Received 30 January 2023, accepted 14 February 2023, date of publication 27 February 2023, date of current version 17 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3249666

RESEARCH ARTICLE

Real-Time Power System Event Detection: A Novel Instance Selection Approach

GABRIEL INTRIAGO[1,2], (Student Member, IEEE), AND YU ZHANG[1], (Member, IEEE)
[1]Department of Electrical and Computer Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064, USA
[2]Department of Mathematics, Escuela Politécnica Nacional, Quito 170525, Ecuador

Corresponding author: Yu Zhang (zhangy@ucsc.edu)

This work was supported in part by the Seed Fund Award from Center for Information Technology Research in the Interest of Society (CITRIS); in part by the Banatao Institute, University of California; and in part by the Hellman Fellowship.

ABSTRACT This study presents a novel adaptation of the Hoeffding Adaptive Tree (HAT) classifier with an instance selection algorithm that detects and identifies cyber and non-cyber contingencies in real time to enhance the situational awareness of cyber-physical power systems (CPPS). Wide-area monitoring, protection, and control (WAMPAC) systems allow system operators to operate CPPS more efficiently and reliably. WAMPAC systems use intelligent devices such as phasor measurement units (PMUs) to monitor the CPPS state. However, such devices produce continuous and unbounded data streams, posing challenges for data handling and storage. Moreover, WAMPAC devices and the communication links connecting them are vulnerable to cybersecurity risks. In this study, we consider several cyber and non-cyber contingencies affecting the physics and monitoring infrastructure of CPPS. Our proposed classifier distinguishes disturbances from cyberattacks using a novel instance selection algorithm with three algorithmic stages to ease data management. A cost and complexity analysis of the algorithm is discussed. With reduced computational effort, the classifier can handle high-velocity, high-volume, and evolving data streams from the PMUs. Six case studies with extensive simulation results corroborate the merits of the proposed classifier, which outperforms state-of-the-art classifiers. Moreover, the classifier demonstrated a high performance using a dataset outside the contingency detection domain. Finally, the real-time applicability of the proposed methodology is assessed, and its limitations are discussed.

INDEX TERMS Classification, cyber-attacks, disturbances, instance selection, streaming data.

I. INTRODUCTION

The deployment of intelligent monitoring devices has led to significant transformations of electric power systems from purely physical systems into cyber-physical power systems (CPPS) [1]. The CPPS consists of a physical power system jointly integrated with cyberinfrastructure for communication, control, protection, and monitoring. A CPPS allows the bidirectional flow of electricity and information to enable smart grid technologies and operate the electric power grid more efficiently and reliably [2]. Despite the benefits of the CPPS, their main shortcoming is that CPPS are vulnerable to cybersecurity threats. Numerous incidents of successful cyberattacks on power grids have occurred worldwide. For example, the slammer worm of the Davis-Besse nuclear plant in Ohio in 2003 [3], the Stuxnet worm that attacked the monitoring system of a nuclear power plant in Iran in 2009 & 2010 [4], the cyberattack on an electric power facility in Ukraine in 2015 [5], or the cyberattacks at western US power grid facilities for about ten hours in 2019 [6]. Cyberattacks can severely impact a country's economy by causing economic losses due to the interruptions in the electricity supply [7]. Therefore, it is necessary to develop classification systems that can timely and accurately distinguish between non-cyber and cyber contingencies.

The CPPS trend towards automated systems using supervisory, control, and data acquisition systems (SCADA) or wide area monitoring protection and control (WAMPAC) systems

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

to provide near real-time capabilities across the grid [2]. Fig. 1 shows a generic diagram of the power system monitoring architecture. The SCADA system provides steady-state observability to system operators and can be applied to systems with slow dynamics [8]. In contrast, WAMPAC systems provide transient observability to operators suitable for the power grid's actual conditions with fast dynamics, owing to the high penetration of inverter-based resources [9]. In this study, we consider a CPPS monitored using a WAMPAC system. A WAMPAC systems include several connected devices that are potentially subject to cyberattacks. In addition, these devices produce incessant data streams at high speed, posing significant challenges for data storage, handling, and management. Hence, a classification system that detects CPPS behaviors that deviate from normal operation must handle continuous and unbounded streams of data while using limited memory and without affecting its classification performance.

It is reasonable to consider the CPPS as a real-time non-stationary environment in which the feature or target space evolves in time, and many factors, such as disturbances and cyberattacks, may affect the steady-state operating point of the CPPS. According to the stream learning lexicon, such factors are called concept drifts [10]. A concept is the target information a model tries to predict using a set of features, whereas a drift is the change in the concept over time [11]. In evolving environments, the trained model is frequently outdated owing to existing concept drifts. Thus, the model must be retrained continuously to keep track of the current concept that underlies the data distribution. With continuous unbounded data streams, the retraining procedure must be executed in a single pass and as soon as a new instance arrives. In addition, it is indispensable to design an effective instance selection policy that selects the most relevant instances from the original stream to ease data handling. Streaming learning algorithms address the previously mentioned problems and can handle real-time applications for CPPS. Such applications may be too large for traditional machine-learning techniques. In this paper, we propose a streaming learning classification system with an instance selection policy for contingency detection in CPPS. Our proposed approach can retrain, forget, and detect concept drifts in evolving data streams from the PMUs.

The existing solutions for the contingency detection task are model-based, machine learning, neural networks (NN), and streaming learning approaches. The list of available solutions presented in Table 1 is by no means exhaustive, but it indicates that many research projects have taken place. Event detection in power systems is challenging because most events are infrequent, the system's measurements are noisy, and the system's operating point changes in real time. Traditionally, model-based methods detect events by observing the deviation between the measurements of the real system and the model of the system. A significant deviation indicates that an event has occurred [13], [14]. In [12], the authors design a decentralized moving target defense framework to

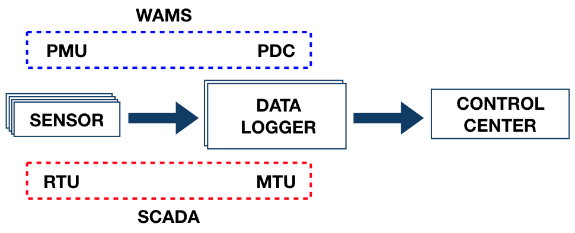

FIGURE 1. A generic diagram of infrastructure for power systems monitoring. The wide area monitoring systems (WAMS) architecture relies on phasor measurement units (PMUs) and phasor data concentrators (PDCs) to generate and acquire real-time data, respectively. The supervisory control and data acquisition (SCADA) systems obtain data from remote terminal units (RTUs), which are collected and sent to the control center by master terminal units (MTUs).

detect false data injection (FDI) attacks that randomly select a set of signal replicas to transmit information and the replicas that reach their intended destination. Nonetheless, the replica selection depends on the topology of the network.

The small-signal model of the system can be used to monitor the parametric sensitivity of the eigenvalues of the system against load-altering attacks, as shown in [15]. However, large-scale attacks may result in significant changes in the power grid state, demanding an analysis under generalized nonlinear grid models rather than the linearized small-signal model. In [16], the authors develop a framework that leveraged the system model and a cumulative sum detector to identify stealthy FDI attacks in an AC smart grid. A fractional-order state transition matrix was combined with iterative weighted least squares (IWLS) to describe the system dynamics. Nonetheless, the size of the transition matrix affects the speed at which a solution can be obtained using the IWLS technique. Despite the merits of these prior arts, these works face essential challenges, such as requiring a mathematical description of the system and its components, scaling to larger power networks, and being specific to certain events. In contrast, our work adopts a model-agnostic approach that can adapt to time-variant scenarios and discriminate events of multiple types.

