Data Fusion and Pattern Classification in Dynamical Systems via Symbolic Time Series Analysis

Xiangyi Chen and Asok Ray, *Fellow ASME*

***Abstract*—Symbolic time series analysis (*STSA*) plays an important role in the investigation of continuously evolving dynamical systems, where the capability to interpret the joint effects of multiple sensor signals is essential for adequate representation of the embedded knowledge. This technical brief develops and validates, by simulation, an *STSA*-based algorithm to make timely decisions on dynamical systems for information fusion and pattern classification from ensembles of multi-sensor time series data. In this context, one of the most commonly used methods has been neural networks (*NN*) in their various configurations; however, these *NN*-based methods may require large-volume data and prolonged computational time for training. An alternative feasible method is the *STSA*-based probabilistic finite state automata (*PFSA*), which has been shown in recent literature to require significantly less training data and to be much faster than *NN* for training and, to some extent, for testing. This technical brief reports a modification of the current *PFSA* methods to accommodate (possibly heterogeneous and not necessarily tightly synchronized) multi-sensor data fusion and (supervised learning-based) pattern classification in real time. Efficacy of the proposed method is demonstrated by fusion of time series of position and velocity sensor data, generated from a simulation model of the forced Duffing equation.**

***Index Terms*—Probabilistic Finite State Automata; Anomaly & Fault Detection; Forced Duffing equation.**

I. Introduction

Recently data-driven pattern classification by time series analysis has been extensively reported in open literature, for which several standard methods of machine learning (*ML*) are available (e.g., see [1] and references therein). Neural networks (*NN*) in their various configurations [2]–[4] have apparently become one of the most popular methods for time-series-based pattern classification. However, many of these *NN* techniques may not be suitable for real-time detection & classification in evolving dynamical systems, because the training time could be too large for such applications, or because the available training data might not be adequate. In this context, Bhattacharya and Ray [5] have reported the classification of different regimes in models of chaotic dynamical systems as well as in real-life chaotic systems (e.g., combustion processes), based on time-series analysis of single-sensor data. This classification tool needs to be extended for the usage of multi-sensor data to (possibly) achieve better classification performance; however, multiple (i.e., two or more) sensors may require generation of joint probability density functions, which is often computationally expensive for real-time applications even if only two sensors are used.

Copula [6], [7] is one such method that is capable of generating (multivariate) joint probability density functions by combining the marginal densities of individual sensors with a postulated kernel function. In this context, Iyengar et al. [8] reported copula-based fusion of audio and video signals to obtain the joint density for anomaly detection in a binary classification problem. Another alternative feasible approach to data fusion is mutual information-based [9] treatment of multiple-sensor time series, which also requires computation of joint probability density functions. To this end, Sarkar et al. [10] reported anomaly detection in a swirl-stabilized combustor by making use of the mutual information between pressure and temperature signals, where the concept of symbolic time series (*STSA*) was applied for analyzing the probabilistic finite state automata (*PFSA*) [11], [12].

This technical brief proposes a real-time analytical method for sensor data fusion as well as the associated problem of (supervised learning-based) pattern classification from ensembles of (possibly heterogeneous and not necessarily tightly synchronized) multi-sensor time series data. The proposed method, which does not require the computation of joint probability density functions, is an extension of the single-sensor *STSA*-based *PFSA* algorithm of Bhattacharya and Ray [5] for real-time execution. The underlying algorithm has been validated by simulation on a model of the forced Duffing equation [13].

II. Probabilistic Finite State Automata

This section introduces the basic concepts of probabilistic finite state automata (*PFSA*) [11], [12] in the setting of symbolic time series analysis (*STSA*) [14], [15].

