Mechanical Systems and Signal Processing 164 (2022) 108213

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

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

Thresholdless Classification of chaotic dynamics and combustion instability via probabilistic finite state automata

Chandrachur Bhattacharya [a], Asok Ray [b,*]

[a] Departments of Mechanical Engineering and Electrical Engineering, Pennsylvania State University, University Park, PA, United States of America

[b] Departments of Mechanical Engineering and Mathematics, Pennsylvania State University, University Park, PA, United States of America

ARTICLE INFO

Communicated by J. Antoni

Keywords:
Symbolic time-series analysis
Probabilistic finite state automata
Hidden Markov model
Neural networks
Chaotic systems

ABSTRACT

The objective of the work reported in this paper is to make decisions on the current state of a dynamical system for pattern classification and anomaly/fault detection, which is often achieved by time series analyses of pertinent measured signals. In this context, one of the most commonly used methods is hidden Markov model (*HMM*), while yet another popular method is neural networks (*NN*) in their various configurations; however, both of these methods may require large training data and computational time. An alternative feasible method is probabilistic finite state automata (*PFSA*), which is much faster for training and also for testing. In its current state-of-the-art, the standard *PFSA*, called *s-PFSA*, has certain shortcomings that this paper attempts to remedy. Therefore, *s-PFSA* is modified into the proposed projection-based *PFSA*, abbreviated as *p-PFSA*, to yield better classification accuracy and robustness. Efficacy of *p-PFSA* is first demonstrated on four different models of chaotic dynamical systems by comparison with *s-PFSA*, *HMM* and *NN*, which are used to serve as baseline methods for validation of classification performance; the *NN* models consist of two vanilla *NNs* and another *NN* with long short term memory (*LSTM*). Then, these results of comparison are extended to assess the relative performance of *p-PFSA* for a real-life application in terms of accuracy, robustness, and computational complexity on a laboratory-scale apparatus that emulates the essential characteristics of industrial-scale combustion systems.

1. Introduction

Data-driven pattern classification of time series has seen a large surge recently, with several standard methods of machine learning (ML) being available [1]. One of the most common methods is hidden Markov model (*HMM*) [2] that has been used in many applications including speech recognition, image classification, hand gesture recognition, and DNA sequence extraction. Recently, Bhattacharya and Ray [3] have reported the usage of this framework in an attempt to solve the difficult (and largely unexplored) problem of classifying multiple regimes in chaotic dynamical systems. However, it is a fact that the architecture of *HMM* slows down both the training and testing processes due to complex and (sometimes) iterative mathematical operations.

Another popular method for time-series classification is neural networks (*NN*) in their various configurations, such as deep neural networks (*DNN*) [4], recurrent neural networks (*RNN*) [5], or neural networks with long short term memory (*LSTM*) layers [6], which have been used for text and audio analysis, among many other applications. Nevertheless, many of these *NN* techniques may

* Corresponding author.
E-mail addresses: chandrachur.bhattacharya@gmail.com (C. Bhattacharya), axr2@psu.edu (A. Ray).

https://doi.org/10.1016/j.ymssp.2021.108213
Received 15 September 2020; Received in revised form 21 June 2021; Accepted 4 July 2021
Available online 28 July 2021

not be suitable for online detection in dynamical systems that are restricted to use short lengths of time-series, or if there is no sufficient training due to the substantial data resources that *NNs* may need for training.

An alternative feasible method is probabilistic finite state automata (*PFSA*) [7,8], which requires much less training time and data resources, compared to both *HMM* and *NN*. The underlying principle of *PFSA* is built upon the concepts of reduced-order Markov modeling in the setting of symbolic time series analysis (STSA) [9–12], which has been used for pattern classification in diverse applications (e.g., combustion instabilities [13] and failure prognosis of structural materials [14]), as well as in sensor networks for detection of moving targets [15]. However, the standard *PFSA* method [7,8], called *s-PFSA*, needs to be modified to have its classification accuracy to be comparable to that of *HMM* and *NN*.

The current paper proposes major modifications to the architecture and algorithms of *s-PFSA* by alleviating some of its shortcomings in order to significantly improve its performance (e.g., classification accuracy and robustness). To demonstrate the efficacy of proposed modifications, the authors first address a difficult problem of classifying different regimes in chaotic dynamical systems from their respective time-series [3].

In general, chaos has been studied in great details, having been initially observed in several natural phenomena like weather fluctuations and climate changes [16], but now chaos is seen to exist in many human-engineered dynamical systems such as urban traffic [17] and the stock market [18]. Since classification of data from chaotic systems is often very difficult, the results of the proposed classification method have been compared to those of *HMM*-based classification presented in [3] as a baseline. Boulle et al. [19] have recently reported similar type of work in the framework of a neural network (*NN*), where the prime objective is to discriminate between chaotic and non-chaotic signals in a data-driven fashion by using a machine learning version of the well-known 0–1 test [20]. In order to provide further comparison with *NN* architectures, results using two vanilla *NN* models as well as a *NN* with a *LSTM* layer have also been compared in the current paper. While the reported work has been restricted to shallow nets, its extension to *DNN* and their various configurations is suggested as a topic of future research in Section 7.

The efficacy of *p-PFSA* for anomaly classification in real-life physical systems was previously demonstrated for fatigue crack detection [21] by using a thresholded approach, whereas the current paper addresses thresholdless classification. The underlying algorithms are validated on different experimental data from a laboratory-scale apparatus [22] that emulates the essential characteristics of industrial-scale combustion systems. In these data sets, there are several pressure-wave time series corresponding to: (i) stable operation (typically observed as low-amplitude chaotic signals); and (ii) unstable operation (seen as high-amplitude ordered sinusoidal signals). The classification of these two types of signals has been widely addressed in several ways and with different combustion data (e.g., [23]). In essence, a real-life application of the proposed thresholdless classification has been reported here.

Contributions: Major contributions of the current paper are summarized below:

(1) *Extensions of the standard PFSA (s-PFSA)*: The inherent shortcomings of *s-PFSA* are first identified; then, modifications are proposed to alleviate them to a large extent. Based on the algebraic and geometric properties of state transition probability matrices, accuracy and robustness of classification are significantly enhanced, especially for operating regimes that are difficult to discriminate.
(2) *Classification of chaotic signals*: The proposed *p-PFSA* is compared with other standard classification methods (i.e., *s-PFSA*, *HMM* and *NN*) on data sets of signals generated from four different chaotic dynamical systems, which have been rarely attempted because of the difficulties to classify such systems.
(3) *Regime classification related to combustion stability*: This is a real-life engineering problem [23,24], where the efficacy of *p-PFSA* is demonstrated by comparison with *s-PFSA*, *HMM* and *NN* on experimental data, collected from a combustor apparatus [22].

Organization: The paper is organized in seven sections, including the present one. Section 2 briefly introduces the basic concepts of various data analysis methods that are used later. Section 3 formulates the proposed algorithm of the *PFSA*-based thresholdless classification of operating regimes in dynamical systems. Section 4 modifies the procedure for normalization of raw time series data prior to partitioning and symbolization. Section 5 describes generation of two types of data for algorithm validation: (i) synthetic data from four standard models of chaotic dynamical systems, and (ii) experimental data of pressure-wave time-series from a combustor apparatus. Section 6 validates the results along with discussions for both data sets. Section 7 summarizes and concludes the paper with recommendations for future research.

2. Background and mathematical theory

This section provides the basic concepts of probabilistic finite state automata (*PFSA*), hidden Markov model (*HMM*), and neural networks (*NN*) to provide the necessary backgrounds for the concepts presented in the sequel.

2.1. Introduction to probabilistic finite state automata

In the setting of symbolic dynamics [25] for analysis of time series signals, the observed data are initially partitioned (quantized) into a finite number of cells. Then, the time series is symbolized, where the symbol indicates the cell in which the value of the continuous-signal data-point lies. This process converts the (continuous-valued) time series into a symbol string, where each symbol in the string belongs to a (finite-cardinality) alphabet [26,27]. (It is noted that the cardinality of an alphabet is equal to the number of cells used for quantization.) The final step in this process is the construction of the probabilistic finite state automaton (*PFSA*). The following definitions are introduced toward this goal.

Definition 1. A finite state automaton (FSA) G, having a deterministic algebraic structure, is a triple $(\mathcal{A}, Q, \delta)$ where:

- $\mathcal{A}$ is a (nonempty) finite alphabet, i.e., its cardinality $|\mathcal{A}|$ is a positive integer.
- Q is a (nonempty) finite set of states, i.e., its cardinality $|Q|$ is a positive integer.
- $\delta : Q \times \mathcal{A} \to Q$ is a deterministic state transition map.

Definition 2. A symbol block, also called a word, is a finite-length string of symbols belonging to the alphabet $\mathcal{A}$, where the length of a word $w \triangleq s_1 s_2 \cdots s_\ell$ with every $s_i \in \mathcal{A}$ is $|w| = \ell$, and the length of the empty word ϵ is $|\epsilon| = 0$. The parameters of FSA are extended as:

- The set of all words, constructed from the symbols in $\mathcal{A}$ and including the empty word ϵ, is denoted as $\mathcal{A}^\star$.
- The set of all words, whose suffix (respectively, prefix) is the word w, is denoted as $\mathcal{A}^\star w$ (respectively, $w\mathcal{A}^\star$).
- The set of all words of (finite) length ℓ, where ℓ is a positive integer, is denoted as $\mathcal{A}^\ell$.

Remark 3. A symbol string (or word) is generated from a (finite-length) time series by symbolization.

Definition 4. A probabilistic finite state automaton (PFSA) J is a pair (G, π), where:

- The deterministic FSA G is called the *underlying FSA* of the PFSA J.
- The probability map $\pi : Q \times \mathcal{A} \to [0, 1]$ is called the morph function (also known as symbol generation probability function) that satisfies the condition: $\sum_{s\in\mathcal{A}} \pi(q, s) = 1$ for each $q \in Q$. The map π can be represented by a $|Q| \times |\mathcal{A}|$ stochastic matrix Π (i.e., each element of Π is non-negative and each row sum of Π is unity). In that case, the PFSA is a quadruple $J = (\mathcal{A}, Q, \delta, \Pi)$.
- The state transition probability mass function $\kappa : Q \times Q \to [0, 1]$ is constructed by combining δ and Π, which can be structured as a $|Q| \times |Q|$ state transition probability matrix T. In that case, the PFSA can also be described as the triple $J = (\mathcal{A}, Q, T)$.

Remark 5. It is noted that, prior to partitioning, the time series is typically normalized to have zero-mean and unity-variance to remove bias from the signal and to ensure that a fixed set of partition boundaries can be used across a wide range of data. A fixed partitioning ensures that there is no need to recompute the partitioning boundaries at every step and also allows different PFSAs to be compared, which is essential for classification. Details are provided in Section 4.

2.2. D-Markov machines

The PFSA structure of a D-Markov machine generates symbol strings $\{s_1 s_2 \cdots s_\ell : \ell \in \{1, 2, 3, \ldots\}$ and $s_j \in \mathcal{A}\}$ based on the underlying Markov process. In a D-Markov machine, it is assumed that generation of the next symbol depends only on a *finite* history of last D or less consecutive symbols, i.e., the (most recent) symbol block of length not exceeding D. A D-Markov machine [8] is defined as follows.