Unlike model-based methods, machine learning and NN methods can operate without any information regarding the physics, and mathematical models governing the power system [17], [18], [22], [23], [24]. In [19], the authors design an anomaly detection method using multivariate Gaussian models from micro-phasor measurement units (μPMUs) to detect malicious attacks. Nevertheless, the efficacy of such a method is limited to only two types of attacks, transient and steady. Another approach presented in [20] developed a framework that combines wavelet denoising, maximum likelihood estimation, Kalman filtering, and clustering to accurately detect data anomalies. Although Gholami et al. designed a promising method, it did not show a favorable trade-off between computational time and accuracy. A collaborative machine learning-based framework for detecting attacks was proposed in [21]. This method can efficiently

TABLE 1. Related literature for contingency detection in power systems.

Category	Methodology	Limitations	Ref.
Model-based	Decentralized moving target defense	Network topology dependence	[12]
	Residual-based online detection	Requires a mathematical model	[13]
	Stable kernel representation of the small-signal model	Analysis at single operating point	[14]
	Eigenvalue sensitivity and second-order systems	Linearized small-signal model	[15]
	Cumulative sum test and fractional order theory	Two types of events considered	[16]
Machine Learning	Non-nested exemplars and state extraction method	Combinatorial explosion	[17]
	Naive Bayes, Decision Trees, SVMs, and Boosting	Hyperparameter tuning	[18]
	Gaussian models	Two types of events considered	[19]
	Ensembles, maximum likelihood estimation and DBSCAN	Trade-off between time and accuracy	[20]
	Principal component analysis and clustering	Lack of retraining mechanism	[21]
Neural Networks	Deep neural networks	Lack of retraining mechanism	[22]
	Neural Networks and Principal Component Analysis	Computational burden	[23]
	Time-series analysis and recurrent neural networks	Training multiple NNs for DERs	[24]
	Deep sequence learning	Limited types of events	[25]
	Federated learning and generative adversarial networks	Lack of retraining mechanism	[26]
Stream Learning	Hoeffding Adaptive Tree classifier	Two types of events considered	[27]
	Hoeffding Adaptive Tree and drift detectors	Moderate accuracy performance	[28]
	Transfer Learning and Hoeffding Adaptive Tree	Limited types of events	[29]
	Semisupervised learning and online dictionary learning	Computational burden	[30]
	Ensemble learning	Single type of attack considered	[31]

extract patterns from attack vectors. However, this method is limited to FDI attacks and does not incorporate a mechanism to adapt the model to evolving scenarios. Our proposed classifier includes an instance selection algorithm that retrains the model at low computational cost when multiple types of contingencies affect the CPPS.

In [25], the authors proposed an attack detection approach using a long short-term network trained using electric waveform data from current and voltage sensors in small microgrids. However, the network is trained offline, demanding vast computational effort, and the approach is limited to data-integrity attacks. Habibi et al. [24] combined time-series analysis with nonlinear autoregressive neural networks to study the effect of FDI attacks that attempt to affect accurate voltage regulation and current sharing by affecting the voltage and current sensors in DC microgrids. Nonetheless, this approach needs to train neural networks for each distributed energy resource in the microgrid, which imposes a high computational burden. Several attack-detection applications require neural networks to be trained centrally by moving data into a cloud server, compromising data privacy, and imposing a communication overhead. The authors in [26] overcame these concerns by leveraging federated and semi-supervised learning to design a trustworthy framework for detecting anomalies in power systems. Nevertheless, the method does not address model retraining in nonstationary scenarios. Machine learning and neural network methods have shown outstanding performance and flexibility in detecting events in power systems. However, their main shortcomings are the assumption of a static environment, inability to retrain the models in real-time, and high computational burden for model learning, especially for NN methods. The approach proposed in this work can detect changes in the underlying data distribution and retrain the model accordingly. The retraining process improves the accuracy of the model without adding a significant processing burden.

Motivated by the discussions above, researchers have turned to stream learning to tackle the event detection task for power systems. In [27], Dahal et al. used a Hoeffding adaptive tree (HAT) to classify the normal operation and electrical faults of a power system operating with load fluctuations. Dahal demonstrated the applicability and ability of HAT to classify power system events. However, the experiments in Dahal's study were not designed to handle multiple events. The authors in [28] take a step further and modified the HAT classifier to incorporate two change detectors, the drift detection method (DDM) and adaptive windowing (ADWIN), to classify binary, ternary, and multiple events, including disturbances and cyberattacks. The work of Adhikari et al. relies on a discretization of the dataset according to domain knowledge which may not be affordable in a real-time scenario. Moreover, the results report a noticeable performance for binary and ternary events, but the performance for the multiple events scenario was moderate. In this study, our proposed classifier exhibits high accuracy and computational cost when dealing with binary, ternary, or multiple events.

A method based on selecting the most promising features using gradient boosting trees to enhance the performance of several machine learning classifiers was presented in [32]. This study shows that feature engineering is a crucial consideration in event detection; however, the classifiers are trained to assume a stationary environment and may not capture the real-time changes occurring in the system. Intriago et al. [30] leveraged semi-supervised learning and online dictionary learning to build a new dataset based on a set of hidden representations to enhance the performance of the HAT classifier for the identification of disturbances and cyberattacks in power systems. This work shows that the performance of the HAT classifier is slightly better than that of previous studies, which is not compensated by the computational burden required to build the new dataset. A framework that dynamically detects and classifies cyberattacks was presented in [31]. The framework consists of three modules for detection, classification, and signal retrieval. Although the method has been proven to detect and classify events under harsh learning conditions, it only detects false data-injection attacks. Moreover, this method depends on vast amounts of historical data, which may not represent the current system state.

A. CONTRIBUTION

To overcome the aforementioned challenges, our proposed approach studies the adaptation of the HAT classifier with a novel instance selection algorithm to deal with real-time event detection task in power systems. Our approach is model-agnostic and relies on streaming learning to handle high-velocity and volume data streams with reduced computational effort. HAT uses an incremental decision tree classifier that learns from evolving data streams and handles data with concept drift. The instance selection algorithm facilitates data management by selecting the most relevant instances for the learning task, further reducing the computational burden and memory consumption. The instance selection algorithm is combined with the HAT classifier to deal with evolving PMU data streams by constantly retraining the classifier as soon as system changes are detected. The main contributions of this study are as follows:

- We propose an effective instance selection algorithm for contingency classification in power systems. Our focus is to select the instances most similar to the target instance by using a spatiotemporal distance function that adapts to the range of the PMU measurements. The proposed selection algorithm can generally determine the optimal subset of data instances in a nonstationary streaming environment.
- We develop a novel modification of the Hoeffding Adaptive Tree (HAT) classifier by combining it with the instance selection algorithm. The proposed classifier operates under stream learning requirements while maintaining low memory consumption and running time.
- We investigate scenarios impacting the system physics and its monitoring infrastructure, such as similar events with different loading schemes, the sudden disruption of a PMU, and similar measurements. Extensive simulation results corroborate the merits of the proposed classifier, which is used to test 37 power system events grouped into binary, ternary, and multiclass datasets.
- The proposed classifier was assessed using a price forecasting dataset to measure the performance of the classifier outside the event detection domain. The simulations demonstrate that our classifier outperforms its competitors while maintaining high accuracy and low time and memory consumption.