A. Background and Mathematical Theory

For *PFSA*-based signal analysis, the ensemble of observed time-series data from a homogeneous (i.e., statistically stationary) Markov chain are partitioned into a finite number of cells. Then, the time series is symbolized, where a symbol represents the cell in which the signal data-point lies. This process converts the (continuous-valued) time series into a string of symbols, where each symbol in the string belongs to a (finite-cardinality) alphabet[1] [16], [17]. The final step

Xiangyi Chen was with the Department of Nuclear Engineering, Pennsylvania State University, University Park, PA 16802, USA and is currently with the Department of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou, China (e-mail: xchen909@outlook.com).

Asok Ray is with the Department of Mechanical Engineering and the Department of Mathematics, Pennsylvania State University, University Park, PA 16802, USA (e-mail: axr2@psu.edu).

[1]The cardinality of a (necessarily finite) alphabet of symbols is equal to the number of cells used for quantization of signal data points. In other words, each cell in the partition uniquely represents a symbol.

in this process is the construction of probabilistic finite state automata (*PFSA*). The following definitions [5], [11], [12] are introduced here for completeness of this technical brief.

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..*
- *$\delta : Q \times \mathcal{A} \rightarrow Q$ 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 an 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 $\mathcal{A}^\ell$.*

Definition 3. *A probabilistic finite state automaton (PFSA) J is a pair (G, π), where:*

- *The (deterministic) finite state automaton (FSA) G is called the underlying FSA of the PFSA J.*
- *The probability map $\pi : Q \times \mathcal{A} \rightarrow [0, 1]$ 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 [18], [19]). In that case, the PFSA can be represented as a quadruple $J = (\mathcal{A}, Q, \delta, \Pi)$.*
- *The state transition probability function $\tau : Q \times Q \rightarrow [0, 1]$ is constructed by combining the map δ and the matrix 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 $J = (\mathcal{A}, Q, T)$.*
- *The sum-normalized left-eigenvector of such an ergodic[2] matrix T, corresponding to the (unique) eigenvalue 1, is the (positive) state probability vector, p, of the PFSA. This is guaranteed by stochasticity and ergodicity [18] of T.*

Prior to partitioning, the ensemble of time series for each sensor is normalized to have zero mean and unity variance to remove any bias from the signal and to ensure that a fixed set of partition boundaries can be used across the complete range of time-series data for training [5]. A fixed partitioning ensures that there is no need to recompute the partitioning boundaries at every step, which is important for real-time applications. This procedure also allows different *PFSA*s to be compared, which is essential for dynamically representing the evolving physical process for real-time pattern classification.

[2]A homogeneous finite-state Markov chain is called ergodic if each state of the chain can move to any state of the chain, including itself, in finitely many transitions. In other words, its state transition probability matrix is ergodic if and only if ($q_i \rightarrow q_j \Leftrightarrow q_j \rightarrow q_i \; \forall i, j$ in finitely many transitions) [18].

B. D-Markov Machines

The *PFSA* structure of a D-Markov machine generates symbol strings $\{s_1 s_2 \cdots s_\ell : \ell \in \mathbb{N}$ and $s_j \in \mathcal{A}\}$ based on the underlying homogeneous Markov chain. The construction of a D-Markov machine assumes that generation of the next symbol depends only on a *finite* history of the last D or less consecutive symbols, i.e., the (most recent) symbol block of length not exceeding D. A D-Markov machine is defined as:

Definition 4. *A D-Markov machine [11], [12] is a PFSA in the sense of Definition 3 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 Markov chain $S = \cdots s_{-1} s_0 s_1 \cdots$, 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.

A numerical procedure to generate the morph matrix Π from a finite-length symbol string follows [11].

Given a fixed alphabet size $|\mathcal{A}|$ and depth D of a D-Markov machine, 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 in Eq. (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 a clearly defined morph matrix Π and the associated state transition probability matrix T; both of these matrices must be (element-wise) strictly positive. The ergodicity and stochasticity [18], [19] of both Π and T are guaranteed by this construction.