Definition 6. A D-Markov machine [7,8] is a PFSA in the sense of Definition 4 and it generates symbols that solely depend on the (most recent) history of at most D consecutive symbols, where the positive integer D is called the *depth* of the machine. Equivalently, a D-Markov machine is a statistically stationary stochastic process $S = \cdots s_{-1} s_0 s_1 \cdots$, where the probability of occurrence of a new symbol depends only on the last consecutive (at most) D symbols, i.e.,

$$P[s_n \mid \cdots s_{n-D} \cdots s_{n-1}] = P[s_n \mid s_{n-D} \cdots s_{n-1}]$$

Consequently, for $w \in \mathcal{A}^D$ (see Definition 2), the equivalence class $\mathcal{A}^\star w$ of all (finite-length) words, whose suffix is w, is qualified to be a D-Markov state that can be denoted as w.

Remark 7. If the depth of the D-Markov machine is unity (i.e., $D = 1$), then the state set Q and symbol alphabet $\mathcal{A}$ become equivalent; therefore, the morph matrix Π and the state transition probability matrix T are also the same if $D = 1$. In that case, equiprobability of every state (i.e., uniform probability distribution of the states) is ensured if maximum entropy partitioning (MEP) of time series [26] is used for symbol generation. Thus, at least for the base regime (and possibly even others) the concept of making them measure preserving [14,28] is being imposed. However, that restriction is not needed for the current work, making the proposed method more versatile than the method using measure preserving transformations [14,28].

The procedure to generate the morph matrix Π from a (finite-length) symbol string is as follows. Given a fixed alphabet size $|\mathcal{A}|$ and depth D, the maximum possible number of states is $|\mathcal{A}|^D$. For a (finite-length) symbol string S, the occurrence of each state is sequentially counted and let N_{ij} denote the number of times the symbol $s_j \in \mathcal{A}$ is emitted from the state $q_i \in Q$. Thus;

$$\Pi_{ij} = \pi(q_i, s_j) \triangleq \frac{1 + N_{ij}}{|\mathcal{A}| + \sum_\ell N_{i\ell}} \tag{1}$$

The rationale for initializing the count of each element to 1 is that if no event is generated at a state $q \in Q$, then there should be no preference to any particular symbol and it is logical to have $\pi(q, s) = 1/|\mathcal{A}|\ \forall s \in \mathcal{A}$, i.e., the uniform distribution of event generation at the state q. The above procedure guarantees that the *PFSA*, constructed from a (finite-length) symbol string, must have an (element-wise) strictly positive morph map Π. This also ensures ergodicity and stochasticity of the morph matrix Π by construction [14,28].

Remark 8. Construction of the morph matrix Π and state transition probability matrix T guarantees that these matrices are both stochastic (i.e., each matrix element is non-negative and each row sum is unity [29]) and ergodic (i.e. every state of the PFSA can be reached in a finite number of iterations irrespective of the starting state). The stochasticity and ergodicity properties of these matrices are key points in the mathematical theory of the modified projection-based PFSA (p-PFSA) developed in this paper.

2.3. Standard PFSA-based classification

The classification algorithms, based on the standard *PFSA* (*s-PFSA*), are developed for training and testing phases as described below.

Training phase: The *s-PFSA* classification algorithm is trained on time-series from each of the C possible classes, based on the user-specified parameters: window length WL, downsampling rate DS, alphabet size $|\mathcal{A}|$, and depth D. The PFSA morph matrices for each class is computed as Π^c, $c = 1, \dots, C$, by taking an average of the individual morph matrices obtained from each segmented time series of window length WL.

Testing (or classification) phase: The *s-PFSA* algorithm is tested on a (single) data window of length WL from a time series, belonging to an unknown class. The test time-series is symbolized using the same partitioning and the same parameters (i.e., $\mathcal{A}$, D, WL, WS, and DS) as for training. The resulting morph matrix Π^{tst} is also generated by following the same procedure as in the training phase. The distance (e.g., Kullback–Leibler divergence or Euclidean norm) of the classification morph matrix from each of the C trained morph matrices are computed. The class that yields the smallest distance is classified to be the class the testing time-series belongs to:

$$\text{Identified Class} = \underset{c\in\{1,2,\dots,C\}}{\text{argmin}} \ \|\Pi^{tst} - \Pi^c\| \tag{2}$$

Thus, the decision making in Eq. (2) is thresholdless. In this paper, *s-PFSA* has been used as one of the three standard benchmark methods, where the other two are and shallow NN, to compare the results of the proposed PFSA-based projection method (*p-PFSA*) which is presented later in Section 3. Algorithms for *s-PFSA* are available in [8].

2.4. Hidden Markov modeling for classification

There are various applications of hidden Markov modeling (*HMM*), which have shown high classification accuracy [2]. Therefore, *HMM* has been chosen as one of the two baseline methods for performance evaluation of *p-PFSA* which is presented later in Section 3.

In the training phase of a *HMM* classification problem having C classes, the Baum–Welch algorithm [1] is used to learn a from the time series belonging to each regime, i.e., classes $c = 1, \dots, C$. Similar to *s-PFSA*, the time series from each class is windowed at user-specified window length (WL), window shift (WS), and downsampling rate (DS).

The testing phase takes time series from a window corresponding to an unknown regime, and the *Forward Procedure* [1] is used to compute the loglikelihood, L^c, where $c = 1, \dots, C$, of the given data to belong to each of the C classes. The final decision as to which class the unknown data is most likely to belong is made by selecting the class with the largest loglikelihood as follows, i.e, this method is also thresholdless:

$$\text{Identified Class} = \underset{c\in\{1,2,\dots,C\}}{\text{argmax}} \ L^c \tag{3}$$

A continuous formulation has been used in this paper, which uses a Gaussian mixture model with M Gaussian components to model the emission and there are N hidden states. Detailed algorithms for are available in [1].

2.5. Neural networks for classification

Neural networks are very versatile and have been widely used for different applications, including object recognition [30], system modeling and analysis [31], time series classification [32] and many more.

The mathematical theory behind neural networks is well known [1,33] and has not been included in this paper for brevity. The important point to note is that the neural network is an arrangement of an input layer that receives the data to be classified, hidden layers with their respective weight matrices and activation functions, and an output layer that produces the decision. The weight matrices are learned from the data, and inherently the neural network algorithm is thresholded, i.e., it learns thresholds for itself that provides the optimal separation of the data among the C classes.

In the current paper, the authors have reported three *NN* architectures for a comparative performance evaluation of *p-PFSA* that is presented next in Section 3. The first two *NNs* are shallow fully connected nets while the third is a shallow fully connected net, preceded by an additional *LSTM* layer. The LSTM layer is used to learn the temporal evolution by back-propagating through time [34]. In all these *NN*s, the dense layers except the output have the rectified linear unit (*ReLU*) activation; output layers have the Softmax activation; and the cells in the *LSTM* layers have the tangent hyperbolic (tanh) activation functions.

3. Projection-based PFSA for classification

This section develops a thresholdless classification algorithm, which is built on the concept of projection-based PFSA, called *p-PFSA*. The key idea is as follows.

> A hyperplane in the feature space is computed from the time series of each regime (i.e., class). The projection of a feature (obtained from the data to be classified) is projected onto each hyperplane to achieve thresholdless classification [35]. The core concept is explained below.

Let $T \in \mathbb{R}^{|Q|\times|Q|}$, where $|Q|$ is the cardinality of the set of PFSA states (see Definition 1), be an ergodic stochastic matrix [29]. The ergodic property of the stochastic matrix T guarantees the existence of exactly one eigenvalue $\lambda_0 = 1$ while the remaining $(|Q|-1)$ eigenvalues are located on or inside the unit circle (with center at 0) in the complex plane, i.e., $|\lambda_i| \le 1$ for $i = 1, 2, \dots, (|Q|-1)$. In the situation that some of the eigenvalues are repeated, it is possible that there will be only m linearly independent $(1 \times |Q|)$ left eigenvectors, $v_1, \dots, v_m$, and m linearly independent$(|Q| \times 1)$ right eigenvectors, $u_1, \dots, u_m$, where $1 \le m \le (|Q|-1)$. Let v_0 and u_0 be respectively the (normalized) left and right eigenvectors of T with respect to the unique eigenvalue $\lambda_0 = 1$. The remaining left and right eigenvectors are respectively v_i and u_i corresponding to the distinct eigenvalues λ_i. These eigenvalues are ordered for convenience as: $1 \ge |\lambda_1| \ge \cdots \ge |\lambda_m| \ge 0$. It is noted that these eigenvalues are either real or pairs of complex conjugates and their respective left eigenvectors $\{v_i\}$ and right eigenvectors $\{u_i\}$ are also either real or pairs of complex conjugates as seen below.

Since T is a real matrix, $v_i T = \lambda_i v \Rightarrow \bar{v}_i T = \bar{\lambda}_i \bar{v}_i$; and $T u_i = \lambda_i u_i \Rightarrow T\bar{u}_i = \bar{\lambda}_i \bar{u}_i$, where $\bar{v}_i$, $\bar{u}_i$ and $\bar{\lambda}_i$ are complex conjugates of v_i, u_i and λ_i, respectively.

Claim. *For distinct eigenvalues $\lambda_i \ne \lambda_j$, $(i \ne j)$, $i, j \in \{0, 1, \dots, m\}$, the inner product $\langle v_i', u_j \rangle = 0$, i.e., $v_i' \perp u_j$.*

Justification of Claim: Since $\lambda_i \ne \lambda_j$, at least one of them is non-zero. Without loss of generality, one may set $\lambda_i \ne 0$. Then, $\langle v_i', u_j \rangle = v_i u_j = \frac{1}{\lambda_i} v_i T u_j = \frac{\lambda_j}{\lambda_i} v_i u_j = \frac{\lambda_j}{\lambda_i} \langle v_i', u_j \rangle \Rightarrow \left(1 - \frac{\lambda_j}{\lambda_i}\right) \langle v_i', u_j \rangle = 0$. Given $\lambda_i \ne \lambda_j$ and $\lambda_i \ne 0$, it follows that $\langle v_i', u_j \rangle = 0 \Rightarrow v_i' \perp u_j$.
End of Justification

The above claim leads to the conclusion that, for a PFSA of a given class $c \in \{1, \dots, C\}$, $v_0^c \perp u_j^c$ corresponding to each distinct eigenvalue λ_j^c, $j = 1, 2, \dots, m^c$. In the training phase, therefore, one may identify v_0^c and u_j^c $\forall j = 1, \dots, m^c$ $\forall c \in \{1, \dots, C\}$. During the testing phase, in contrast, it is necessary to identify only the vector v_0^{tst} (corresponding to the unknown regime (i.e., class) to which the test data belongs) for detection and classification of anomalous events, if any. In the absence of any anomaly, the vector v_0^{tst} should be (nearly) coincident with v_0^1 and thus (nearly) orthogonal to the subspace spanned by u_j^1 $\forall j = 1, \dots, m^1$ (of the nominal regime $c = 1$), which has already been identified in the training phase. However, if an anomaly occurs, v_0^{tst} should very likely deviate from v_0 that corresponds to the nominal phase and the task is identify the regime (i.e., the class $c \ne 1$) to which the test time series belongs.