The remainder of this paper is organized as follows. Section II presents the concepts and background of stream learning. Section III presents the details of the instance selection algorithm and the proposed classifier. Section IV describes the experimental setup. Section V presents the case studies, simulations, and experiments of this study, followed by conclusions in section VII.

II. STREAM LEARNING

A. CLASSIFICATION FOR DATA STREAMS

Classification for data streams inherits a large number of problems from traditional machine learning. There are also new challenges, such as one-pass learning, limited processing time and memory, and changes in data distribution. In this study, we focus on data stream classification. Let $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{\infty}$ denote a data stream containing a set of labeled instances. At time t, $\boldsymbol{x}_t \in \mathbb{R}^m$ denotes the vector of m features while y_t is the corresponding class label. Let $\mathcal{X}$ represent the entire feature space and $\mathcal{Y}$ the class space. A classification algorithm learns a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ such that it can be used to predict the class label for a new instance. Traditional classification can load all the data into memory. By contrast, stream classification is a one-pass strategy, meaning that the processed instances are automatically discarded or stored temporarily.

A learning model should meet the following criteria to comply with data stream learning [33]:

- Learn an instance at a time and inspect it at most once.
- The model must use a limited amount of memory.
- The working time is limited.
- The model must be able to predict at any time.

B. CONCEPT DRIFT

In the stream learning lexicon, *concepts* are defined as the target information that a model aims to predict using a set of features [34]. Data streams are inherently infinite, temporal, and dynamic. The data distribution may evolve, whereas the mapping between instances and targets can be time-varying. This situation results in the *concept drift* phenomena [10].

As a particular case of concept drift, feature drift occurs when a subset of features becomes irrelevant to a learning task [34]. In this study, the features are the PMU measurements, such as voltages, currents, and impedances. Feature drift in the context of power systems includes the following: (i) the removal of an existing PMU from the monitoring system, (ii) less informative voltage magnitude measurements owing to bad data injection, and (iii) changes in the PMU measurements due to cyber-attack events.

C. PREQUENTIAL EVALUATION

We follow the rules of *Prequential evaluation* [35] to be compliant with the stream learning framework. Essentially, prequential evaluation comprises two major stages: testing and training. In the test stage, the base learner predicts the class of the next available instance from the stream. After the test stage, the model metrics are updated. The base learner processes the actual instance during the training stage to update its structure and statistics. In our study, the training

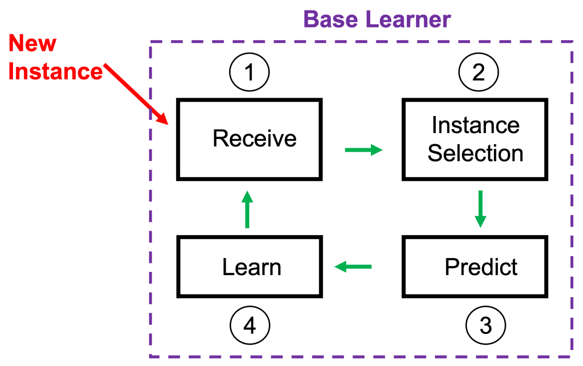

FIGURE 2. Streaming learning framework with instance selection.

stage is governed by our proposed streaming instance selection method, which is explained in detail in the next section. Pseudocode 1 shows the streaming learning process under the rules of prequential evaluation. Fig. 2 depicts the stream learning framework with instance selection.

III. METHODOLOGY

In this section, we first present the concept of spatiotemporal similarity. Next, we discuss the stream learning setup and details of our proposed streaming instance selection. Finally, we briefly describe the base learners.

A. LINEAR SPATIO-TEMPORAL SIMILARITY

Our proposed instance selection technique for data streams is based on the similarity among instances. The similarity captures the comparability of a pair of instances, and is measured using a distance function. The concept of similarity is inversely related to the concept of distance [36]. In other words, the smaller the distance between the instances, the more similar they are. Usually, the term *distance* is associated with distance in space. However, distance can not only be defined in space but also in time. Moreover, distance can be defined as a function of time and space, as suggested in [37] and shown in Fig. 3.

Remark 1: Potentially accurate results can be obtained using a spatiotemporal distance for contingency detection in power systems. For example, an instance with a low time distance but a high spatial distance may represent an abrupt change in the concept, such as transitioning from normal operation to a double line fault. An instance with a low spatial distance and high time distance may indicate the evolution of a concept such as a single line fault under two different loading conditions.

Let $\boldsymbol{x}_t \in \mathbb{R}^m$ be the target instance whose class is to be predicted, $\boldsymbol{x}_i \in \mathbb{R}^m$ an already observed instance, $T(\cdot)$ a distance function in time, and $S(\cdot)$ a distance function in space. More formally, the spatiotemporal distance between $\boldsymbol{x}_t$ and $\boldsymbol{x}_i$ is defined as the following linear relationship:

$$D(\boldsymbol{x}_t, \boldsymbol{x}_i) = T(\boldsymbol{x}_t, \boldsymbol{x}_i) + S(\boldsymbol{x}_t, \boldsymbol{x}_i) \qquad (1)$$

Pseudocode 1 Main Loop for Streaming Learning

```
Require: $\mathcal{M} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{\infty}$, base learner $\mathcal{L}$
1: for $t = 1, 2, 3, \ldots$ do
2:     ▷ Scale the feature vector $\boldsymbol{x}_t$ incrementally
3:     ▷ Test $\mathcal{L}$ with $(\boldsymbol{x}_t, y_t)$
4:     ▷ Update the model metrics
5:     ▷ Train $\mathcal{L}$ with SIS using Algorithm 2
6: end for
```

To simplify the notation, we refer to the distance from $\boldsymbol{x}_i$ to the target instance $\boldsymbol{x}_t$ as D_{t-i}. In Figure 3, we illustrate the concept of linearly combining spatial and time distances. There are multiple candidate functions for $S(\cdot)$, such as, the Euclidean distance, Manhattan distance, cosine similarity distance, and any other existing function that measures the spatial distance. In this study, we choose the Euclidean distance:

$$S(\boldsymbol{x}_t, \boldsymbol{x}_i) = \|\boldsymbol{x}_t - \boldsymbol{x}_i\|_2 \qquad (2)$$

Assuming uniformly spaced time intervals and considering the N most recent observed instances, we choose the distance in time, defined as a linear function of the time indices:

$$T(\boldsymbol{x}_t, \boldsymbol{x}_i) = \frac{|t-i|}{N} \qquad (3)$$

Other more complex time distances can be chosen; for example, the exponential function $T(\boldsymbol{x}_t, \boldsymbol{x}_i) = e^{|t-i|}$. This choice gives more importance to recent instances; however, we leave this and other complex spatial and time distance functions for future work.

A weight can be assigned to the Euclidean distance function using a fixed or cross-validation strategy [37]. This work assign a time-variant weight α_t to the Euclidean distance function to reduce the impact of an inappropriate range of feature values. To do so, we use a scaling function $s : \mathbb{R}^m \mapsto \mathbb{R}^m$ that transforms the feature domain of all instances in such a way that the values of the features are on a similar scale. Specifically, we scale the instances such that the values of the features have zero mean and unit variance. At each time step, the running mean and variance are maintained. Scaling is different from offline scaling because the exact means and variances are unknown beforehand [38]. Thus, the Euclidean distance becomes:

$$S(\boldsymbol{x}_t, \boldsymbol{x}_i) = \|s(\boldsymbol{x}_t) - s(\boldsymbol{x}_i)\|_2 = \alpha_t \|\boldsymbol{x}_t - \boldsymbol{x}_i\|_2 \qquad (4)$$

Remark 2: Power systems measurements make the spatiotemporal distance vulnerable to improper feature scaling. For example, data packets from PMUs come from different units. Therefore, measurements exhibit different orders of magnitude.