Remark 5. *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 (e.g,, $|Q| = |\mathcal{A}|$); therefore, the morph matrix Π and the state transition probability matrix T become indistinguishable if $D = 1$.*

III. Development of the Underlying Algorithms

This section develops the algorithms of data fusion and pattern classification from ensembles of time series data, generated from multiple (possibly heterogeneous and not necessarily tightly synchronized) sensors. Let there be m (≥ 2) sensors that generate the above ensembles of time series of vector data $\{x_k^j\}$, where the superscript $j \in \{1, \cdots, m\}$ points to the specific sensor in the array, and the subscript $k \in \mathbb{N}$

indicates the (possibly slightly approximate) time instants at which the data are collected.

The individual $PFSA$s are now constructed for each of the m sensors from the respective (normalized) time series $\{x_k^j\}$ with (possibly different) partitioning schemes for individual sensors. In this technical brief, the maximum entropy partitioning (MEP) (or possibly uniform partitioning (UP) [16]) has been used for symbolization of time series. Consequently, since there are no iterative loops in the PFSA algorithms, the execution of both training and classification (or testing) algorithms should be reasonably fast for real-time applications.

Two constraints are imposed for the $PFSA$ construction from the time series of each of the m sensor data sets: (i) identical alphabet size $|\mathcal{A}| > m$ and (ii) depth $D = 1$. While the first constraint assures dimensional compatibility of the matrices and vectors among different classes, the second constraint reduces computational complexity of the underlying algorithms for real-time applications. The following two problems are presented below in the $PFSA$ framework.

Problem 1: Fusion of the information extracted from the data set of m (≥ 2) sensors, or a selected subset of this set, without the need of computing their joint probability density functions (e.g., from individual marginal density functions).

Problem 2: Real-time (supervised learning-based) pattern classification of test data by using the knowledge base of training data, generated from the aforesaid m sensors, into exactly one of the following C classes: $c \in \mathcal{C} \triangleq \{0, 1, \cdots, (C-1)\}$, where the class $c = 0$ is the nominal class, and any individual anomaly is designated by a specific class, $c > 0$.

To address the above two problems, algorithms are built upon the concept of orthogonal projection of the emerging real-time information from the test data onto the null-space of the matrix formed by stacking the $(1 \times |\mathcal{A}|)$ state probability vectors generated from the ensemble of training data by using all of the m sensor data. This action results in the construction of an $(m \times |\mathcal{A}|)$ matrix, for each of the C classes, i.e., for each $c \in \mathcal{C}$, in the training phase. It is noted that, in this technical brief, these matrices have been obtained by simulation of individual sensor time series for all $c \in \mathcal{C}$ classes. The key idea is succinctly presented below as:

> *From the (training) time series of each pattern class, a hyperplane is constructed in the feature space. The feature vector, generated from the sensor time series of test data to be classified, is projected onto each hyperplane for thresholdless classification [20].*

The above concept is elaborated below.

Training phase: In this technical brief, the training phase is conducted based on the (simulated) sensor time series for each class $c \in \mathcal{C}$. The training algorithm is built upon an ensemble of (normalized) time series from each of the C classes, based on the user-specified parameters, such as alphabet size $|\mathcal{A}| > m$, and depth $D = 1$. The morph matrix for each class $c \in \mathcal{C}$ is computed as Π_{trn-c}, for $c = 0, \cdots, (C-1)$; in this setting, the individual projection matrix for each class is obtained from the respective morph matrices as explained below.