The main objective here is quantification of the anomaly as a deviation of v_0^{tst} from v_0^c, where $c \in \{1, \dots, C\}$. This is achieved by projecting v_0^{tst} onto each of the C subspaces spanned by u_j^c $\forall j = 1, \dots, m^c$. From the perspectives of energy distribution, for each regime $c \in \{1, \dots, C\}$, the absolute values $|\lambda_j^c|$ of the eigenvalues signify the energy associated with the (normalized) eigenvector u_j^c; therefore, each u_j^c, $j = 1, \dots, m^c$ is weighted as $\tilde{u}_1^c \triangleq \sqrt{|\lambda_j^c|}\, u_j^c$. The normalized version of the vector $\sqrt{|\lambda_j^c|}\, u_j^c$ is u_j^c, because the u_j^c's were generated as normalized vectors. For a general ergodic stochastic matrix T, the right eigenvectors may not be orthogonal to each other [29]. Therefore, for each class $c \in \{1, \dots, C\}$, the (linearly independent) vectors $\sqrt{|\lambda_j^c|}\, u_j^c, j = 1, \dots, m^c$ are further transformed by Gram–Schmidt orthogonalization as a set of mutually orthogonal vectors $\tilde{u}_j^c, j = 1, \dots, m^c$, i.e., $\langle \tilde{u}_i^c, \tilde{u}_k^c \rangle = 0$ $\forall i \ne k$. The procedure follows for each class $c \in \{1, \dots, C\}$ as:

$$\tilde{u}_1^c \triangleq \sqrt{|\lambda_1^c|}\, u_1^c; \tag{4}$$

and for $i = 2, \dots, m^c$:

$$\begin{aligned} \tilde{u}_i^c &\triangleq \sqrt{|\lambda_i^c|} u_i^c - \sum_{j=1}^{i-1} \left\langle \sqrt{|\lambda_i^c|} u_i^c, \sqrt{|\lambda_j^c|} u_j^c \right\rangle u_j^c \\ &= \sqrt{|\lambda_i^c|} \left(u_i^c - \sum_{j=i}^{i-1} \left\langle u_i^c, \sqrt{|\lambda_j^c|} u_j^c \right\rangle u_j^c \right) \end{aligned} \tag{5}$$

For each class $c \in \{1, \dots, C\}$, having identified the m mutually orthogonal n-dimensional vectors $\tilde{u}_i^c$, the n-dimensional weighted error vector $\mathcal{E}^c$ is obtained by projection of the vector v_0^{tst} onto the m^c-dimensional subspace, spanned by $\tilde{u}_i^c$, $i = 1, \dots, m^c$, where $1 \le m^c \le (|Q|-1)$, as:

$$\mathcal{E}^c \triangleq \sum_{i=1}^{m} \langle v_0^{tst}, \tilde{u}_i^c \rangle u_i^c \quad \text{with } \|\mathcal{E}^c\| = \sqrt{\sum_{i=1}^{m^c} |\langle v_0^{tst}, \tilde{u}_i^c \rangle|^2} \tag{6}$$

Remark 9. Conceptually, the proposed $p - PFSA$ algorithm is equivalent to projection onto the null space of v^{tst}. However, having computed the eigenvalues and eigenvectors of the state transition probability matrices, the proposed $p - PFSA$ algorithm is numerically more efficient than projection onto the null space of v_0^{tst}. Furthermore, $p - PFSA$ is expected to yield more robust classification than if v_0^{tst} were simply compared with v_0^c $\forall c \in \{1, \dots, C\}$.

3.1. The projection PFSA algorithm: p-PFSA

This section applies the concept of projection, described above, to the classification task that requires discrimination among a total number of C regimes (i.e., classes). In the training phase, a PFSA is generated for each class as described in Section 2.1. For each class $c \in \{1, 2, \ldots, C\}$, the $(|Q| \times |Q|)$ state transition probability matrix T^c is treated as the class feature for the class c. In this paper, T^c is equivalent to Π^c because the PFSA is constructed with $D = 1$ (see Remark 7). Further, each T^c is ergodic and stochastic (see Remark 8); therefore, for T^c, there exists a set of orthogonalized right eigenvectors $\{\tilde{u}_i^c\}$, where for each class c, $i = 1, \ldots, m^c$, and $1 \le m^c \le (|Q| - 1)$ is the number of linearly independent eigenvectors for class c. Hyperplanes of dimension m^c are generated from each set of orthogonalized right eigenvectors $\{\tilde{u}_i^c\}$ corresponding to each trained class, and it is expected that, for different classes that represent distinct operational regimes, the hyperplanes will be distinct, i.e, a hyperplane belonging to class c will have its distinct normal direction v_0^c. Classes that are close to each other will have directions close to each other but they are still expected to be distinct.

For the testing (i.e., classification) phase, a time series is observed, which belongs to an unknown class; and then its state transition probability matrix T^{tst} is computed. It follows from Section 2 that the left eigenvector $(v_0^{tst})'$ of the (stochastic and ergodic) T^{tst}-matrix is obtained corresponding to $\lambda_0^{tst} = 1$. Then, v_0^{tst} is projected onto each of the C hyperplanes that were generated in the training phase; and the norm of the error vector in each hyperplane is computed as $\|\mathcal{E}^c\|$, where $c \in \{1, \ldots, C\}$. The class corresponding to the lowest error magnitude is identified to be the class to which the testing time series belongs. This makes *p-PFSA* thresholdless as seen below.

$$\text{Identified Class} = \underset{c\in\{1,2,\ldots,C\}}{\operatorname{argmin}} \|\mathcal{E}^c\| \tag{7}$$

Algorithms for both training and classification phases of *p-PFSA* are provided as Algorithms 1 and 2, respectively. Similar to *s-PFSA*, the *p-PFSA* algorithms are tested on a (single) data window of length WL from a time series, belonging to an unknown class. The test time-series (of the unknown class) is symbolized using the same partitioning and the parameters (i.e., $\mathcal{A}$, D, WL, WS, and DS) as for training.

Algorithm 1 Training of *p-PFSA*

```
Input: Windowed (of length $WL$) & downsampled (with rate $DS$) time-series data corresponding to each of $C$ classes;
    Label of the class (c) for each time series window.
    Initialization: $C$ counters ($n^c$) set to 0.
    User-set PFSA parameters: $|\mathcal{A}|$ and $D$.
Output: $C$ sets of orthogonalized right eigenvectors forming a hyperplane for each class $c \in \{1, \cdots, C\}$.
1: for $j = 1$ to total number of windows do
2:     Symbolize time-series segment $j$ and generate PFSA
3:     Store state transition matrices $T_j^c$ as the extracted feature where $c$ is the labeled class (using Eq. (1))
4:     $n^c = n^c + 1$
5: end for
6: $T^c = \frac{1}{n^c} \sum_{j=1}^{n^c} T_j^c \; \forall c \in C$
7: for $j = 1$ to $C$ do
8:     Compute eigenvalues ($\{\lambda_i^c\}$), left eigenvectors ($\{v_i^c\}$) and right eigenvalues ($\{u_i^c\}$) for $T^c$
9:     Compute and store orthogonalized basis vectors to form hyperplane for class $c \in \{1, \cdots, C\}$, called $\{\tilde{u}_i^c\}$ (using Eqs. (4) and (5))
10: end for
11: return Hyperplanes for $C$ classes ($\{\tilde{u}_i^c\}$)
```

Algorithm 2 Regime Classification by *p-PFSA*

```
Input: A single window (of length $WL$) of downsampled data (with rate $DS$) from the observed time-series belonging to an unknown regime
    Orthogonalized surfaces for $C$ classes ($\{\tilde{u}^c\}$) obtained using the procedure in Algorithm 1
    User-set PFSA parameters: $|\mathcal{A}|$ and $D$.
Output: : Regime class $c \in \{1, \cdots, C\}$.
1: Symbolize given time-series segment from unknown regime, generate PFSA and state transition matrix ($T^{tst}$)
2: Compute the left eigenvector $(v_0^{tst})'$ corresponding to the eigenvalue $\lambda_0^{tst} = 1$ for $T^{tst}$
3: Compute projection error for each class ($||\mathcal{E}^c||$) (following Eq. (6))
4: Classify data window to belong to class with lowest value of $||\mathcal{E}^c||$ (as per Eq. (7))
5: return Identified the regime class $c \in \{1, \cdots, C\}$.
```

Example 10. Fig. 1 displays a simple example to elucidate the underlying concept of p-PFSA. For ease of understanding, the surfaces have been shown to be 2-D planes with four classes (i.e., $m^c = 2$ and $c = I, \ldots, IV$), which would be true if the dimension of each T-matrix is 3×3 (e.g., the alphabet size $|\mathcal{A}| = 3$ and $D = 1$), and each T-matrix has 3 linearly independent eigenvectors.

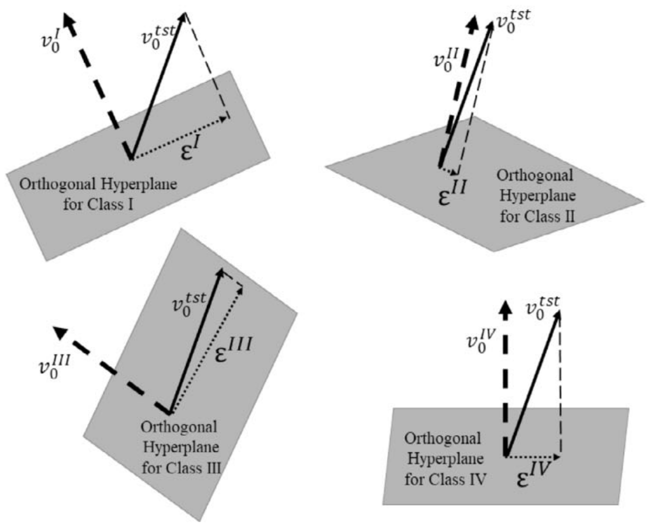

Fig. 1. Graphical representation of the PFSA-based projection algorithm *p-PFSA* with parameters: $|\mathcal{A}| = 3$, $D = 1$, and $C = 4$.

Each class c has its corresponding left eigenvector for the eigenvalue $\lambda_0 = 1$, denoted as $(v_0^c)'$, $c = I, \dots, IV$ in the four individual plates of Fig. 1 as indicated by arrowed dashed lines. By construction, these eigenvectors are normal to the 2-D planes. For an unknown time-series, the corresponding left eigenvector for $\lambda_0^{tst} = 1$ is shown as $(v_0^{tst})'$ (denoted by the arrow with the solid line). The v_0^{tst} is projected onto all four hyperplanes and the corresponding error vectors, denoted as $\mathcal{E}^c$, are also drawn in each plate of Fig. 1 and the corresponding norms $\|\mathcal{E}^c\|$ are magnitudes of the projected error vectors.

