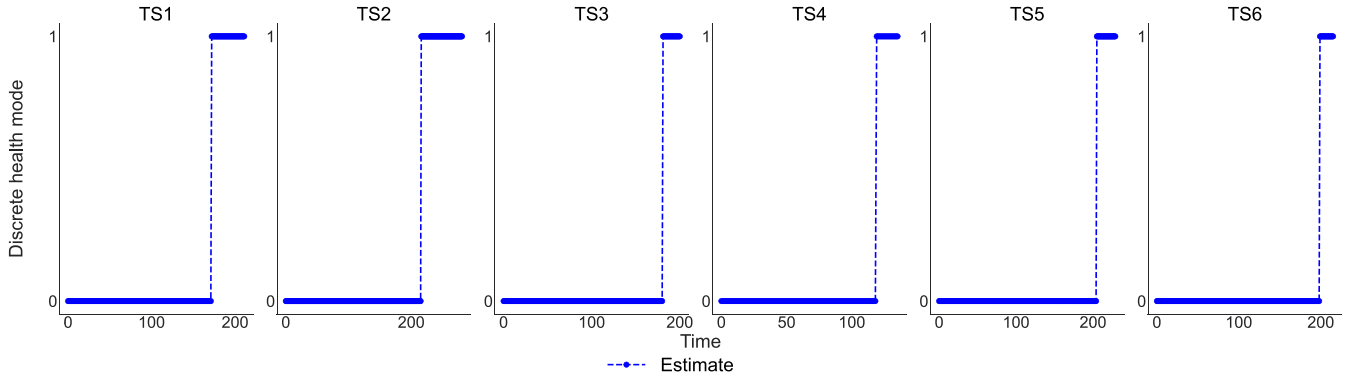

Figure 11. The estimated values of the discrete latent states for six samples in FD001.

variables. To do so, we first show two sample time series (TS1-TS2) with true and estimated sensor values and their corresponding 95% prediction intervals in Figure 10 for FD001. Similar results are provided for FD002-FD004 in Appendix 10. In Table 3, we summarized the RMSE values for all 14 sensor outputs for both train and test sets in FD001-FD004. From Figure 10 and Table 3, we can conclude that the model performs (and generalize) well in terms of predicting sensor values. Also, it is clear that the errors are smaller for some sensor outputs, but overall, almost all errors are within a very low range. This is a significant result because a model is trained that can mimic the stochastic behavior of all sensors without imposing prior assumptions (distributional or parametric) about the system dynamics and its stochastic behavior. Also, we can see from Table 3 that the coverage probability of prediction intervals (PI) is very close to the nominal confidence of 95%. It should be pointed out that the sensor values for FD002 and FD004 are very sensitive to operational settings and they are relatively easier to predict. That is why the true and estimated values are very close to each other (see Table 3) and confidence intervals (see Figures 5-7 in Appendix 10) are very narrow.

C. Insights for latent variables

There are two types of latent state variables (the discrete latent state and the continuous latent state). In Figure 11, we show the estimated values for the discrete state (m) for six sample time series in FD001. It can be seen that the change in the discrete binary state occurs toward the end of the engine lifetimes. This observation is very consistent across all engines in the train and test sets in FD001, which is subject to one fault mode. We also evaluated the behavior of the latent state x (normalized between zero and one) for all test time series in FD001. As shown in Figure 12, the latent state has a monotonic decreasing trend with minor fluctuations. For engines with smaller lifetimes, the degradation is faster (see for instance TS4), which is compatible with our expectation of latent degradation in such systems. We also found that for all engines, the failure occurs when this latent degradation state is close to 0.1. This is an important finding, particularly for decision making and preventive maintenance. We also checked the behavior of the latent hazard function and found consistent and reasonable trends for all engines. Due to the page limit, more results and discussions, including for FD002-FD004, are given in Appendix 11. Results in the Appendix also help with understating the usefulness of defining the failure process (o) and the overall interpretability of the results for FD001-FD004, particularly with respect to the monotonic (decreasing) behavior of the continuous latent state and the increasing trend of the log-hazard probability values.