From the $PFSA$s, belonging to each of the m sensors, the m state probability vectors are constructed, each of which is a $(1 \times |\mathcal{A}|)$ vector. These vectors are stacked together to form an $(m \times |\mathcal{A}|)$ matrix H_{trn-c} of rank m, corresponding to each class $c \in \mathcal{C}$, as explained later in Remark 6. Then, the $(|\mathcal{A}| \times |\mathcal{A}|)$ orthogonal projection matrix onto the null space of H_{trn-c} is constructed for each class $c \in \mathcal{C}$ as:

$$\mathcal{P}_{trn-c} \triangleq I_{|\mathcal{A}|\times|\mathcal{A}|} - H_{trn-c}^T \Big[H_{trn-c} H_{trn-c}^T \Big]^{-1} H_{trn-c} \quad (2)$$

Classification (or test) phase: In the classification phase, the task is to identify, in real time, the unknown class $c \in \mathcal{C}$ to which the vector time series under test is expected to belong. The test time-series is symbolized by using the same partitioning technique and the same parameters (e.g., the same $|\mathcal{A}| > m$ and $D = 1$) as in the training phase. Accordingly, the resulting morph matrix Π_{tst}^j and the corresponding $(m \times |\mathcal{A}|)$ test matrix H_{tst}, constructed from the m sum-normalized state probability vectors $p_{tst}^j,\ j = 1, \cdots, m$, of the m sensors, are generated by following the same procedure as in the training phase. Now the $(m \times |\mathcal{A}|)$ matrix H_{tst} is projected onto the null space of H_{trn-c} for each class $c \in \mathcal{C}$ resulting in C different $(m \times |\mathcal{A}|)$ projected matrices $H_{tst}\mathcal{P}_{trn-c}$ (see Eq. (2)) for each $c \in \{0, 1, \cdots,\ (C-1)\}$. A class $c^\star$ that yields the smallest norm of the $(m \times |\mathcal{A}|)$ matrix $H_{tst}\mathcal{P}_{trn-c^\star}$ is identified to be the class to which the test data are expected to belong, i.e.,

$$\text{Identified Class } c^\star = \underset{c \in \{0,1,\cdots,\ (C-1)\}}{\operatorname{argmin}} ||H_{tst}\mathcal{P}_{trn-c}|| \quad (3)$$

where the norm, used in Eq. (3) is the induced 2-norm (i.e., the norm $\| X \| \triangleq$ square root of the maximum singular value of $X^T X$): however, the type of the norm is not critical here because of the norm equivalence in finite-dimensional vector spaces. It is also noted that the above decision-making in Eq. (3) is thresholdless [20].

Remark 6. *The $(1 \times |\mathcal{A}|)$ state probability vector, p_{trn-c}^j, is associated with the respective sensor j in each class $c \in \mathcal{C}$. Even if the maximum entropy partitioning (MEP) [16] is used for all of the m sensor time series, the generated set of m row vectors p_{trn-c}^j's in H_{trn-c} should be mutually linearly independent (i.e., the $(m \times m)$ matrix $H_{trn-c}H_{trn-c}^T$ being invertible) for each $c \in \mathcal{C}$, because of the following reasons:*

- *The partitioning is done with a modestly large ensemble of training data belonging to all C different classes.*
- *The individual p_{trn-c}^j's are generated for each class $c \in \mathcal{C}$ from smaller segments of training data, which are also contaminated with unavoidable sensor noise [15].*

However, as a very rare event, if the above assertion is not true (e.g., the inversion of the $(m \times m)$ matrix $H_{trn-c}H_{trn-c}^T$ causes a numerical problem), then either an alternative data partitioning tool could be used to make the rank of $H_{trn-c} = m$, or the classes should be redefined in the training phase.

IV. Validation on a Chaotic Simulation Model

The process model in this simulation task is developed on a second-order forced Duffing equation [12], [13], which represents the non-autonomous dynamics of a mass-spring-damper system, equipped with a nonlinear spring having a cubic stiffness coefficient, under a harmonic excitation, with initial conditions $y(0) = 0$ and $\dot{y}(0) = 0$, as described below:

$$\ddot{y}(t) + \beta\ \dot{y}(t) + \alpha_1\ y(t) + \alpha_3\ (y(t))^3 = A\ \cos(\Omega t) \tag{4}$$

where the unit of time is *second* (abbreviated as 's' in the sequel), y is the position, $\dot{y}$ is the velocity, $\ddot{y}$ is the acceleration, and Ω is the angular frequency (in $radians/s$) of the input excitation; and the parameters $\alpha_1 = 1.0$, $\alpha_3 = 1.0$, $A = 22.0$, and $\Omega = 5.0$ are held fixed for all simulation runs, and the dissipation parameter β has different constant values in the range of 0.10 to 0.35 at an increment of 0.05 for individual simulation runs, similar to what was done in [12].