It is observed, in this example, that the magnitude $\|\mathcal{E}^{II}\|$ is the smallest, with $\|\mathcal{E}^{IV}\|$ being the second lowest. Also, by a visual inspection of the deviation between v_0^{tst} and the eigenvectors v_0^c, $c = I, II, III, IV$, it is seen that the vectors v_0^{tst} and v_0^{II} are closest to each other. Thus, the unknown vector v_0^{tst} is identified to belong to class II.

Remark 11. It is possible to encounter a matrix ^{c} with two or more repeated eigenvalues, where $m^c < (|Q| - 1)$, i.e., the $(|Q| - 1)$ right eigenvectors are not linearly independent. In such a situation or where a pair of eigenvectors are almost co-linear, the application of Gram–Schmidt orthogonalization would cause the projection of one of the two (almost) co-linear eigenvectors to vanish in magnitude. Thus, the rest of the computation would remain the same, yielding nearly zero weightage for a (nearly) linearly dependent eigenvector. Thus, the provided theory is robust to this situation.

4. Modification of data normalization

This section modifies the procedure for normalization of the raw time series data prior to partitioning and symbolization (see Remark 5). The rationale for this normalization is to remove signal bias and to construct a fixed finite set of *global* partitioning boundaries to be applicable across all data sets encountered in the classification problem, where the individual data sets may have different maximum and minimum values. There are primarily two ways in which normalization procedure can be executed:

(1) The time series is normalized to have a fixed range (i.e., the difference between maximum and minimum values), typically within a unit range so that the normalized data have a minimum value of 0 and a maximum of 1. However, any other set of bounds can also be used.
(2) The time series is normalized to have zero mean and unit variance. This method is preferable since it does not allow a few outliers to cause incorrect normalization, which the fixed range partitioning is prone to do.

However, the above process of normalization removes the information of the signal magnitude while retaining only the information about the signal texture (i.e., waveform), as seen in Fig. 2. The top row of Fig. 2 shows three sinusoidal signals having identical frequencies, but the left-hand and middle signals have an amplitude 10 units while the right-hand signal has an amplitude 20. Further, the left-hand and right-hand have signal means at 0 while the middle signal has a mean at 10 units. The equivalent normalized signals are shown in the bottom row of Fig. 2, where the three normalized zero-mean and unit-variance signals are identical. This implies that the *PFSA* constructed from these normalized signals will also be identical; therefore, it is impossible for *s-PFSA* to discriminate between these three signals although the original signals were different before normalization.

This paper proposes a modification of the normalization procedure to resolve this issue, which stores the numerical values of the mean and standard deviation of the raw (i.e., before normalization) time series in the training phase for each of the C classes, namely

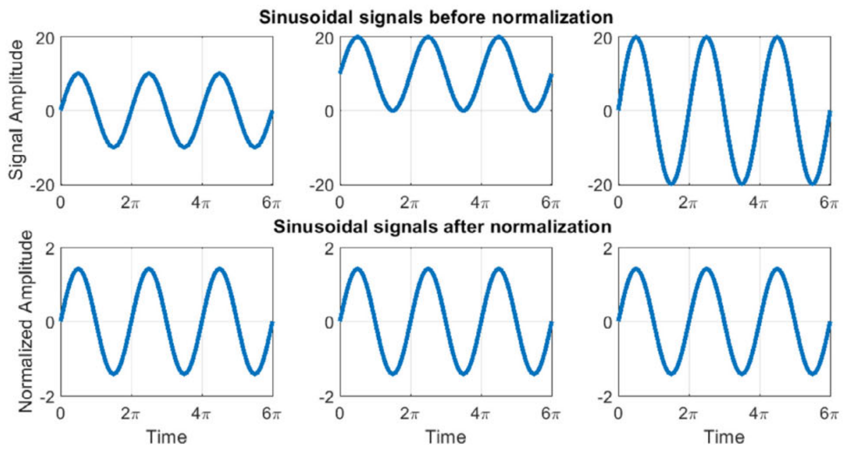

Fig. 2. Demonstration of the drawback of data normalization prior to symbolization showing signals prior to normalization (top row) and signals after zero mean, unit variance normalizing (bottom row).

μ^c and σ^c for $c = 1, \dots, C$. Subsequently, during the testing phase, the mean (μ^{tst}) and standard deviation (σ^{tst}) of the observed time series are computed. Accordingly Eqs. (2) and (7) are respectively modified as:

$$\text{Class}_{\text{s-PFSA}} = \underset{c\in\{1,\dots,C\}}{\text{argmin}} \left|\left|(T^{tst}, \mu^{tst}, \sigma^{tst}) - (T^c, \mu^c, \sigma^c)\right|\right| \tag{8}$$

$$\text{Class}_{\text{p-PFSA}} = \underset{c\in\{1,2,\dots,C\}}{\text{argmin}} \left|\left|(\mathcal{E}^{tst,c}, \mu^{tst}, \sigma^{tst}) - (0, \mu^c, \sigma^c)\right|\right| \tag{9}$$

where the information of each PFSA is augmented to a triplet including the respective mean and standard deviation of the time series both during training and testing. It is also noted that $\mathcal{E}^{tst,c}$ in Eq. (9) denotes the error computed by projecting v_0^{tst} onto the hyperplane of class c (see Section 3.1). The corresponding right hand entry is 0 because the error due to projection of v_0^c onto the hyperplane of class c is zero.

NOTE: The *PFSA* algorithm in its current form is designed to be able to study and classify *statistically quasi stationary* time-series data (e.g., signals arising from sensors in cyber–physical systems). However, *HMM* and *NN* are also further capable of learning long-term dependencies as well as complicated time-varying signals due to the much more complex mathematical algorithms governing them, which implies higher computational times as well as requirement of more data to learn a good model. Therefore, for '*statistically quasi stationary*' signals, it is expected that the proposed *p-PFSA* would compare well to state-of-the-art methods like *HMM* and *NN*.

5. Generation of data for algorithm validation

This section generates data in the form of time series for validation of the proposed pattern classification algorithm *p-PFSA*. Two types of data have been generated: (i) synthetic data (see Section 5.1) from four standard chaotic systems; this is a largely unexplored problem due to inherent difficulties of class separation; and (ii) experimental data of pressure-wave time-series (see Section 5.2) from a combustor apparatus to demonstrate a real-life engineering application.

5.1. Time series data from chaotic systems

This subsection generates data sets of time series from four well-known chaotic dynamical systems, namely, Hénon map [36], forced Duffing [37], Rössler attractor [38], and Lorenz attractor [39]. The objective here is to demonstrate efficacy of the proposed *p-PFSA* and compare its performance to those of *s-PFSA*, *HMM*, and *NN*. Although chaotic dynamical systems are essentially deterministic, they are sensitive to initial conditions and some of the system parameters. These chaotic data have been generated by varying pertinent parameters (see Table 1 for details) and are identical to those studied in a previous publication [3], where the efficacy of *HMM* in classifying chaotic conditions has been investigated.

The four chaotic dynamical systems, considered in this paper, are now briefly presented. The first chaotic system is **Hénon** map [36] that is characterized by the following pair of first-order coupled difference equations:

$$\begin{aligned} x_{n+1} &= 1 - a x_n^2 + y_n; \\ y_{n+1} &= b x_n \end{aligned} \tag{10}$$

Table 1
Case studies for different chaotic maps; listing fixed and varying parameters and initial conditions for training and test data.

Case	Chaotic system	Fixed parameter values	Varying parameter values	Initial conditions for training data	Initial conditions for testing data
I	Henon	$b = 0.3$	$a = 1$ to 1.4 in increments of 0.05	$(x_0, y_0) = (1, 1)$	$(x_0, y_0) = (1, 1)$
I-A	Henon	$b = 0.3$	$a = 1$ to 1.4 in increments of 0.05	$(x_0, y_0) = (1, 1)$	$(x_0, y_0) = (0, 0)$
II	Duffing	$\gamma = 0.2$, $\delta = 0.5$, $\alpha = 1$, $\beta = -1$	$\omega = 0$ to 2 in increments of 0.5	$(x_0, \dot{x}_0) = (0, 0)$	$(x_0, \dot{x}_0) = (0, 0)$
II-A	Duffing	$\gamma = 0.2$, $\delta = 0.5$, $\alpha = 1$, $\beta = -1$	$\omega = 0$ to 2 in increments of 0.5	$(x_0, \dot{x}_0) = (0, 0)$	$(x_0, \dot{x}_0) = (0.1, 0)$
III	Duffing	$\alpha = -1$, $\beta = 1$, $\delta = 0.3$, $\omega = 1.2$	$\gamma = 0.2$ to 0.60 in increments of 0.1	$(x_0, \dot{x}_0) = (0, 0)$	$(x_0, \dot{x}_0) = (0, 0)$
IV	Duffing	$\alpha = -1$, $\beta = 1$, $\delta = 0.3$, $\omega = 1.2$	$\gamma = 0.2$, 0.28, 0.29, 0.37, 0.50 & 0.65	$(x_0, \dot{x}_0) = (0, 0)$	$(x_0, \dot{x}_0) = (0, 0)$
V	Duffing	$\alpha = 1$, $\beta = 1$, $\gamma = 22$, $\omega = 5$	$\delta = 0.1$ to 0.35 in increments of 0.05	$(x_0, \dot{x}_0) = (0, 0)$	$(x_0, \dot{x}_0) = (0, 0)$
VI	Rossler	$b = 2$, $c = 4$	$a = 0.25$ to 0.55 in increments of 0.05	$(x_0, y_0, z_0) = (1, 1, 1)$	$(x_0, y_0, z_0) = (1, 1, 1)$
VI-A	Rossler	$b = 2$, $c = 4$	$a = 0.25$ to 0.55 in increments of 0.05	$(x_0, y_0, z_0) = (1, 1, 1)$	$(x_0, y_0, z_0) = (0, 1, 1)$
VII	Rossler	$b = 2$, $c = 4$	$a = 0.3547$, 0.398, 0.41, 0.4582, 0.5031 & 0.55	$(x_0, y_0, z_0) = (1, 1, 1)$	$(x_0, y_0, z_0) = (1, 1, 1)$
VIII	Rossler	$a = 0.2$, $c = 5.7$	$b = 0.2$ to 1.8 in increments of 0.4	$(x_0, y_0, z_0) = (0, 0, 0)$	$(x_0, y_0, z_0) = (0, 0, 0)$
IX	Rossler	$a = 0.1$, $b = 0.1$	$c = 5$ to 45 in increments of 5	$(x_0, y_0, z_0) = (0.2, 0.3, 0.2)$	$(x_0, y_0, z_0) = (0.2, 0.3, 0.2)$
X	Lorenz	$\sigma = 10$, $\beta = 8/3$	$\rho = 50$ to 250 in increments of 20	$(x_0, y_0, z_0) = (1, 1, 1)$	$(x_0, y_0, z_0) = (1, 1, 1)$
X-A	Lorenz	$\sigma = 10$, $\beta = 8/3$	$\rho = 50$ to 250 in increments of 20	$(x_0, y_0, z_0) = (1, 1, 1)$	$(x_0, y_0, z_0) = (0, 1, 1)$
XI	Lorenz	$\sigma = 10$, $\beta = 8/3$	$\rho = 13$, 14, 15, 28, 99.96, 100.3, 155, 193, 228, 229, 230.5 & 240	$(x_0, y_0, z_0) = (1, 1, 1)$	$(x_0, y_0, z_0) = (1, 1, 1)$

The second chaotic system is forced **Duffing** [37], representing the dynamics of a nonlinear spring, whose governing equation is given as:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma \cos \omega t \tag{11}$$

The third chaotic system is **Rössler attractor** [38], representing chemical reaction kinetics, which has the following three coupled first-order differential equations:

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{12}$$

The fourth chaotic system is **Lorenz attractor**, which is derived by reducing the order of Navier–Stokes equation [39], and is governed by the following three coupled first-order differential equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{13}$$

Since these chaotic systems are sensitive to the parametric values in Eqs. (10), (11), (12) and (13), the evolution of the system state variables may drastically change with a slight perturbation in any one of these parameters. In this context, regime classification would imply being able to identify which regime (i.e., the class represented by a set of system parameters) does the time series of an unknown signal belong to. However, this task is not straightforward, because multiple classes may end up being very close to each other (i.e., may have quite similar signal forms).