B. THE INSTANCE SELECTION ALGORITHM

This section describes the proposed stream instance selection (SIS) algorithm. Let $\boldsymbol{x}_t$ be the target instance, $\mathcal{L}$ be the base learner, and $\mathcal{R} = \{(\boldsymbol{x}_{t-i}, y_{t-i})\}_{i=1}^{N}$ the set containing the most

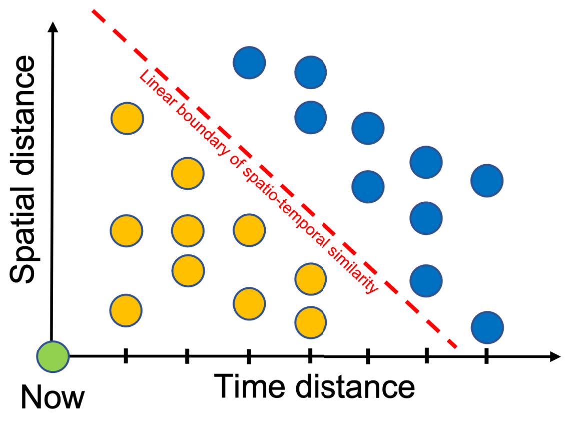

FIGURE 3. Linear combination of time and spatial distances. Each circle corresponds to a historical instance. The green circles are the target instance, while the yellow ones are similar to the targets because they have the least spatiotemporal distance determined by the red similarity boundary.

Pseudocode 2 SIS

```
Require: base learner $\mathcal{L}$, set of most recent instances $\mathcal{R} = \{(\boldsymbol{x}_{t-i}, y_{t-i})\}_{i=1}^{N}$
1: ▷ Reorder instances from set $\mathcal{R}$ using Algorithm 3
2: ▷ Reset the base learner $\mathcal{L}$
3: ▷ Search the optimal size for sliding window, and train the base learner $\mathcal{L}$ using Algorithm 4
4: Return $\mathcal{L}$
```

recent observed instances. Let $g : \mathcal{U} \mapsto \mathcal{V}$ denote the one-to-one natural-valued function, where $\mathcal{U} = \{1, 2, \ldots, N\}$, and $\mathcal{V} = \{t-1, t-2, \ldots, t-N\}$. Pseudocode 1 presents the steps of the main loop of stream learning, where the SIS algorithm is used to enhance the performance of the base learner. The algorithm has three algorithmic stages explained as follows:

(1) Reorder: The set of observed instances is sorted in ascending order according to their distance to the target instance $\boldsymbol{x}_t$. The sorting procedure assigns a new set of indices $\{g(1), g(2), \ldots, g(N)\}$ to the observed instances, so the base learner is trained first with the most similar instances. Pseudocode 3 presents the steps of this stage.

(2) Reset: The base learner is reset to forget what was learned in the previous time step. This reset allows the model to adapt to concept drifts. The effectiveness of bypassing concept drifts depends on N.

(3) Search: In this stage, SIS uses a sliding window $\mathcal{W}$ containing previous instances to train the base learner. The size of the window $\mathcal{W}$ is chosen dynamically, which is upper bounded by N. SIS evaluates the trained base learner with a trial set containing the k most recent instances. The trial set is indexed by time $\{t-1, t-2, \ldots, t-N\}$, and not by the sorted indices $\{g(1), g(2), \ldots, g(N)\}$. The training window $\mathcal{W}$ is found using a warm restart to alleviate the processing of all the most recent instances in $\mathcal{R}$. SIS performs a local search around the previous best window size to find the new best size. Let b be the previous best window size and r a natural number that defines the search size. SIS searches the next best window's size in the interval $[l, u] \subseteq [1, N]$, where $l = b - r$ and $u = b + r$. Finally, SIS stops the search when the learner error on the trial set is less than a threshold ϵ. Pseudocode 4 presents the steps involved in this stage.

Pseudocode 3 Reorder

```
Require: Target instance $\boldsymbol{x}_t$, set of most recent instances $\mathcal{R} = \{(\boldsymbol{x}_{t-i}, y_{t-i})\}_{i=1}^{N}$
1: for $i = 1, \ldots, N$ do
2:     ▷ Compute the distance $D_{t-i}$ according to (1)
3: end for
4: ▷ Sort the distances in ascending order $D_{g(1)} < D_{g(2)} < \cdots < D_{g(N)}$
5: ▷ Build the natural-valued function $g : \mathcal{U} \mapsto \mathcal{V}$, where $\mathcal{U} = \{1, 2, \ldots, N\}$, and $\mathcal{V} = \{t-1, t-2, \ldots, t-N\}$
6: Return $g$
```

TABLE 2. Time and memory cost and complexity of the SIS algorithm per stage.

Stage		Time	Memory
(1) Reorder	Cost	$2N + N\log(N)$	$2N + 1$
	Complexity	$\mathcal{O}(N\log(N))$	$\mathcal{O}(N)$
(2) Reset	Cost	1	1
	Complexity	$\mathcal{O}(1)$	$\mathcal{O}(1)$
(3) Search	Cost	$l + 2rk$	$2N$
	Complexity	$\mathcal{O}(l + 2rk)$	$\mathcal{O}(N)$

At each time step, the SIS assigns a new index to the instances in $\mathcal{R}$ by solving the following optimization problem:

$$\begin{aligned} \min_{g:\mathcal{U}\mapsto\mathcal{V}} \; & \sum_{i=2}^{N} |D_{g(i)} - D_{g(i-1)}| \\ \text{s.t. } & \mathcal{U} = \{1, 2, \ldots, N\} \\ & \mathcal{V} = \{t-1, t-2, \ldots, t-N\} \end{aligned} \tag{5}$$

Then, SIS finds the optimal window $\mathcal{W}$ by solving:

$$\begin{aligned} \min_{\mathcal{W}\subset\mathcal{R}} \; & |\mathcal{W}| \\ \text{s.t. } & l \leq |\mathcal{W}| \leq u \\ & \frac{1}{k}\sum_{i=1}^{k} L_{\mathcal{W}}\left(\mathcal{L}(\boldsymbol{x}_{t-i}), y_{t-i}\right) \leq \epsilon \\ & L_{\mathcal{W}} = \begin{cases} 1; & \mathcal{L}(\boldsymbol{x}_k) = y_k \\ 0; & \mathcal{L}(\boldsymbol{x}_k) \neq y_k \end{cases} \\ & \boldsymbol{x}_{g(j)} \in \mathcal{W}, \quad \forall j \in \{1, \ldots, |\mathcal{W}|\} \end{aligned} \tag{6}$$

Pseudocode 4 Search

```
Require: $|\mathcal{R}| = N$, base learner $\mathcal{L}$, natural-valued function $g$, number of testing instances $k$, previous best window size $b$, natural number $r$, error threshold $\epsilon$
1:  ▷ Set the upper and lower limits $u = b + r$, $l = b - r$
2:  for $i = 1 : u$ do
3:      if $i > u$ then
4:          ▷ break
5:      end if
6:      ▷ Train the base learner $\mathcal{L}$ with $(\boldsymbol{x}_{g(i)}, y_{g(i)})$
7:      if $i < l$ then
8:          ▷ continue
9:      end if
10:     for $j = 1 : k$ do
11:         ▷ Test base learner $\mathcal{L}$ with $(\boldsymbol{x}_j, y_j)$
12:         ▷ Update metric learnerError
13:     end for
14:     if learnerError $< \epsilon$ then
15:         ▷ $b = i$ and break
16:     end if
17: end for
18: Return $\mathcal{L}$, $b$
```

Algorithms 3 and 4 describe the procedures of SIS for solving problems (5) and (6), respectively. Fig. 4 shows the flowchart of the complete process.