The second-order non-autonomous state-space model is constructed with the phase variables $x_1^t \triangleq y(t)$ and $x_2^t \triangleq \dot{y}(t)$. In the simulation runs, the fourth-order Runge-Kutta method of numerical integration [21] has been used with a (fixed) step size of 1 ms (i.e., 0.001 s). The measurement model consists of two sensors that provide the measurements of (noise-contaminated) time series data for y and $\dot{y}$, respectively. In order to emulate a real-life physical environment, two kinds of noise are injected into the dynamical system model [22] as described below.

- *Additive process noise*: Stationary white Gaussian noise $w^t \triangleq A_p N(0,1)$, to emulate the effects of uncertainties due to unmodeled dynamics, where $N(0,1)$ represents zero-mean unit variance Gaussian noise, and the parameter A_p in the process noise model is kept constant for individual simulation runs.
- *Multiplicative sensor noise*: Stationary white Gaussian noise $v_1^t \triangleq A_m N(0,1)$ and stationary white Gaussian noise $v_2^t \triangleq A_m N(0,1)$ have been generated from different seed numbers of a random number generator to emulate the effects of (statistically independent) measurement uncertainties in the position and velocity sensors, respectively. Usually instrumentation manufacturers' specifications guarantee not to exceed a specified fraction, A_m, of the average measured value within a given statistical confidence. The parameter A_m in both sensor noise models is kept constant for individual simulation runs.

Process Model (injected with additive noise):

$$\dot{x}_1^t = x_2^t \tag{5}$$
$$\dot{x}_2^t = -\alpha_1 x_1^t - \alpha_3 (x_1^t)^3 - \beta x_2^t + A\ \cos(\Omega t) + w^t \tag{6}$$

Measurement Model (injected with multiplicative noise):

$$z_1^t = x_1^t(1 + v_1^t) = x_1^t + x_1^t\ v_1^t \tag{7}$$
$$z_2^t = x_2^t(1 + v_2^t) = x_2^t + x_2^t\ v_2^t \tag{8}$$

The injected process noise and sensor noise make the model of system dynamics behave as a random process, which is represented by the (stochastic) forced Duffing system in Eqs. (5) to (8). Simulated time series plots of y have been displayed in a previous publication [12] for the same model parameters including different constant values of the dissipation parameter β. Since these plots have similar trends as the current simulation plots, they are not presented in this technical brief. However, Figure 1 displays a family of phase plots for the six different values of β in the absence and presence of both process noise and sensor noise.

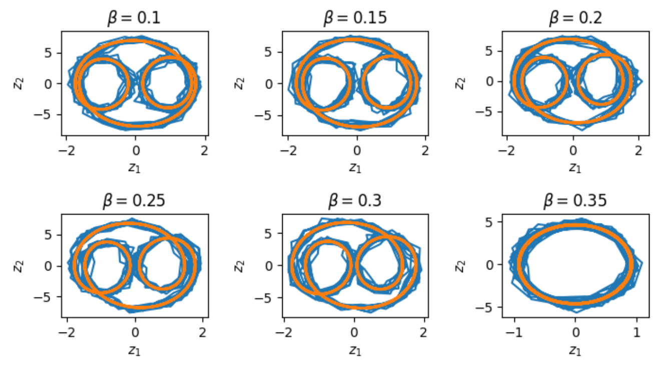

Fig. 1. Phase-space trajectories for the forced Duffing equation with $\beta = 0.10, 0.15, 0.20, 0.25, 0.30$, and 0.35; $\alpha_1 = 1.0; \alpha_3 = 1.0; \Omega = 5.0$; and $A = 22.0$. The smooth trajectories (in red) are without any noise (i.e., $A_p = 0$ and $A_m = 0$); the wrinkled trajectories (in blue) are with additive process noise $w^t \triangleq A_p N(0,1)$, where $A_p = 0.20$, and multiplicative sensor noise $v_1^t \triangleq A_m x_1^t N(0,1)$ and $v_2^t \triangleq A_m x_2^t N(0,1)$, where $A_m = 0.05$.