This difficulty can be easily appreciated by observing phase diagrams of the Rössler system in Fig. 3, which demonstrates how a small perturbation in the parameter c in Eq. (12) and keeping the other system parameters fixed at $a = 0.1$ and $b = 0.1$) can lead to a change in the signal switching between periodic and chaotic signals. Similarly, Figs. 4 and 5 show a segment of different time-series from Case IX (Rössler system) and Case X (Lorenz System), respectively. These two figures display how complex these chaotic signals are, while it is also observed that the signals between classes are almost visually indistinguishable in some cases. Thus, the class discrimination can be extremely difficult, which is probably the reason why there has not been much attempt to address the classification problem in chaotic dynamical systems.

5.2. Time series of data from a combustor apparatus

This subsection addresses signal generation for investigation of thermoacoustic instability (TAI) [40], which is a well-known problem encountered in enclosed combustors under certain operational conditions (e.g., Mondal et al. [23] and Bhattacharya et al. [24]). TAI is caused by constructive interference between fluid flow and thermal energy release; if they become phase &

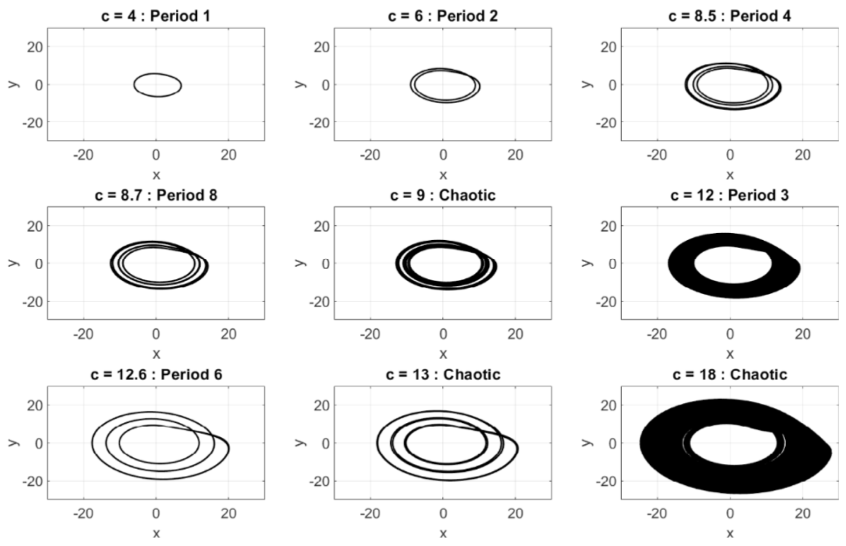

Fig. 3. Phase diagrams of the Rössler system for varying parameter 'c' indicating the system regime for that value, $a = 0.1$, $b = 0.1$, $(x_0, y_0, z_0) = (0, 0, 0)$.

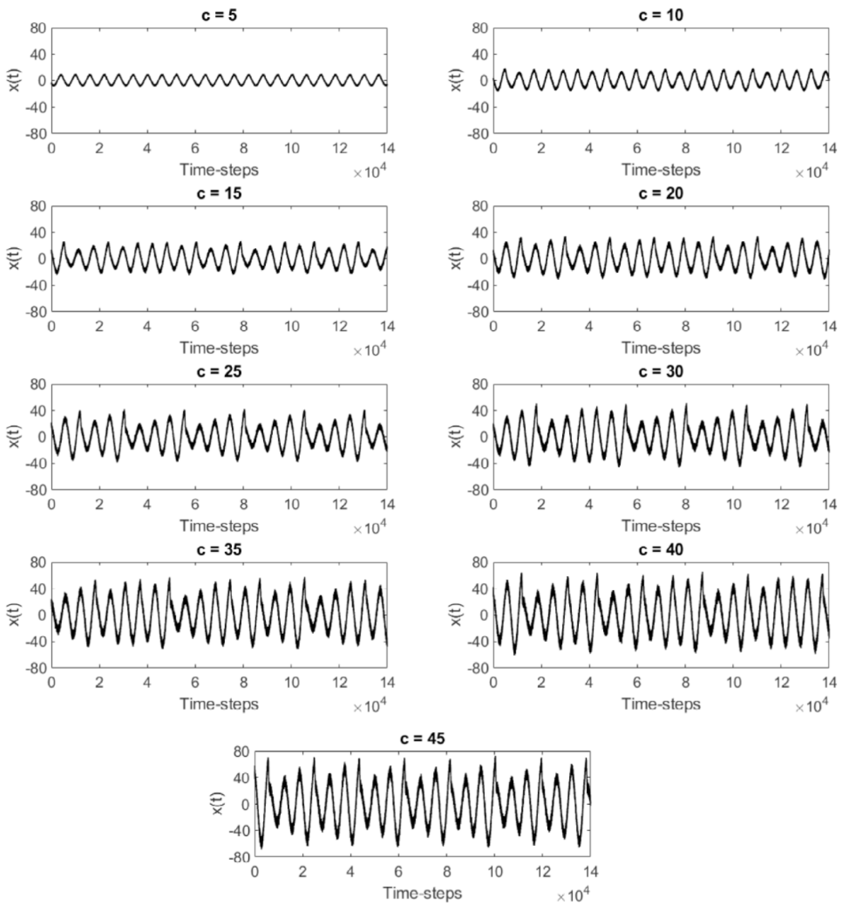

Fig. 4. Time-series of the Rössler system evolution for varying parameter 'c' (Case IX), $a = 0.1$, $b = 0.1$, $(x_0, y_0, z_0) = (0.2, 0.3, 0.2)$.

frequency aligned, then large-amplitude well-ordered pressure oscillations may occur. This causes mechanical stresses and vibrations in the structural components, none of which are desirable.

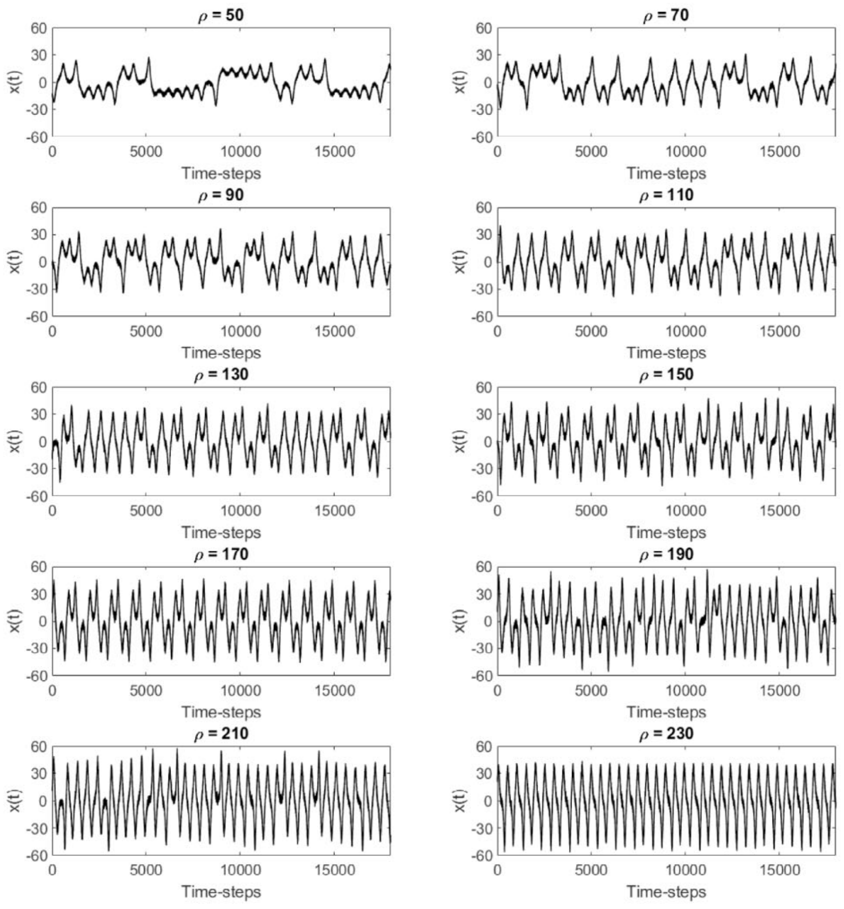

Fig. 5. Time-series of the Lorenz system evolution for varying parameter 'ρ' (Case X), $\sigma = 10, \beta = 8/3$, $(x_0, y_0, z_0) = (1, 1, 1)$.

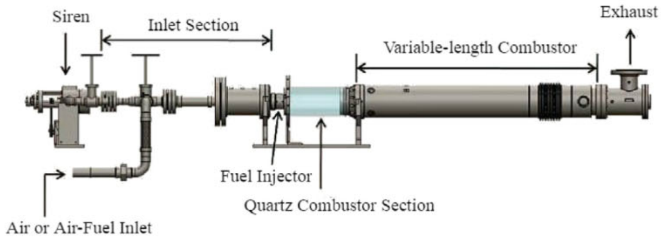

Fig. 6. Schematic diagram of the combustion apparatus.

Ensembles of pressure-wave time-series, corresponding to a variety of operational parameters, have been generated from a laboratory-scale combustor apparatus [22]. Fig. 6 shows a schematic diagram of the swirl-stabilized, lean-premixed, variable-length combustor apparatus that consists of an inlet section, an injector, a combustion chamber, and an exhaust section. There is an optically accessible quartz section followed by a variable length steel section which is set to have various lengths between ∼ 25–59 inches (∼ 64–150 mm) with ∼ 1 inch (∼ 25 mm) increments. A compressor supplies high pressure air which is preheated to 250 °C by an electric heater with the air flow rate ranging from 25 to 30 m/s with increments of 5 m/s. The fuel is natural gas (approximately 95% methane), the flowrate of which is adjusted to obtain fuel–air equivalence ratios (ϕ) of 0.525, 0.55, 0.60 and 0.65.