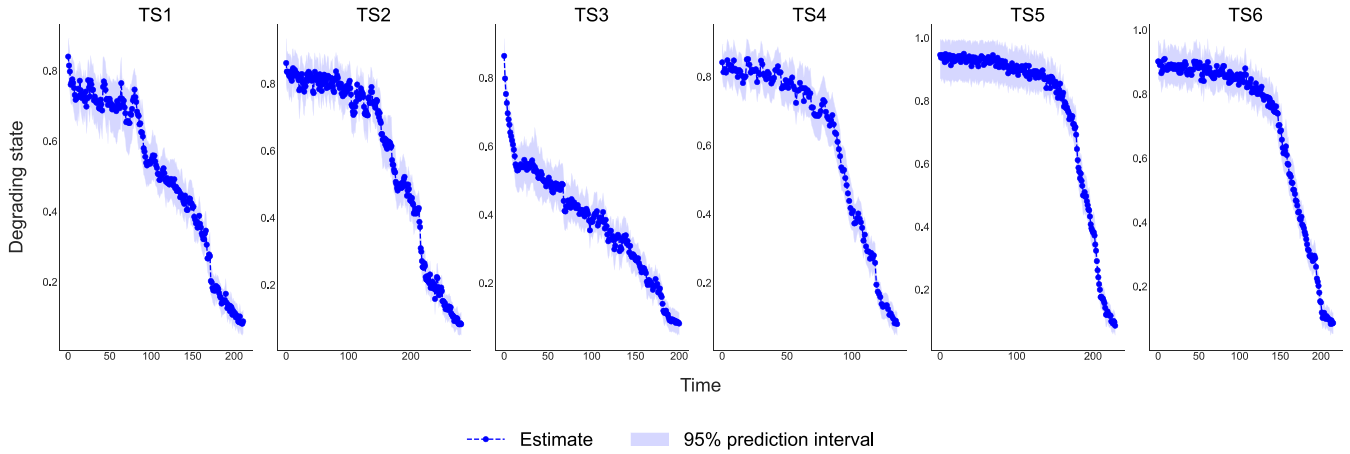

Figure 12. The estimated values of the continuous latent state x for 6 samples in FD001.

Table 4. Comparison with variational inference-based hybrid SSMs over five experiment runs.

		Benchmark Models		*Proposed Work*						
	Measure	D3SN	SNLDS	Run 1	Run 2	Run 3	Run 4	Run 5	Average	STDEV
Forecasting	RMSE for y	15.142 ± 1.084	16.541 ± 0.024	14.4707	13.2252	12.3841	17.1151	14.1583	14.27068	1.788
	duration of $m=0$	8.211 ± 2.717	1.282 ± 0.001	6.79	20.5	10.0	9.14	11.04	11.49	5.273
	duration of $m=1$	8.296 ± 2.624	1.925 ± 0.012	10.41	21.08	19.94	9.0	11.64	14.41	5.657
	Accuracy for m	0.777 ± 0.028	0.543 ± 0.001	0.7916	0.8597	0.8176	0.8257	0.8457	0.82806	0.026
	F1 score	0.768 ± 0.022	0.549 ± 0.001	0.8213	0.8659	0.8602	0.8036	0.8451	0.83922	0.026
Filtering	Accuracy for m	0.843 ± 0.017	0.692 ± 0.003	0.824	0.888	0.836	0.868	0.86	0.8552	0.025
	F1 score	0.821 ± 0.025	0.544 ± 0.002	0.8508	0.8923	0.8734	0.8429	0.8577	0.86342	0.02

6.3. Comparison with similar models

The proposed framework is recommended for systems that can be modeled through an SSM structure when no prior information is available regarding the parametric form or distributions of latent and observable layers of the system dynamics. Although the framework discussed in this article is not developed to replace existing SSMs, its competitiveness compared to similar models is numerically evaluated.

6.3.1. Neural network-based nonlinear switching SSMs

We compare our work with the results provided in Xiuqin and Ying (2021) (referred to as DS3M), which also includes results from Dong *et al.* (2020) (referred to as SNLDS). Both of these references utilize neural networks for a hybrid SSM with discrete and continuous latent states. These methods have been shown to outperform a few other SSM-based time series methods (Xiuqin and Ying, 2021). We chose these models because they are nonlinear, can incorporate discrete and continuous latent states, and their structures are based on neural networks. Both DS3M and SNLDS have used a variational inference approach for training and inference. For a better comparison, we used the same toy example structure given in Xiuqin and Ying (2021) and compared our model with the results shown in that paper. First, we simulated time series of length 2000 cycles for five independent runs from a nonlinear hybrid SSM given in Appendix 12. Similar to Xiuqin and Ying (2021), we considered 1000, 500, and 500 samples for training, validation, and testing. Results are shown in Table 4 for five experimental runs of our model, which includes the one-step-ahead prediction of y (represented by RMSE), one-step-ahead prediction of m (represented by accuracy and F1 score), and filtering of discrete state m (represented by accuracy and RMSE). We also report the mean duration at each level of discrete state m given the theoretical mean of 20 for both $m=0$ and $m=1$ (i.e., 1/0.05). It can be seen from Table 4 that our model performs slightly better than both DS3N and SNLDS in terms of forecasting and filtering for both m and y. However, our results seem to have higher variances, which could be due to the stochastic nature of PF and randomness in the expectation step of the training phase. One potential solution is to increase the number of particles or change the stopping criterion of the algorithm for a better convergence. Also, our model is more flexible as it considers one RNN structure for each layer of system dynamics. Both DS3N and SNLDS marginalize the discrete latent variable which can lead to bias when the discrete state has a complex impact on other layers. In other words, the effective reparameterization of the posterior $q(m_{1:T}, x_{1:T}|\boldsymbol{y}_{1:T})$ (even without the layer of failure process) remains a challenge in variational inference-based works, such as Dong *et al.* (2020) and Xiuqin and Ying (2021).