V. Results of Algorithm Validation by Simulation

To validate the results of the data fusion and (supervised learning-based) pattern classification algorithm, six pattern classes (i.e., $C = 6$) are defined in the training phase corresponding to six different values of the dissipation parameter $\beta = 0.1, 0.15, 0.20, 0.25, 0.3$, and 0.35 in the model of the forced Duffing system in Eqs. (5) to (8). All simulated data are down-sampled to have a fixed interval of 0.1 s in the time series to be analyzed, where each sample of the time-series data contains such 500 data points, implying that the time window size is 50 s. Since the effects of window size selection on classification accuracy have been reported in details earlier (e.g., see [5], [11], [19]), such analysis is not repeated here.

The training set consists of 6,000 samples (of stationary oscillations) of time sereis (i.e., data acquired after the initial transients have died out), where each class contains 1,000 samples; similarly, each set of test data consists of 1,000 samples. Simulations were also conducted with different numbers of training data in each of the six classes as well as different numbers of testing data points. Since these results were largely similar, on the average, and no specific conclusions could be drawn from these results, they are not reported here.

In each of the above simulation runs, maximum entropy partitioning (MEP) [16] was used for symbol generation, and the depth of the corresponding D-Markov machine (see Definition 4 in Section II) was set as $D = 1$. The alphabet size was selected to be $|\mathcal{A}| = 6$, which would allow up to 5 sensors to be included in the measurement set. A larger value of alphabet size (i.e., $|\mathcal{A}| > 6$) was found to yield no appreciable increase in the classification performance and may require longer data strings, as explained in [11].

The classification performance is summarized in Table I. These results suggest that the proposed data fusion may indeed improve the classification performance, even in the presence of process noise, provided that the amplitude of the sensor noise is sufficiently low, which necessitates the usage of high-quality sensors. The following observation is made from the results presented in Table I,

As the quality of sensors degrades (e.g., the sensors become more noisy), the classification performance

TABLE I
CLASSIFICATION ACCURACY UNDER DIFFERENT SENSORS WITH ADDITIVE PROCESS NOISE AND MULTIPLICATIVE SENSOR NOISE

Multiplicative sensor noise	Additive process noise								
	$A_p = 0.000$			$A_p = 0.100$			$A_p = 0.200$		
	y	$\dot{y}$	y and $\dot{y}$	y	$\dot{y}$	y and $\dot{y}$	y	$\dot{y}$	y and $\dot{y}$
$A_m = 0.000$	96.0%	87.3%	**99.7%**	91.7%	86.0%	**97.5%**	86.4%	83.1%	**93.8%**
$A_m = 0.005$	94.5%	85.3%	**98.6%**	91.4%	83.5%	**97.2%**	86.1%	81.3%	**92.9%**
$A_m = 0.010$	92.6%	81.4%	**97.1%**	90.4%	80.5%	**95.3%**	86.3%	78.5%	**91.3%**
$A_m = 0.020$	90.2%	73.3%	**92.8%**	89.5%	71.9%	**92.2%**	85.3%	71.6%	**89.4%**
$A_m = 0.030$	88.7%	65.9%	**89.3%**	87.2%	65.0%	**88.8%**	85.1%	64.7%	**85.6%**
$A_m = 0.040$	**86.0%**	57.7%	83.5%	**85.5%**	57.7%	84.3%	**83.8%**	57.7%	83.0%
$A_m = 0.050$	**83.5%**	50.4%	80.6%	**84.1%**	50.5%	79.7%	**81.8%**	49.7%	79.1%