A total of 780 eight-second long time-series of pressure-wave signals exist for various combinations of the operational parameters (sampled at 8192 Hz using pressure transducers). Fig. 7 shows a comparison between typical stable and unstable time-series. It had been observed experimentally that the combustor becomes unstable once the root mean square (RMS) value of the pressure signal exceeds 0.07 psi (483 pascal). This knowledge has been used as the ground truth for training and testing the algorithms in this paper. It is important to note that a lot of these signals belong close to the borderline of this threshold and are not easy to classify [23].

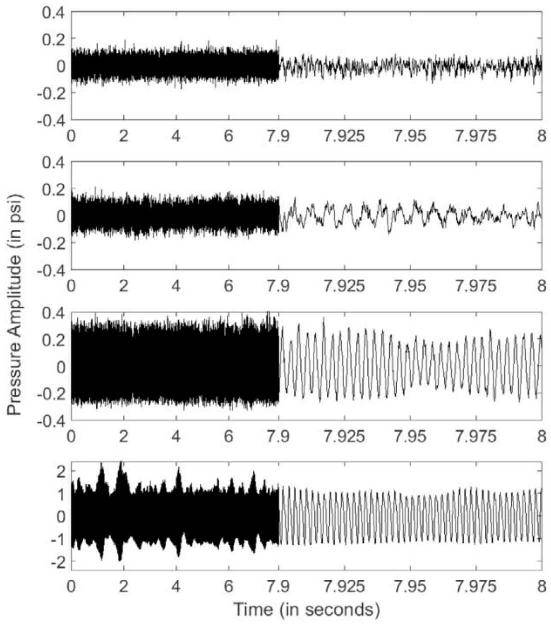

Fig. 7. Typical pressure time-series plots for various operating conditions: Top two plots represent stable operation; Bottom two plots represent unstable operation (TAI). It is noted that the range of the ordinate in the bottom-most plot is from −2 to +2 psi (-13,790 to 13,790 pascal), while that for the remaining three plots is −0.4 to +0.4 psi (-2,758 to +2,758 pascal).

It is seen in Fig. 7 that RMS values of the displayed pressure-wave signals tend to increase from top to bottom. The top-most plate represent stable operation, the second plate from top shows the 'border-line' operation, while the bottom two plates show unstable operation. It is evident that the top-most plate in Fig. 7 is the most stable with low-amplitude noise data. In the next plate, the signal amplitude is larger and begins to get ordered (i.e., tends toward instability although the RMS value is below the threshold of 0.07 psi). Such a case is prone to misclassification even if one uses typical state-of-the-art methods like *HMM*, *NN*, or visibility maps [23] as well as symbolic time series analysis [24]. In these border-line cases, the signals contain characteristics of both stable and unstable operation, leading to poor classification as compared to the visibly stable and unstable signals. The bottom two plates show large-amplitude highly ordered signals (i.e., full-blown instability). It is noted that the amplitude of the bottom-plate signal is 5 times the amplitude of the signal above it.

6. Results of algorithm validation

This section presents and discusses the results of validation of *p-PFSA* on the data sets, generated in both Sections 5.1 and 5.2 ; these results are compared with those for *s-PFSA*, *HMM*, and *NN*.

6.1. Algorithm validation with chaotic data

This subsection validates the algorithms of *p-PFSA* with synthetic data generated from four chaotic systems. Before presenting the results with normalization fix (see Section 4) on comparison of the proposed *p-PFSA* alongside *s-PFSA*, *HMM,* and *NN*, it is necessary to explain the specific nature of the data that are used for validating the algorithms (see Section 3.1). These chaotic data are identical to those reported by Bhattacharya and Ray [3], and the system parameters are listed in Table 1. For example, the signal data are generated by forward difference in the Hénon map (see Eq. (10)) and by fourth-order Runge–Kutta integration with a fixed time-step size of 1 ms for Duffing, Rössler and Lorenz systems (see Eqs. (11), (12) and (13)). For each parameter set, 5,000 s long data have been generated for both training and testing phases, yielding 5,000,000 data points per condition. Additionally, the signals are artificially contaminated with 10% noise (amplitude ratio formulation), which corresponds to a signal-to-noise ratio (SNR) of 20 dB. Table 1 also lists the initial conditions used to generate the training and testing data. These parameters and their ranges are taken from cited research publications (e.g., [7,36–39,41,42]), and are reported here for reproducibility of the results. The parameters of PFSA (i.e., both *p-PFSA* and *s-PFSA*) and *HMM*, namely, the alphabet size ($|\mathcal{A}|$ and the Markov depth $D = 1$), number of Gaussian components (M) and number of hidden states (N), need to be assigned. It is noted that higher depths (i.e., $D > 1$) of PFSA have not improved the results significantly.

The optimal values of parameters for each system of equations are slightly different. For a fair comparison, two values for each of the parameters $|\mathcal{A}|$ and N are studied, which are 10 and 30. The parameter M=4 is maintained across all trials of *HMM*. For *NN*s, three different nets are considered: (i) the first net, called *NN-1*, with 1 hidden layer consisting of 100 neurons; (ii) the second

Table 2
Results of case studies for the different chaotic maps; comparing the three *NN* methods and the *HMM*, *s-PFSA* (in presence and absence of the normalization fix) and p-PFSA methods for two values of alphabet size ($|\mathcal{A}|$) and number of hidden states (N).

Case	Chaotic system	DS = 100, WL = 1000										
		NN-1 error	*NN-2* error	*NN-LSTM* error	$|\mathcal{A}| = N = 10$				$|\mathcal{A}| = N = 30$			
					HMM error	*s-PFSA* error		*p-PFSA* error	*HMM* error	*s-PFSA* error		*p-PFSA* error
						Without fix	With fix			Without fix	With fix	
I	Henon	0.00%	0.00%	16.95%	1.03%	7.59%	3.64%	1.73%	0.97%	6.93%	3.25%	1.44%
I-A	Henon	31.61%	29.53%	41.27%	11.83%	7.33%	14.59%	12.47%	12.24%	8.31%	14.73%	12.98%
II	Duffing	0.00%	0.00%	79.99%	0.00%	2.67%	0.78%	0.00%	0.00%	8.67%	0.44%	0.00%
II-A	Duffing	0.00%	0.00%	79.99%	0.00%	3.63%	0.41%	0.00%	0.00%	8.19%	0.59%	0.00%
III	Duffing	0.00%	0.00%	39.49%	23.11%	25.15%	21.48%	22.33	4.48%	30.07%	28.59	23.11%
IV	Duffing	0.03%	0.00%	1.23%	17.81%	10.06%	10.40%	4.20%	1.11%	13.70%	13.06%	2.93%
V	Duffing	21.41%	1.91%	8.51%	46.33%	32.87%	30.43%	22.87%	30.77%	29.72%	26.70%	9.48%
VI	Rossler	0.00%	0.00%	85.71%	6.06%	24.47%	5.48%	4.34%	1.69%	19.23%	4.02%	5.11%
VI-A	Rossler	2.06%	1.98%	0.01%	6.51%	23.84%	4.74%	3.47%	1.69%	18.70%	3.44%	3.62%
VII	Rossler	0.00%	0.09%	0.84%	2.44%	10.12%	2.62%	1.51%	1.17%	5.19%	2.50%	1.57%
VIII	Rossler	0.00%	0.00%	0.0%	0.00%	21.22%	1.48%	1.11%	0.85%	21.93%	0.07%	0.96%
IX	Rossler	9.65%	9.82%	0.82%	0.37%	51.42%	0.93%	0.93%	0.10%	46.15	0.91%	0.93%
X	Lorenz	0.00%	8.00%	0.0%	0.00%	0.76%	0.11%	0.20%	0.00%	1.65%	0.17%	0.20%
X-A	Lorenz	61.66%	61.35%	0.0%	8.98%	1.02%	0.26%	0.39%	0.00%	2.31%	0.33%	0.41%
XI	Lorenz	25.69%	28.04%	0.0%	6.59%	23.28%	1.44%	5.17%	0.00%	25.57%	3.56%	5.22%
Average error		**10.14%**	**9.38%**	**23.66%**	**8.74%**	**16.36%**	**6.59%**	**5.38%**	**3.67%**	**16.42%**	**6.82%**	**4.53%**

net, called *NN-2*, having 2 hidden layers of respective sizes 100 and 20 neurons; and (iii) the third net, called *NN-LSTM*, that has a single *LSTM* layer with 100 hidden units, followed by 2 fully connected dense layers, which consist of 2 hidden layers of sizes 100 and 20 neurons, respectively. Each net is trained over 100 training epochs.

6.1.1. Fixed window size and downsampling rate

Tabulated comparisons of the total percentage error in classification for *NN*, *HMM*, *s-PFSA* and *p-PFSA* are presented in Table 2. For all of the above methods to be compared, the time series data are first downsampled by $DS = 100$. This is necessary because the time-step of integration is intentionally made small to assure the accuracy of integration, which leads to oversampling. The processed data are then windowed with a moving window formulation (see Section 2) for a window size of $WL = 1{,}000$ and a window shift value of $WS = 100$ data points as seen in [3]. The parameters DS, WL, and WS are held constant across all training and test cases. It is seen in Table 2 that, across the board, there is no preferred algorithm that performs the best in all of the cases. Both the *HMM* and PFSA-based algorithms work better for higher values of number of hidden states (N) and alphabet size ($\mathcal{A}$), respectively. It is apparent that *s-PFSA* without the normalization fix does not fare too well with an average error of about $\sim 16\%$ for both values of $|\mathcal{A}|$.

Significantly different results are seen after the normalization fix is introduced (see Section 4), which causes the accuracy of *s-PFSA* to improve to an error of about $\sim 6\%$. classification accuracy for *p-PFSA* (with the normalization fix included), in general, is superior to that for *s-PFSA* in nearly all data-sets with marked improvements in some cases (e.g., Case IV). The rationale is that inclusion of the signal amplitude and mean values in the classification metric allows the algorithm to compare both signal shape and value. The absence of normalization fix is a drawback of the *s-PFSA* formulation.

Table 2 lists the results generated from the four chaotic systems in Eqs. (10) to (13) with the same parameter values as those reported in [3], where *HMM* was chosen as a baseline method for testing the classification accuracy. It is seen in Table 2 that both *s-PFSA* and *p-PFSA* yield accuracy comparable to that of *HMM* after applying the normalization fix, where the average performance of *p-PFSA* is better than that of *s-PFSA*. While the two neural networks outperformed both *HMM* and PFSA-based methods in some cases (e.g., Cases III and V), they performed poorly in some other cases (e.g., Cases I-A, X-A and XI). It is noted that, for Case III except the neural networks, none of the other methods had good classification accuracy, with the exception of *HMM* with $N = 30$ to some extent. A possible reason is that the signals for some of the values of a varying parameter are almost identical, but when its values are chosen such that there is some change in the signals (Case IV). All cases perform better for *HMM* with $N = 30$ and *p-PFSA* with $|\mathcal{A}| = 30$ doing the best. Interestingly, a similar trend is seen in other data sets, where the signals are of similar characteristics; for example, Case V (see [3]), *p-PFSA* significantly outperforms *HMM* at both values of alphabet size / hidden states, while the *NN* methods do well only when there are 2 hidden layers (*NN-2*). The results of *NN-LSTM* are very interesting where, for some cases, namely, I, II, III and VI, this network performs very poorly as seen in Table 2, while it largely outperforms the other methods in a majority of the remaining cases. The rationale is probably the fact that the networks were not optimized in terms of their depths and neurons per layer, i.e., the network architectures were not optimized. The procedure of architecture optimization is a general problem in the *NN* literature and is a drawback by itself, because of the large training times needed as the process of exhaustive parameter search is very slow.