The computational complexity per stage of the SIS algorithm is presented in Table 2. The complexity is expressed using the big-O notation and corresponds to the worst case. The cost expresses the time iterations and memory size based on the parameters of the SIS algorithm, such as N, l, r, and k. The complexity operator considers the higher-order terms of the cost only, and without scaling. The cost and complexity of each stage are added to obtain the total cost and complexity of the SIS algorithm. The algorithm uses memory $\mathcal{O}(N)$ and time $\mathcal{O}(N \log(N) + l + 2rk)$, both of which are concentrated in stages (1) and (3). The second stage, Reset, is a one-line statement that reinitializes the parameters of the base learner. Considering stage (1), the distance computation (lines 1-3 of Pseudocode 3) requires time and memory costs of N and $N + 1$, respectively. The sorting statement (line 4 of Pseudocode 3) sorts the set $\mathcal{R}$ and is executed using the Tim sort algorithm with a time cost of $N \log N$ and a memory cost of N. The natural-valued function g is a mapping with a memory size N, which is built using N iterations. The third stage trains the base learner using the first l instances (lines 6-9) of the reordered set $\mathcal{R}$. The base learner is then validated k times with the following $2r$ ordered instances of the set $\mathcal{R}$; thus, the time cost of the third stage is $l + 2rk$. The third stage requires having in memory the set $\mathcal{R}$ and the function g, each with size N. It is desirable to set $r \ll N/2$ and $k \ll N$ to reduce the complexity of the third stage.

C. THE HOEFFDING ADAPTIVE TREE

We propose the Hoeffding Adaptive Tree (HAT) as the base learner for the SIS algorithm. HAT is based on the Hoeffding Tree (HT), which establishes the Hoeffding bound to quantify the number of observations needed to compute some running statistics within a prescribed precision. Specifically, n independent observations of a random variable of range R are considered. The Hoeffding bound asserts that with a high probability $(1 - \delta)$, the estimated mean deviates from the true mean for no more than

$$\epsilon = R\sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{7}$$

The HAT comprises three main components: a window to remember recent examples, the adaptive windowing (ADWIN) method as a distribution-change detector, and ADWIN as an estimator for some input data statistics. Once a change is detected, an alternate tree is created and grows with the instances appearing immediately after the change. If the alternate tree is more accurate than the current tree, the alternate tree replaces the current tree.

ADWIN serves as an estimator and change detector that maintains a variable length window $\mathcal{W}$ of recent data such that the window has the maximal length statistically consistent with the null hypothesis that the average value inside the window has not changed. When two "big enough" subwindows of $\mathcal{W}$ have "distinct enough" averages, it can be said that there is a high probability that a change in the data distribution has occurred and the older items in $\mathcal{W}$ should be dropped. The "big and distinct enough" can be quantitatively defined by the Hoeffding bound [39]. In [28], the authors introduced the HAT+DDM classifier referred to as HAT+ADWIN+DDM in [28]. The DDM is a concept drift detector based on statistical process control to detect changes in data streams. DDM considers two levels of the base learner's error rate: i) the drift level when the error rate is very different from the past, and ii) the warning level when the error rate did not reach the drift level. Once the error rate reaches the drift level, a new context is established. The base learner is then trained with the instances between the warning and drift levels exclusively; see details in [11].

IV. EXPERIMENTAL SETUP

We use Massive Online Analysis (MOA) [40] and River [41] to conduct experiments. The MOA is an open-source Java framework for stream machine learning. River is an open-source python package dedicated to developing online/streaming machine learning algorithms. We run the experiments using a MacBook Pro 2019, 2.8 GHz Intel Core i7 processor, 16 GB 2133 MHz LPDDR3 RAM, and 1 TB hard disk drive. Based on initial tests, we set the SIS hyperparameter $\epsilon = 0.1$.

A. TESTBED ARCHITECTURE

The testbed architecture shown in Fig. 5 comprises three main elements: a physical power system, communication infrastructure, and monitoring and control components. The physical power system is simulated using the Real Time Digital Simulator (RTDS), which can emulate the behavior of

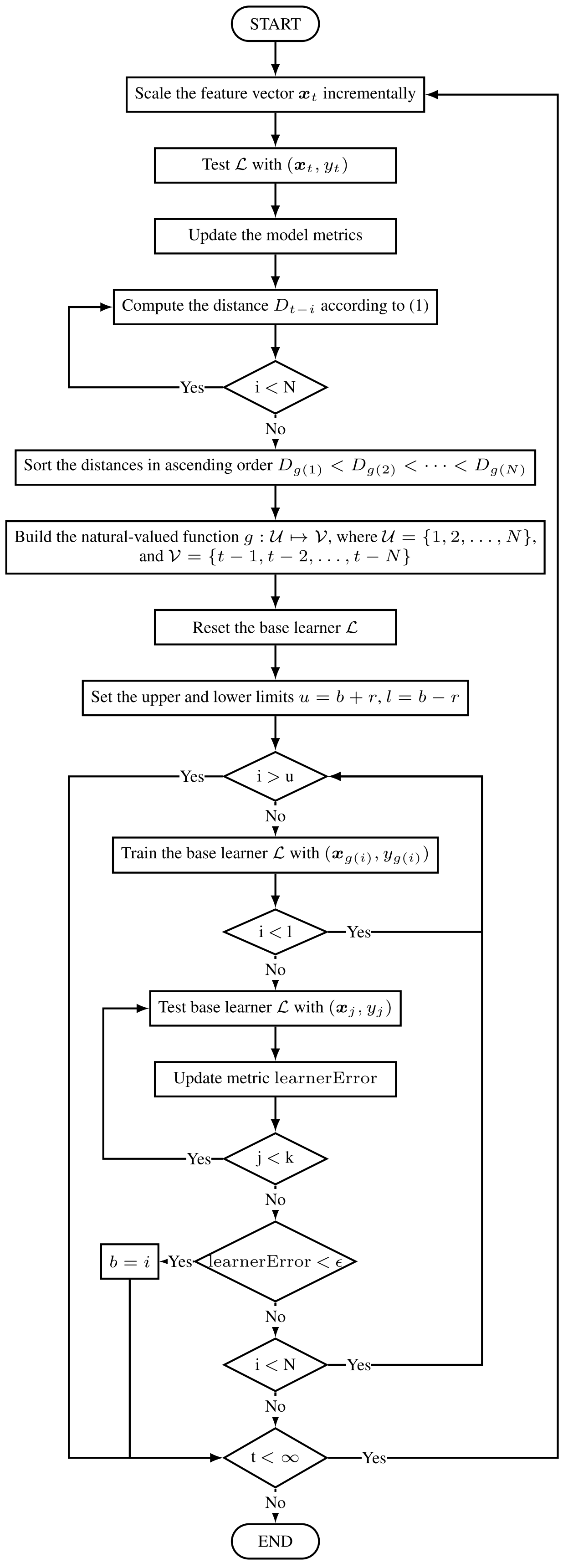
START
Scale the feature vector $\boldsymbol{x}_t$ incrementally
Test $\mathcal{L}$ with $(\boldsymbol{x}_t, y_t)$
Update the model metrics
Compute the distance D_{t-i} according to (1)
$i < N$
Yes
No
Sort the distances in ascending order $D_{g(1)} < D_{g(2)} < \cdots < D_{g(N)}$
Build the natural-valued function $g : \mathcal{U} \mapsto \mathcal{V}$, where $\mathcal{U} = \{1, 2, \ldots, N\}$, and $\mathcal{V} = \{t-1, t-2, \ldots, t-N\}$
Reset the base learner $\mathcal{L}$
Set the upper and lower limits $u = b + r, l = b - r$
$i > u$
Train the base learner $\mathcal{L}$ with $(\boldsymbol{x}_{g(i)}, y_{g(i)})$
$i < l$
Test base learner $\mathcal{L}$ with $(\boldsymbol{x}_j, y_j)$
Update metric learnerError
$j < k$
$b = i$
learnerError $< \epsilon$
$i < N$
$t < \infty$
END

FIGURE 4. Flowchart of the complete process.