6.3.2. Deep nonlinear SSMs

Neural networks and deep learning have been used for system identification to characterize the stochastic evolution of typical dynamic systems with only a latent state (x) and an observation process (y). Here, we compare our work with the results in Gedon *et al.* (2021) (referred to as VAE-RNN), which is a deep learning model, that combines RNNs with variational autoencoder (VAE) to characterize

nonlinear SSMs. We also compare our model with results provided in Gedon *et al.* (2021) for other variations of VAE-RNN, referred to as variational RNN (VRNN) (Chung *et al.*, 2015), and stochastic RNN, referred to as STORN. We used the Narendra-Li Benchmark dataset used in Gedon *et al.* (2021), which is a highly nonlinear system with two layers of dependent hidden states, a 1-dimensional observation process y_k, and a known control input of u_k at time k. We discuss the results in Appendix 13.

7. Conclusions and future work

This article introduces a new framework for modeling degrading systems through a state-space structure that includes multiple layers of latent, observable, and controllable variables and inputs/outputs. The framework has generic characteristics that are inspired by BHM and recurrent neural networks. The temporal behavior of system dynamics and the relationship between variables are characterized by stochastic neural networks and trained with data. Numerical experiments demonstrated the application of the proposed framework on modeling state-space degrading systems without having prior knowledge regarding the stochastic behavior of such systems and on modeling the stochastic evolution of a degrading system and generating diagnostic and prediction insights. The framework developed in this article has two main limitations. First, it requires a larger amount of training data compared with fully parametric models. Second, the model training phase is an iterative process that may converge slowly partially due to the stochastic nature of system's inputs and outputs and the quality of the initialized solution. In future, we will study how parallel computing and vectorization can be utilized within the Bayesian structure of the system to speed up the learning process and to develop a training framework that can work with limited data. Also, we will explore the use of the proposed framework for predictive analytics (mainly for remaining useful life prediction).

Funding

This material is based upon work supported by the National Science Foundation under Grant No. (1846975).

References

Banjevic, D. and Jardine, A. (2006) Calculation of reliability function and remaining useful life for a Markov failure time process. *IMA Journal of Management Mathematics*, **17**(2), 115–130.

Bao, Y., Velni, J., Basina, A. and Shahbakhti, M. (2020) Identification of state-space linear parameter-varying models using artificial neural networks. *IFAC-PapersOnLine*, **53**(2), 5286–5291.

Bao, Y., Velni, J. and Shahbakhti, M. (2021) Epistemic uncertainty quantification in state-space LPV model identification using Bayesian neural networks. *IEEE Control Systems Letters*, **5**(2), 719–724.

Benidris, M., Elsaiah, S. and Mitra, J. (2015) Power system reliability evaluation using a state space classification technique and particle swarm optimisation search method. *IET Generation, Transmission and Distribution*, **9**(14), 1865–1873.

Blom, H.A. and Bloem, E. (2007) Exact Bayesian and particle filtering of stochastic hybrid systems. *IEEE Transactions on Aerospace and Electronic Systems*, **43**(1), 55–70.

Che, Z., Purushotham, S., Cho, K., Sontag, D. and Liu, Y. (2018) Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, **8**, 1–12.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A.C. and Bengio, Y. (2015) A Recurrent Latent Variable Model for Sequential Data, in C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, Volume 28. Curran Associates, Inc.

Dong, Z., Seybold, B., Murphy, K. and Bui, H. (2020, 13–18 July) Collapsed amortized variational inference for switching nonlinear dynamical systems, in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Virtual, pp. 2638–2647.

Dourado, A. and Viana, F.A. (2020) Physics-informed neural networks for missing physics estimation in cumulative damage models: A case study in corrosion fatigue. *Journal of Computing and Information Science in Engineering*, **20**(6), 1–15.

Gedon, D., Wahlström, N., Schön, T.B. and Ljung, L. (2021) Deep state space models for nonlinear system identification. *arXiv*, 2003.14162.