Table 3
Results of *p-PFSA* for case studies of four chaotic maps with optimal DS and WL along with normalization fix and two different values of alphabet size ($|\mathcal{A}|$).

Case	Chaotic system	Optimal DS and WL			
		$\|\mathcal{A}\| = 10$		$\|\mathcal{A}\| = 30$	
		s-PFSA error	*p-PFSA* error	*s-PFSA* error	*p-PFSA* error
I	Henon	2.25%	0.56%	2.05%	0.52%
I-A	Henon	3.06%	0.42%	2.03%	0.48%
II	Duffing	0.52%	0.00%	1.42%	0.00%
II-A	Duffing	0.37%	0.00%	1.88%	0.00%
III	Duffing	21.33%	22.65%	28.23%	23.50%
IV	Duffing	8.42%	3.12%	9.89%	1.38%
V	Duffing	17.31%	6.29%	6.34%	0.85%
VI	Rossler	5.08%	4.72%	3.86%	4.71%
VI-A	Rossler	4.37%	3.54%	3.30%	3.54%
VII	Rossler	3.04%	1.44%	3.29%	1.33%
VIII	Rossler	1.56%	1.17%	0.00%	0.42%
IX	Rossler	0.78%	0.78%	0.77%	0.78%
X	Lorenz	0.09%	0.09%	0.03%	0.09%
X-A	Lorenz	0.20%	0.21%	0.11%	0.19%
XI	Lorenz	0.44%	0.46%	0.22%	0.32%
	Average error	**4.59%**	**3.03%**	**4.23%**	**2.54%**

Considering the average errors across all cases as an overall performance metric of each method, it is seen that *HMM* with $N = 30$ does the best, while *p-PFSA* with $|\mathcal{A}| = 30$ and 10 come a very close second and third, respectively. The worst performance is seen by *s-PFSA* without the normalization fix.

6.1.2. Optimal window size and downsampling rate

One of the reasons for the PFSA-based methods (i.e., *s-PFSA* and *p-PFSA*) lagging behind *HMM* is the inherent sensitivity of PFSA to the window length (WL) and downsampling rate (DS). The construction of PFSA dictates that oversampling of the data could lead to "self looping" in the graph of PFSA (i.e., "strong diagonals" in the state transition probability matrix T), while undersampling does not capture the signal form well; both situations degrade the classification power. Regarding the window length WL, since PFSA is an algebraic method that uses the knowledge of the texture of a signal wave-form, a sufficient window-length is needed to accurately analyze the signal.

In view of the above, fixed values of the parameters WL and DS would not work across all cases. A logical method is proposed for choosing a reasonably good value of these the parameters WL and DS, which should be conducted in the training phase. In order to allow for correct sampling, each complete periodic waveform needs to be sampled at an adequate rate, i.e., enough discrete signal values from a single complete wave. For capturing a sufficient number of waveforms, the window length would need to be such that several complete cycles are seen for appropriate averaging.

In order to arrive at the exact values of WL and DS, a FFT based analysis of the training signals has been conducted to find the dominant frequencies from the ensemble of data. Taking an inverse of these frequencies yields a set of prominent wavelengths in the data, where DS is set such that the highest frequency wave (i.e., shortest wavelength) is sampled for at least 40 points, while WL is set to be at least 60 times the longest wavelength. Finally, to allow for a fair comparison to the results in Section 6.1.1, it is ensured that the value of the product $WL \times DS$ is less than or equal to $1,000 \times 100 = 10^5$. If the product exceeded this value, WL and DS must be adjusted to be less than 10^5, while a lower bound of $DS_{min} = 10$ is set. These values of bounds have been obtained by several tested trials and apparently produced the highest accuracy. A comparison of the classification accuracy for both *s-PFSA* and *p-PFSA* using these optimal values of WL and DS is presented in Table 3, where it is seen that an optimal selection of WL and DS with normalization fix improves the performance of both *s-PFSA* and *p-PFSA*. It is noted that the accuracy of *p-PFSA* in Table 3 exceeds that of *HMM* with fixed WL and DS in Table 2. Classification by *HMM* has not been attempted in this paper for optimal WL and DS, because of the requirements of much larger training and testing times of *HMM*; this topic is further discussed in Section 6.1.3 and is suggested as atop[c of future research in Section 7. Optimal values of WL and DS may often lead to very large values of WL and/or low values of DS or both, resulting in each window to have large data points, which would slow down the training of *HMM*. Similarly, the *NN* method was also not attempted for optimal WL and DS.

6.1.3. Time complexity

The *PFSA* architecture, adopted for both *s-PFSA* and *p-PFSA* is essentially algebraic and does not require any recursion (see Sections 2.1 and 2.2), which has the advantage of having fast execution time for both training and testing. This architecture was used by the authors in [43] for online discovery and detection of classes. In contrast, *HMM* uses an iterative maximization method for both training and testing phases, which are computationally more complex (see Section 2.4), leading to much larger training and testing times. Similarly. the *NN*-based methods are relatively slow to train due to the large number of parameters to be learnt (e.g., via gradient descent), but are relatively faster than *HMM*, though still slower than *PFSA*-based methods. Previously, literature has been lenient to this fact, because of relatively high accuracy of *HMM* and *NN*. However, it is shown in this paper that *p-PFSA*

Table 4
Time complexity chart for comparison of *NN-1*, *NN-2*, *HMM* and *s-PFSA* with *p-PFSA* for two values of alphabet size ($|\mathcal{A}|$) and number of hidden states (N).

		Mean training time (in ms)	Mean testing time (in ms)
NN-1		19.96	6.97
NN-2		20.48	6.83
NN-LSTM		2224.34	7.65
$|\mathcal{A}| = N = 10$	*HMM*	946.29	41.83
	s-PFSA	0.34	0.48
	p-PFSA	0.47	0.46
$|\mathcal{A}| = N = 30$	*HMM*	2037.21	93.02
	s-PFSA	0.42	1.32
	p-PFSA	1.25	1.12

(and even *s-PFSA* in a few cases) can provide similar or superior accuracy for time-series classification with significantly smaller training and testing times.

Table 4 compares the training and testing times of *NN-1*, *NN-2*, *NN-LSTM* and *HMM* with those of *p-PFSA* for two selected values of the parameters $|\mathcal{A}|$ and N, while the other the parameters are held fixed at $WL = 1,000$ and $DS = 100$. It is seen that the training time for *HMM* is approximately 3 orders of magnitude larger than either of the PFSA-based methods, while the testing time is about 2 orders of magnitude larger. The training and testing times for vanilla NN lie in between those for PFSA and *HMM*. The training time of *NN-LSTM* is even worse than that of *HMM*, while still not being able to provide good results. It is noted that deeper and more complex nets may take even longer time to train. The testing time is similar to that of the vanilla NNs. To put into context of time complexity, *NN-LSTM* took about 90 h to train and test all data sets, while *p-PFSA* did the same tasks in ~ 3 minutes. It is also seen in Table 4 that both *s-PFSA* and *p-PFSA* have similar training and testing times. Thus, both PFSA methods are faster than *HMM* and using the proposed modifications (i.e., normalization fix and using optimal parameters WL and DS), *p-PFSA* is capable of achieving an almost similar and possibly slightly better accuracy than *HMM* and *NNs* across a wide range of chaotic data, considered in this paper.

NOTE: *A single core processor of a DELL Precision Tower 7910 Workstation running on an Intel® Xeon® E5-2670 CPU has been used to evaluate the computation times for all of the methods. The MATLAB-based PFSA codes are developed in-house,*[1] *while the HMM codes are from [1] Hidden Markov Model ((HMM) Toolbox for MATLAB.*[2] *The NN algorithms are developed in a Python environment using the Keras library.*

6.2. Results of algorithm validation with combustion data

This subsection validates the algorithms of *p-PFSA* with experimental data of a real-life physical process. Using the data set of pressure time series, generated from the combustor apparatus (see Section 5.2), the proposed *p-PFSA* is compared to *s-PFSA*, *HMM*, *NN*, and *NN-LSTM*. The ensemble of time series is separated into respective training and testing data by randomly using a 70-30 split. Similar to the time series of chaotic data sets (see Section 6.1), the time series of combustion data sets too must be broken into windows of respective length WL, with a window skip WS and a downsampling rate DS. Consequently, each 8-second (i.e., 65,536 data points) long pressure-wave time-series is split into windows of length $WL = 100$ ms, at a sampling rate of 20 Hz, i.e., 20 windows per second, leading to $WS = 50$ ms at a downsampling rate of $DS = 2$.

For the combustion data, PFSA alphabet size $|\mathcal{A}|$ and number of hidden *HMM* states N are kept at 10 each. The *HMM* yields best results for the number of Gaussian mixtures $M = 2$. Because of limited availability of experimental data, only two *NN* architectures are tested, one having a single hidden layer of 100 neurons (*NN-1*), with the other having an *LSTM* layer with 100 neurons followed by a single hidden layer with 100 neurons (*NN-LSTM*). Each net was trained for 100 epochs. The computational time complexities of *p-PFSA*, *s-PFSA*, *HMM*, *NN* and *NN-LSTM* are qualitatively similar to those presented in Table 4 (see Sub-Section 6.1.3).

Table 5 presents the classification accuracy of four pattern classification methods under consideration: *p-PFSA*, *s-PFSA*, *HMM*, *NN-1* and *NN-LSTM*, where *p-PFSA* outperforms both *HMM* and *s-PFSA*. It is also seen that *NN-1* and *NN-LSTM* perform better than *p-PFSA* due to the following fact.

The inherent architecture of *NN* allows the learned weights to generate results that are similar to those produced by thresholded classification. The *p-PFSA* can be used in a thresholded fashion too [21]; however, this paper focuses on a thresholdless approach.

[1] Available at: https://github.com/Chandrachur92/PFSA.
[2] Available at: https://www.cs.ubc.ca/~murphyk/Software/HMM/*HMM*.html.

Table 5
Classification accuracy of combustion data: Using *NN*, *NN-LSTM*, *HMM*, *s-PFSA* and *p-PFSA*.