may deteriorate, because the additional sensor may bring in less useful (and possibly harmful) information. In other words, data fusion and pattern classification with additional (inferior-quality) sensors may actually degrade the performance. Therefore, a decision to use the information from an additional channel of sensor data should be carefully made for data fusion, especially if these additional sensors are not of very good quality. Hence, it is recommended to avoid the usage of low-precision sensor data, because the imprecise information added by such sensors may actually poison the information generated by the remaining good sensors, instead of improving the classification performance.

Remark 7. *The above observation of possible degradation of the classification performance may not apparently agree with the classical notion of mutual information that is always non-negative [9]. This fact implies that the addition of a low-quality sensor should not reduce the total information, regardless of how noisy the sensor is. This apparent anomaly is explained below from an information-theoretic point of view.*

*The mutual information $I(s_1, s_2)$ is the relative entropy between the joint distribution, $p(s_1, s_2)$ of time series of two sensors (s_1 & s_2) and their product distribution $p(s_1)p(s_2)$ [9]. This notion is conceptually different from that of pattern classification (e.g., see **Problem 2** in Section III) that addresses identification of the class to which test data belong. Therefore, future research is recommended in Section VI on information-theoretic aspects of both data fusion and pattern classification.*

1) Computation Time: The simulation runs were conducted on a personal computer running with Intel I9-1085K CPU in the Python3.8 environment. In this setting, the computation time for a typical training sample was ~ 250 ms, while that for classification of a typical single test sample was ~ 9 ms.

VI. SUMMARY, CONCLUSIONS, AND FUTURE WORK

This technical brief has developed a concept of real-time data fusion and (supervised learning-based) pattern classification from ensembles of (possibly heterogeneous and not necessarily tightly synchronized) multi-sensor time series data, where the usage of high-quality sensors is recommended. The reported work is an extension of an earlier similar concept that was validated for (real-life) combustion data of single-sensor time series from an experimental apparatus [5].

The proposed concept of multi-sensor data fusion and pattern classification has been validated in this technical brief by simulation on a dynamic model of the forced Duffing equation that may exhibit chaotic behavior and bifurcation as a model parameter is perturbed. The underlying theory is built upon symbolic time series analysis (*STSA*)-based probabilistic finite state automata (*PFSA*) [11], [12]. Since the algorithm does not require the construction of joint probability density functions from the available multiple-sensor measurements, the underlying algorithms are computationally efficient and are ideally suited for real-time decision-making. Specifically, the proposed method does not require a large ensemble of training data and the training time is significantly smaller than that of a typical neural network [2]–[4] to perform similar tasks [5].

While there are many areas of theoretical and experimental research to improve the proposed method of (real-time) multi-sensor data fusion and pattern classification, the following six topics are suggested by the authors for future research:

1) *Quantitative comparison (e.g., accuracy, robustness, and computational complexity) of data fusion & classification performance of the proposed method with those of deep neural networks in their different configurations*: This research is necessary to demonstrate that the proposed method is a competitive tool of machine learning.
2) *Information-theoretic research for accommodation of low-quality sensors*: This research is necessary to enhance the proposed method as a tool of machine learning that is compatible with a variety of inexpensive sensors and quasi-reliable sources of information.
3) *Handling mixed synchronous and asynchronous information*: This research is necessary to enhance the capability of the proposed data fusion & pattern classification method by taking advantage of different available sources of relevant mixed information (e.g., audio data, video images, and text messages).
4) *Learning dynamically changing data that are not restricted to be statistically quasi-stationary [19]*: This research is necessary to improve the proposed data fusion & classification under transient operations.
5) *Augmentation of pattern classification to include unsupervised learning*: This research is necessary for situations, where the classes need to be defined autonomously instead of prior class allocation by the user.
6) *Experimentation for concept validation in various physical applications*: This research is necessary for acceptance of the proposed method for (real-life) scientific & industrial applications.