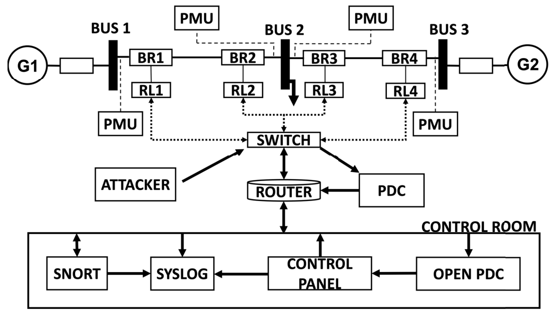

FIGURE 5. Cyber-physical power system testbed architecture. The circuit breakers and relays are shown as BR and RL, respectively.

power lines, buses, electrical machines, and load variations. The 3-bus power system with two generators, as shown in Fig. 5, is modified from the IEEE 9-bus test case with three generators. The reduced system is sufficiently small to comprehend its behavior in detail and apprehends the gist of the larger power system. The communication infrastructure comprises a physical network, industry-standard communication protocols, and control signals between control centers and substations. Networking monitoring devices, such as SNORT and Syslog, track malicious network activity and log events or messages to the control panel. The monitoring and control components include hardware such as PMUs, phasor data concentrators (PDCs), relays, data-processing engines, and industry-standard software. Each relay controls a breaker and sends information to the control panel through a communication network. Please refer to [42] for more details regarding the testbed architecture.

B. DATASET

We test the three learners with the multiclass industrial control system (ICS) cyber-attack dataset, which includes measurements related to 37 events in an electric transmission system [43]. The publicly available dataset consists of 15 sets with 5000 instances and 128 features each. The datasets have three versions: binary, ternary, and multiclass dataset. Each of the four PMUs measure 29 features (voltages, currents, frequency, and impedances) adding up to a total of 116 features. In addition, 12 additional features correspond to information from SNORT, Syslog, and the control panel, totaling 128 features. The 37 simulated events are listed and distributed as follows:

- **Normal operation (1 event).** The system operated under stable conditions with smooth load changes.
- **Line maintenance (2 events).** The power lines are open via the protection relays.
- **Short-circuit fault (6 events).** A power line shortage can occur at different locations across lines.
- **Attack of remote tripping command injection (6 events).** An attacker opens a breaker by sending a command to the relay.

- **Attack of relay setting change (16 events).** An attacker disables the secure relay configuration, which forces the relay to not trip against real faults and valid commands.
- **Attack of data Injection (6 events).** The attacker mimics a valid fault by changing the values of measurements such as voltages, currents, and impedances.

C. PERFORMANCE METRICS

The performance metrics used in this study are listed below:

1) **Accuracy (%):** The ratio of the number of correctly predicted instances to the total number of observed instances,

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of observed instances}}. \quad (8)$$

2) **Kappa (%):** This statistic takes into account the probability of predictions agreement by chance [44],

$$\text{Kappa} = \frac{\rho_o - \rho_{\text{ran}}}{1 - \rho_{\text{ran}}}, \quad (9)$$

where ρ_o is the accuracy of the base learner under study, and ρ_{ran} is the accuracy of a random base learner. If Kappa is positive, the base learner is better than a random prediction.

3) **Time (s):** Processing time taken by the base learner.
4) **Size (KB):** Size of the base learner in Kilobytes.
5) **Cost (RAM-hour):** Amount of RAM (KB) deployed for one hour.

D. LEARNERS

We use the Hoeffding Tree (HT) [45] and variants of the Hoeffding Adaptive Tree (HAT) [46] as the base learners for the experiments. After an initial trial on the MOA classifiers for streaming machine learning, we select the HAT+DDM and HT+DDM learners because they show better performance using a portion of the multiclass dataset. We set the base learners with the hyperparameters suggested in the related literature. More detailed tuning of the base learner's hyperparameters is left for future work. HAT+SIS and HT+SIS exhibit the same performance for the multiclass dataset. Finally, we choose HAT+SIS because it performs better for a price forecasting dataset, as shown in one of our experiments.

V. NUMERICAL RESULTS

In this section, we assess the performance of the three learners in six case studies using the ICS cyber attack datasets. Case studies are simulated by imitating real fault disturbances and cyber-attacks. We explore scenarios that affect the system's physics and monitoring architecture, including loading variation, PMU disappearance, and measurement overlapping. Additionally, we explore the performance of our proposed classifier using a price forecasting dataset.

A. CASE STUDY I: MULTIPLE EVENTS

This case study evaluates the performance of the three learners with 37 events from the multiclass dataset. The hyperparameter tuning of the SIS method using grid search is presented in Table 3. A performance comparison is presented in Fig. 6. It can be seen that HAT+SIS is the best performer among all learners, whereas HAT+DDM is the worst performer. As shown in Fig. 6(a), HAT+SIS achieves an accuracy of more than 99% in the first 250 instances, whereas the accuracy of HAT+DDM and HT+DDM is less than 99% during the same interval. Around instance 4000, HAT+DDM and HT+DDM obtain an abrupt decrease in accuracy, while HAT+SIS remains changed. The learners HAT+DDM and HT+DDM exhibit the same performance during the first 4,000 instances. Then, HT+DDM shows a slightly higher recovery rate accuracy than HAT+DDM. Fig. 6(b) presents the time comparison of the learners. We observe that HAT+SIS maintains a linear running time as the stream progresses. The model sizes of the three learners are shown in Fig. 6(c). It can be seen that the three learners demand moderate memory. HAT+DDM and HT+DDM use the same model sizes, while HAT+SIS exhibits reduced peak memory demands. Fig. 6(d) shows the combined effect of the time and model size.

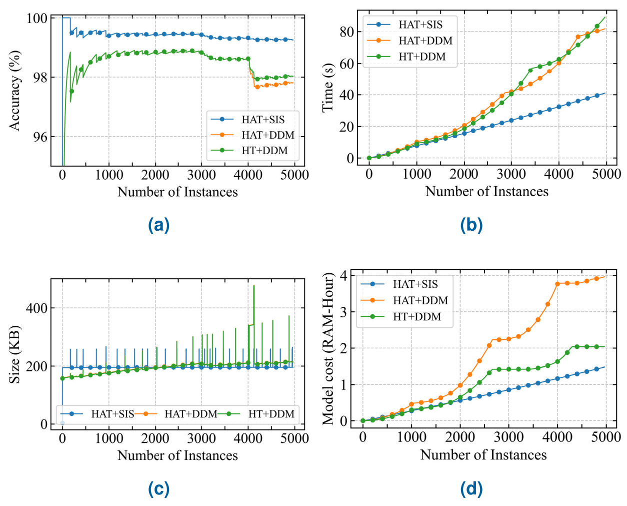

FIGURE 6. Performance comparison of the three learners in the case study I using the first set from the multi-class dataset. The evaluation considers the 37 events from the multi-class dataset in sequence. The comparison is shown for the following metrics: (a) Accuracy, (b) Time, (c) Size, and (d) Model cost.