	NN	*NN-LSTM*	*HMM*	*s-PFSA*	*p-PFSA*
Classification error	5.73%	5.39%	16.38%	12.48%	9.13%

7. Summary, conclusions, and future work

This paper has proposed a projection-based extension of the standard probabilistic finite state automaton, called *s-PFSA* [7,8], to perform data-driven thresholdless classification in chaotic dynamical systems. It has also demonstrated an application of this classification method to a (real-life physical) process on a laboratory-scale apparatus [22] that emulates the essential characteristics of industrial-scale combustion systems. The key concept of the proposed projection-based method, called *p-PFSA*, is built upon the ergodicity and stochasticity properties [29] of the state transition probability matrix T that is derived by symbolization of time series of both training and test data. The projection of the left eigenvector, corresponding to the (unique) unity eigenvalue of T^{tst} onto the hyperspaces, spanned by the respective right eigenvectors corresponding to the remaining distinct eigenvalues of trained state transition probability matrices T^c, for different classes $c = 1, \ldots, C$. Furthermore, analysis to remedy a drawback in *s-PFSA* due to normalization of time series has been investigated, and a fix has been provided to improve the classification accuracy. An additional methodology for selection of optimal hyper-parameters (e.g., window-length WL and downsampling rate DS) is introduced, which shows a further improvement in the classification accuracy.

Efficacy of the proposed *p-PFSA* has been shown on synthetic data, generated from four different chaotic dynamical systems. Classification of chaotic data itself is a rather difficult task and *HMM*-based classification results from [3] have been used as a baseline, along with results of additional comparison with shallow neural network (*NN*)-based methods with long short-term memory (*LSTM*). In order to demonstrate how *p-PFSA* performs in real-life situations, experimental data of pressure-wave time-series from a laboratory-scale combustor apparatus have been used to classify whether the combustor is stable or having thermoacoustic instability. Once again *p-PFSA* performs well compared to *s-PFSA* and *HMM*, but it could be (possibly) outperformed by *NN* if sufficient training data are available, which is not a requirement for *p-PFSA*.

While there are many areas of theoretical and experimental research to improve the proposed classification methodology, the following topics are suggested for future research:

(1) *Identification of self-learned thresholds for multi-class pattern classification in the setting of p-PFSA*: This research is necessary, because thresholded approaches have been shown to yield major improvements in the accuracy of binary classification problems [21].
(2) *Theoretical research on optimal identification of hyper-parameters of window-length (WL) and downsampling rate (DS)*: This research is necessary to enhance the performance of *p-PFSA* for a wide range of dynamical systems, which include different types of physical processes [14,21,28].
(3) *Comparison of performance (e.g., classification accuracy, robustness, and computational complexity) of p-PFSA with that of deep neural networks and their configurations*: This research is necessary to make *p-PFSA* a competitive tool of machine learning.
(4) *Experimental research on validation of p-PFSA for a variety of physical applications*: This research is necessary to make *p-PFSA* acceptable for (real-life) industrial applications
(5) *Extension of the p-PFSA algorithm to have a capability of learning dynamically changing data that are not restricted to be statistically quasi-stationary [14,28]*: This research is necessary to improve the performance of *p-PFSA* under transient operations of the dynamical system under consideration.

Acronyms of frequently used terms

CNN	convolutional neural networks
DNN	deep neural networks
DS	downsampling rate
FSA	finite state automaton
HMM	hidden Markov model
LSTM	long short-term memory
MEP	maximum entropy partitioning
ML	machine learning
NN	neural networks
p-PFSA	projection-based PFSA
PFSA	probabilistic finite state automaton
ReLU	Rectified linear unit
RNN	recurrent neural networks
s-PFSA	standard PFSA
STSA	symbolic time series analysis
WL	window length
WS	window shift

Nomenclature of frequently used symbols

$\mathcal{A}$	alphabet of symbols for *PFSA*
C	number of classes or regimes
c	class or regime
D	depth of a D-Markov machine
L	loss function
m	number of linearly independent eigenvectors
M	number of Gaussian mixtures in a *HMM*
N	number of hidden states in a *HMM*
Π	morph matrix for *PFSA*
T	state transition probability matrix for *PFSA*
Q	set of *PFSA* states
λ	eigenvalue of T
u	right eigenvector of T
v	left eigenvector of T
μ	statistical mean
σ	statistical standard deviation

CRediT authorship contribution statement

Chandrachur Bhattacharya: Conceptualization, Methodology, Software, Formal analysis, Data curation, Writing – original draft, Writing – review & editing. **Asok Ray:** Conceptualization, Methodology, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Funding acquisition.

Declaration of competing interest

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

Acknowledgments

The authors are thankful to Professor Domenic Santavica at Penn State, who has kindly provided the experimental data. The work reported here has been supported in part by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant No. FA9550-15-1-0400 in the area of dynamic data-driven application systems (DDDAS). The authors also thank Indo-US Science and Technology Forum (IUSSTF) for granting the Research Internship for Science and Engineering (RISE) scholarship to the first author. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

All authors have read and agreed to the preparation of the manuscript.

References

[1] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012, ISBN: 0262018020, 9780262018029.
[2] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[3] Chandrachur Bhattacharya, Asok Ray, Data-driven detection and classification of regimes in chaotic systems via hidden Markov modeling, ASME Letters in Dynamic Systems and Control 1 (1) (2021) 021009, http://dx.doi.org/10.1115/1.4047817.
[4] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1578–1585.
[5] M. Husken, P. Stagge, Recurrent neural networks for time series classification, Neurocomputing (ISSN: 0925-2312) 50 (2003) 223–235.
[6] Yong Yu, Xiaosheng Si, Changhua Hu, Jianxun Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Comput. 31 (7) (2019) 1235–1270.
[7] A. Ray, Symbolic dynamic analysis of complex systems for anomaly detection, Signal Process. 84 (7) (2004) 1115–1130.
[8] K. Mukherjee, A. Ray, State splitting and merging in probabilistic finite state automata for signal representation and analysis, Signal Process. (ISSN: 0165-1684) 104 (2014) 105–119.
[9] Peter Beim Graben, Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics, Phys. Rev. E 64 (5) (2001) 051104.
[10] C.S. Daw, C.E.A. Finney, A review of symbolic analysis of experimental data, Rev. Sci. Instrum. 74 (2003) 915–930.
[11] P. Dupont, F. Denis, Y. Esposito, Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms, Pattern Recognit. (ISSN: 0031-3203) 38 (9) (2005) 1349–1371.
[12] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R.C. Carrasco, Probabilistic finite-state machines - part I and part II, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1013–1039.
[13] S. Sarkar, S.R. Chakravarthy, V. Ramanan, A. Ray, Dynamic data-driven prediction of instability in a swirl-stabilized combustor, Int. J. Spray and Combust. Dyn. 8 (4) (2016) 235–253.
[14] Najah F. Ghalyan, A. Ray, Symbolic time series analysis for anomaly detection in measure-invariant ergodic systems, J. Dyn. Syst. Meas. Control 142 (2020) 061003, (1 to 11).
[15] Y. Li, D.K. Jha, A. Ray, T.A. Wettergren, Information-theoretic performance analysis of sensor networks via Markov modeling of time series data, IEEE Trans. Cybern. 48 (6) (2018) 1898–1909.

[16] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.
[17] D.S. Dendrinos, Traffic-flow dynamics: A search for chaos, Chaos Solitons Fractals 4 (4) (1994) 605–617.
[18] D.A. Hsieh, Chaos and nonlinear dynamics: Application to financial markets, J. Finance 46 (5) (1991) 1839–1877.
[19] Nicolas Boullé, Vassilios Dallas, Yuji Nakatsukasa, D. Samaddar, Classification of chaotic time series with deep learning, Physica D 403 (2020) 132261.
[20] G.A. Gottwald, I. Melbourne, A new test for chaos in deterministic systems, Proc. R. Soc. 460 (2042) (2004) 603–611.
[21] Chandrachur Bhattacharya, Susheel Dharmadhikari, Amrita Basak, Asok Ray, Early detection of fatigue crack damage in ductile materials: A projection-based probabilistic finite state automata approach, ASME Lett. Dyn. Syst. Control (2021).
[22] Kyu Tae Kim, Dom A. Santavicca, Interference mechanisms of acoustic/convective disturbances in a swirl-stabilized lean-premixed combustor, Combust. Flame 160 (8) (2013) 1441–1457.
[23] Sudeepta Mondal, Najah F. Ghalyan, Asok Ray, Achintya Mukhopadhyay, Early detection of thermoacoustic instabilities using hidden Markov models, Combust. Sci. Technol. 191 (8) (2019) 1309–1336.
[24] Chandrachur Bhattacharya, Jacqueline O'Connor, Asok Ray, Data-driven detection and early prediction of thermoacoustic instability in a multi-nozzle combustor, Combust. Sci. Technol. (2020) http://dx.doi.org/10.1080/00102202.2020.1820495.
[25] D. Lind, B. Marcus, An Introduction To Symbolic Dynamics and Coding, second ed., Cambridge University Press, Cambridge, UK, 1995.
[26] V. Rajagopalan, A. Ray, Symbolic time series analysis via wavelet-based partitioning, Signal Process. 86 (11) (2006) 3309–3320.
[27] A. Subbu, A. Ray, Space partitioning via Hilbert transform for symbolic time series analysis, Appl. Phys. Lett. 92 (8) (2008) 084107.
[28] Najah F. Ghalyan, A. Ray, Measure invariance of ergodic symbolic systems for low-delay detection of anomalous events, Mech. Syst. Signal Process. 159 (2021) 107746 (1–19).
[29] A. Berman, R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM, Philadelphia, PA, USA, 1994.
[30] Rahul Dev Singh, Ajay Mittal, Rajesh K. Bhatia, 3D convolutional neural network for object recognition: a review, Multimedia Tools Appl. 78 (12) (2019) 15951–15995.
[31] Saeed Reza Mohandes, Xueqing Zhang, Amir Mahdiyar, A comprehensive review on the application of artificial neural networks in building energy analysis, Neurocomputing 340 (2019) 55–75.
[32] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.
[33] Daniel S. Levine, Introduction To Neural and Cognitive Modeling, Routledge, 2018.
[34] Alex Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, Physica D 404 (2020) 132306.
[35] A. Ray, R. Luck, An introduction to sensor signal validation in redundant measurement systems, IEEE Control Syst. Mag. (ISSN: 0165-1684) 11 (2) (1991) 44–49.
[36] M. Hénon, A two-dimensional mapping with a strange attractor, in: The Theory of Chaotic Attractors, Springer, 1976, pp. 94–102.
[37] J.M.T. Thompson, H.B. Stewart, Nonlinear Dynamics and Chaos, second ed., John Wiley and Sons, New York, USA, 1986.
[38] O.E. Rössler, An equation for continuous chaos, Phys. Lett. FrajA 57 (5) (1976) 397–398.
[39] E.N. Lorenz, Predictability: does the flap of a butterfly's wing in Brazil set off a tornado in Texas? in: American Association for the Advancement of Sciences, 139th Meeting, 1972.
[40] Lord Rayleigh, The Theory of Sound, Dover Publications, 1945.
[41] D. Jordan, P. Smith, Nonlinear Ordinary Differential Equations: An Introduction for Scientists and Engineers, 10, Oxford University Press on Demand, Oxford, U.K., 2007.
[42] H. Asghari, M. Dardel, Parameter converting method for bifurcation analysis of nonlinear dynamical systems, 27 (1) (2020) 310–329, Scientia Iranica.
[43] Chandrachur Bhattacharya, Asok Ray, Online discovery and classification of operational regimes from an ensemble of time series data, J. Dyn. Syst. Meas. Control 142 (11) (2020).