Acknowledgments

The work reported in this technical brief has been supported in part by the U.S. Army Research Office (Grant No. W911NF-20-1-0226) and by the U.S. National Science Foundation (Grant no. CNS-1932130). Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

References

[1] K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA, USA, 2012.

[2] S. R. Mohandes, X. Zhang, and A. Mahdiyar, "A comprehensive review on the application of artificial neural networks in building energy analysis," *Neurocomputing*, vol. 340, pp. 55–75, 2019.

[3] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[4] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar, "Classification of chaotic time series with deep learning," *Physica D: Nonlinear Phenomena*, vol. 403, p. 132261, 2020.

[5] C. Bhattacharya and A. Ray, "Thresholdless classification of chaotic dynamics and combustion instability via probabilistic finite state automata," *Mechanical Systems and Signal Processing*, vol. 154, pp. 108213 (1–18), March 2022.

[6] P. Trivedi and D. Zimmer, "Copula modeling: An introduction for practitioners," *Foundations and Trends in Econometrics*, vol. 1, no. 1, pp. 1–111, 2005.

[7] R. Nelson, *An introduction to Copulas, 2nd ed.* Springer, New York, NY, USA, 2006.

[8] S. Iyengar, P. Varshney, and T. Damarla, "A parametric copula-based framework for hypothesis testing using heterogeneous data," *IEEE Trans. Signal Processing*, vol. 59, no. 5, pp. 2308 – 2319, 2011.

[9] T. Cover and J. Thomas, *Elements of Information Theory, 2nd ed.* Hoboken, NJ, USA, 2006.

[10] S. Sarkar, S. Chakravarthy, V. Ramanan, and A. Ray, "Dynamic data-driven prediction of instability in a swirl-stabilized combustor," *International Journal of Spray and Combustion Dynamics*, vol. 8, no. 4, pp. 235 – 253, 2016.

[11] K. Mukherjee and A. Ray, "State splitting and merging in probabilistic finite state automata for signal representation and analysis," *Signal Processing*, vol. 104, pp. 105 – 119, 2014.

[12] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Signal Processing*, vol. 84, no. 7, pp. 1115 – 1130, 2004.

[13] J. Thompson and H. Stewart, *Nonlinear Dynamics and Chaos*. Wiley, Hoboken, NJ, USA, 2 ed., 2002.

[14] C. Daw, C. Fenney, and E. Tracy, "A review of symbolic analysis of experimental data," *Review of Scientific Instruments*, vol. 74, pp. 915–930, February 2003.

[15] P. Beim Graben, "Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics," *Physical Review E*, vol. 64, no. 5, p. 051104, 2001.

[16] V. Rajagopalan and A. Ray, "Symbolic time series analysis via wavelet-based partitioning," *Signal Processing*, vol. 86, pp. 3309–3320, November 2006.

[17] A. Subbu and A. Ray, "Space partitioning via hilbert transform for symbolic time series analysis," *Applied Physics Letters*, vol. 92, p. 084107, February 2008.

[18] A. Berman and R. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*. SIAM Press, Philadelphia, PA, USA, 1994.

[19] N. F. Ghalyan and A. Ray, "Measure invariance of ergodic symbolic systems for low-delay detection of anomalous events," *Mechanical Systems and Signal Processing*, vol. 159, pp. 107746 (1–19), April 2021.

[20] A. Ray and R. Luck, "An introduction to sensor signal validation in redundant measurement systems," *IEEE Control System Magazine*, vol. 11, no. 2, pp. 44–49, 1991.

[21] R. Burden and J. Faires, *Numerical Analysis*. Cengage Learning, New Delhi, India, 2010.

[22] D. Simon, *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. Wiley, Hoboken, NJ, USA, 2006.