To evaluate the performance of the three learners in static and evolving data, we use a window performance evaluator for classification with a window size of 20, as shown in Fig. 7. The three learners present the same accuracy during static data, represented by horizontal lines, reaching 100% accuracy. Evolving data manifests as an abrupt or gradual decrease in accuracy, which is displayed as downward peaks. HAT+SIS exhibits minor peaks during evolving data, especially around instance 4000, where HAT+DDM and HT+DDM exhibit significant accuracy degradation.

The performances of the three learners among the fifteen sets from the multiclass dataset are shown in Table 4. Considering the accuracy and Kappa statistic, HAT+SIS is the

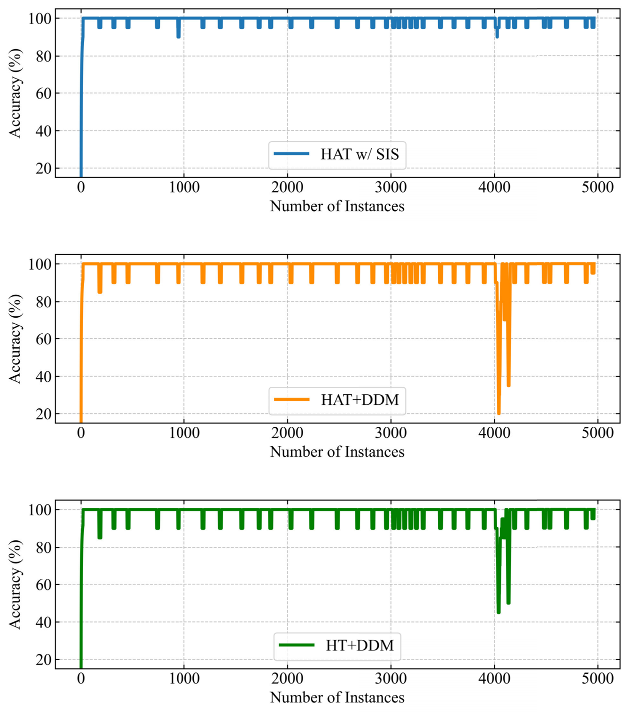

FIGURE 7. Performance evaluation of the base learners using the accuracy metric in dealing with stationary data and concept drift in case study I.

best performer among the fifteen sets, whereas HAT+DDM is the worst performer. Furthermore, HAT+SIS accounts for the smallest running time, model size, and cost. Table 5 presents the mean and standard deviation of the metrics across the fifteen sets. The results indicate that the performance of HAT+SIS remains invariant among the sets while exhibiting the most accurate and precise performance.

B. CASE STUDY II: TERNARY EVENTS

In this case study, we use the three-class dataset to evaluate the performance of the proposed approach. The three-class dataset consists of ternary events, that is, events labeled as natural, normal, and attack events. The attack events consist of twenty eight scenarios, the natural events consist of eight scenarios, and the normal events consist of one scenario. This section is important because the experiments assess the efficacy of the proposed classifier to distinguish normal operation from cyber and non-cyber contingencies in a cyber-physical power system. The hyperparameters of the SIS algorithm are the same as those in section V-A to alleviate the burden of hyperparameter tuning. Notice that the target space of the multiclass dataset is quite diverse and rich; hence, the best hyperparameters from section V-A benefit the HAT+SIS learner in this section.

Fig. 8 presents the performance of the three learners across the entire first set of the three-class dataset. Although HAT+SIS has a higher accuracy performance, the three learners show a similar overall accuracy. HAT+SIS and HT+DDM exhibit a linear time complexity across the entire

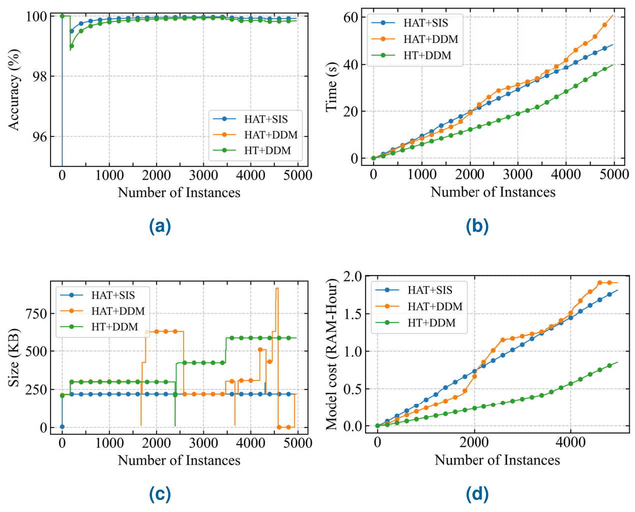

FIGURE 8. Performance comparison of the three learners in the case study II using the first set from the three-class dataset. The comparison is shown for the following metrics: (a) Accuracy, (b) Time, (c) Size, and (d) Model cost.

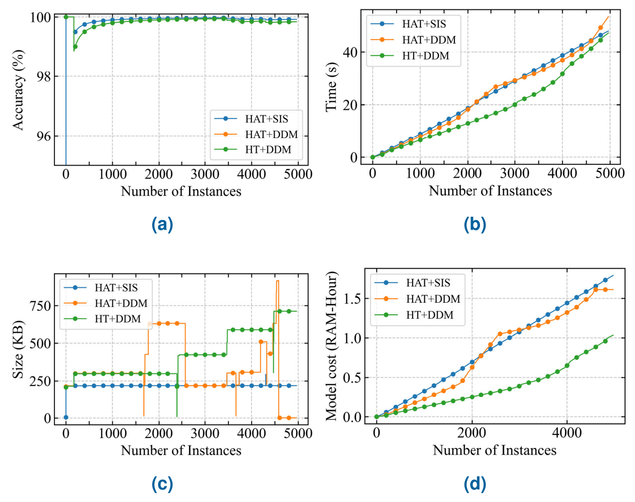

FIGURE 9. Performance comparison of the three learners in the case study III using the first set from the two-class dataset. The comparison is shown for the following metrics: (a) Accuracy, (b) Time, (c) Size, and (d) Model cost.

simulation, whereas the time complexity of HAT+DDM departs from being linear around instance 4000. Notably, HAT+SIS maintains a stable model size of less than 250 KB. HAT+SIS shows a linear model cost during the simulation, whereas HT+DDM has the lowest cost. Table 6 presents the metrics of the three learners for the 15 sets of the three-class dataset. We observe that the accuracy and Kappa metrics are not significantly different between HAT+SIS and the other two learners. Table 7 presents the four statistics computed from the performance metrics of the fifteen sets.

C. CASE STUDY III: BINARY EVENTS

We conduct the experiments in this case study using the two-class dataset corresponding to only two events, normal operation, and attack events. Binary classification is essential

TABLE 3. 15-Fold average hyperparameter tuning of the HAT+SIS learner using the multiclass dataset in case study I.