Hu, Y., Baraldi, P., Di Maio, F. and Zio, E. (2015) A particle filtering and kernel smoothing-based approach for new design component prognostics. *Reliability Engineering & System Safety*, **134**, 19–31.

Kantas, N., Doucet, A., Singh, S., Maciejowski, J. and Chopin, N. (2015) On particle methods for parameter estimation in state-space models. *Statistical Science*, **30**(3), 328–351.

Krieg, M., Science, D., (Australia), Electronics, T.O. and Laboratory, S.R. (2001) *A tutorial on Bayesian belief networks/Mark L Krieg. DSTO Electronics and Surveillance Research Laboratory, Surveillance Systems Division Edinburgh, S. Aust.*

Long, J., Chen, C., Liu, Z., Guo, J. and Chen, W. (2022) Stochastic hybrid system approach to task-orientated remaining useful life prediction under time-varying operating conditions. *Reliability Engineering and System Safety*, **225**, 108568.

Meeker, W. and Hong, Y. (2014) Reliability meets big data: Opportunities and challenges. *Quality Engineering*, **26**(1), 102–116.

Nascimento, R.G., Corbetta, M., Kulkarni, C.S. and Viana, F.A. (2021) Hybrid physics-informed neural networks for lithium-ion battery modeling and prognosis. *Journal of Power Sources*, **513**, 1–13.

Nascimento, R.G. and Viana, F.A.C. (2020) Cumulative damage modeling with recurrent neural networks. *AIAA Journal*, **58**(12), 5459 – 5471.

Orchard, M. and Vachtsevanos, G. (2009) A particle-filtering approach for on-line fault diagnosis and failure prognosis. *Transactions of the Institute of Measurement and Control*, **31**(3–4), 221–246.

Peng, Y., Wang, Y. and Zi, Y. (2018) Switching state-space degradation model with recursive filter for prognostics of remaining useful life. *IEEE Transactions on Industrial Informatics*, **15**, 822–832.

Rigamonti, M., Baraldi, P., Astigarraga, D. and Galarza, A. (2016) Particle filter-based prognostics for an electrolytic capacitor working in variable operating conditions. *IEEE Transactions on Power Electronics*, **31**(2), 1567–1575.

Saxena, A. and Goebel, K. (2008) Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA.

Schön, T., Wills, A. and Ninness, B. (2011) System identification of nonlinear state-space models. *Automatica*, **47**(1), 39–49.

Tafazoli, S. and Sun, X. (2006) Hybrid system state tracking and fault detection using particle filters. *IEEE Transactions on Control Systems Technology*, **14**(6), 1078–1087.

Wang, H., Huang, H., Li, Y. and Yang, Y. (2016) Condition-based maintenance with scheduling threshold and maintenance threshold. *IEEE Transactions on Reliability*, **65**(2), 513–524.

Wang, K., Chen, J., Song, Z., Wang, Y. and Yang, C. (2021) Deep neural network-embedded stochastic nonlinear state-space models and their applications to process monitoring. *IEEE Transactions on Neural Networks and Learning Systems*, **1**, 1–13.

Wei, T., Huang, Y. and Chen, C.L.P. (2009) Adaptive sensor fault detection and identification using particle filter algorithms. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, **39**(2), 201–213.

Xiuqin, X. and Ying, C. (2021) Deep switching state space model DS3M for nonlinear time series forecasting with regime switching. *CoRR abs/2106*.02329.

Yang, Y. and Shi, Y. (2019) State and parameter estimation algorithm for state space model based on linear neural network and Kalman filter, in *Proceedings of 2019 IEEE International Conference on Mechatronics and Automation, ICMA 2019*, IEEE Press, Piscataway, NJ, PP. 2314–2318.

Zhang, C., Chen, N. and Li, Z. (2017) State space modeling of autocorrelated multivariate Poisson counts. *IISE Transactions*, **49**(5), 518–531.

Zhang, Y., Tu, L., Xue, Z., Li, S., Tian, L. and Zheng, X. (2022) Weight optimized unscented Kalman filter for degradation trend prediction of lithium-ion battery with error compensation strategy. *Energy*, **251**, 123890.

Zhao, Y., Gao, W. and Smidts, C. (2021) Sequential Bayesian inference of transition rates in the hidden Markov model for multi-state system degradation. *Reliability Engineering and System Safety*, **214**, 107662.

Zheng, H., Kong, X., Xu, H. and Yang, J. (2021) Reliability analysis of products based on proportional hazard model with degradation trend and environmental factor. *Reliability Engineering and System Safety*, **216**, 107964.