Base Learner	Metrics	$r=10$				$r=5$				$r=2$			
		$k=4$	$k=3$	$k=2$	$k=1$	$k=4$	$k=3$	$k=2$	$k=1$	$k=4$	$k=3$	$k=2$	$k=1$
HAT + SIS	Accuracy	99.07	99.13	99.25	99.30	99.07	99.12	99.23	99.29	99.04	97.91	98.60	99.27
	Kappa	99.04	99.10	99.23	99.27	99.03	99.09	99.20	99.27	99.00	97.83	98.55	99.24
	Time	54.62	50.56	46.93	43.10	54.12	50.46	46.49	42.92	53.54	50.55	47.28	43.28
	Size	195.83	195.82	195.82	195.82	195.82	195.82	195.82	195.82	195.82	195.82	195.82	195.82
	Cost	2.16	1.95	1.76	1.56	2.14	1.95	1.74	1.55	2.12	1.94	1.77	1.59

TABLE 4. Performance of the base learners for the 15 sets of the multiclass dataset in case study I.

Base Learner	Metric	Dataset														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HAT + SIS	Accuracy	99.25	99.29	99.32	99.31	99.30	99.27	99.29	99.30	99.33	99.37	99.31	99.31	99.30	99.30	99.30
	Kappa	99.23	99.26	99.29	99.28	99.28	99.25	99.27	99.28	99.30	99.35	99.29	99.29	99.27	99.27	99.27
	Time	40.95	42.27	44.57	43.51	42.10	41.38	43.19	43.89	44.03	45.54	43.31	42.63	43.36	42.05	43.73
	Size	195.73	195.73	195.92	195.92	195.73	195.97	195.78	195.73	195.92	196.02	195.88	195.73	195.78	195.73	195.78
	Cost	1.48	1.53	1.61	1.57	1.52	1.50	1.56	1.59	1.59	1.64	1.56	1.54	1.57	1.52	1.58
HAT + DDM	Accuracy	97.81	98.42	96.66	98.02	98.57	98.51	98.05	98.61	97.98	97.36	98.59	98.58	98.60	98.55	97.16
	Kappa	97.73	98.37	96.53	97.94	98.52	98.46	97.98	98.55	97.89	97.25	98.54	98.53	98.54	98.50	97.04
	Time	81.71	76.35	75.52	129.42	78.41	99.14	86.84	84.07	211.72	210.14	169.43	88.73	84.13	164.01	72.02
	Size	214.34	217.94	228.26	221.93	220.38	214.81	223.03	224.87	226.26	231.94	223.19	222.14	224.54	219.10	224.17
	Cost	3.96	3.72	4.21	2.97	4.55	3.17	3.95	4.12	2.41	2.80	1.99	3.53	4.31	2.14	3.18
HT + DDM	Accuracy	98.03	98.42	97.67	98.21	98.57	98.51	98.17	98.61	97.98	97.86	98.59	98.58	98.60	98.55	97.82
	Kappa	97.96	98.36	97.58	98.14	98.52	98.46	98.10	98.55	97.89	97.78	98.54	98.53	98.54	98.50	97.73
	Time	89.00	69.63	66.30	84.05	65.36	92.21	78.98	62.95	161.98	162.21	194.52	228.09	70.49	142.79	81.59
	Size	213.68	217.12	227.43	220.94	219.38	213.65	222.03	223.88	225.10	230.94	222.37	221.14	223.54	217.94	223.18
	Cost	2.04	2.64	2.08	2.06	2.48	1.39	2.06	2.19	1.31	1.48	0.70	2.30	2.25	1.45	2.18

TABLE 5. The mean μ, standard deviation σ, minimum and maximum values of the performance metrics tested on the 15 sets of the multiclass dataset in case study I.

Learner	Metrics	Statistics			
		μ	σ	min	max
HAT + SIS	Accuracy	99.30	0.027	99.25	99.37
	Kappa	99.28	0.025	99.23	99.35
	Time	43.10	1.22	40.95	45.54
	Size	195.82	0.10	195.73	196.02
	Cost	1.56	0.044	1.48	1.64
HAT + DDM	Accuracy	98.10	0.62	96.66	98.61
	Kappa	98.02	0.64	96.53	98.55
	Time	114.11	49.97	72.02	211.72
	Size	222.46	4.73	214.34	231.94
	Cost	3.88	1.01	1.99	4.55
HT + DDM	Accuracy	98.28	0.33	97.67	98.61
	Kappa	98.21	0.35	97.58	98.55
	Time	110.01	53.52	62.95	228.09
	Size	221.49	4.71	213.65	230.94
	Cost	1.81	0.494	0.70	2.64

because it allows us to test whether our proposed classifier can detect deviations from normal cyber-physical system behavior. We set the hyperparameters of the SIS algorithm as $r = 10$ and $k = 1$, as described in section V-B. Fig. 9 presents the performance of the three learners across the entire first set of the two-class dataset. HAT+DDM and HT+DDM show similar accuracy across the experiment, whereas HAT+SIS has slightly higher accuracy. HAT+SIS and HAT+DDM exhibit a linear time complexity during the entire simulation. HT+DDM has the smallest time complexity, but it shows a moderate increase at the end of the experiment. HAT+SIS maintains a stable model size of approximately 250 KB overall, whereas HT+DDM has step increases in its model size.

The three learners exhibit model cost behavior similar to their processing time behavior. Table 8 presents the performance of the three learners for the 15 sets of the two-class dataset. In this table, we observe that HAT+SIS is the best performer in terms of the accuracy, Kappa, and model size metrics. Table 9 shows the four statistics calculated using the results of the 15 sets of the two-class dataset. We notice that HAT+SIS has a minor standard deviation for all metrics, except the processing time.

D. CASE STUDY IV: LOADING VARIATIONS

We test the three learners in this case study to classify a fault event under different loading conditions. We simulate a data stream that has three stages in this order: (i) fault from $10-19\%$ on line 2; (ii) the fault is cleared, returning the system to normal operation; and (iii) fault from $10-19\%$ on line 2. Stages (i) and (iii) are considered to have distinct loads in the system. The confusion matrices are shown in Fig. 10. HAT+SIS accounts for the largest number of correct predictions, whereas HT+DDM accounts for the smallest. Although HAT+SIS correctly classifies the instances from the normal operation event, it shows the largest misclassification rate for the fault event. HAT+DDM and HT+DDM exhibit excellent performances for classifying the fault event. However, these performers do not exhibit attractive performance for normal event classification.

E. CASE STUDY V: PMU DISAPPEARANCE

Consider a monitoring system consisting of four PMUs whose measurements are modeled as features. After some time, one of the PMUs gets disconnected from the system. Such a situation can be modeled as a feature disappearance

TABLE 6. Performance of the three learners for the 15 sets of the three-class dataset in the case study II.

Learner	Metric	Dataset														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HAT + SIS	Accuracy	99.92	99.92	99.93	99.92	99.92	99.92	99.92	99.92	99.91	99.93	99.92	99.92	99.92	99.92	99.92
	Kappa	99.77	99.82	99.83	99.84	99.82	99.81	99.80	99.82	99.80	99.83	99.79	99.84	99.78	99.80	99.85
	Time	48.30	107.90	77.53	50.55	106.75	59.33	47.50	48.08	48.36	49.69	47.41	46.58	47.14	45.49	47.79
	Size	219.15	219.15	219.33	219.33	219.15	219.38	219.19	219.15	219.33	219.43	219.29	219.15	219.19	219.15	219.19
	Cost	1.81	1.64	1.86	1.89	1.83	1.65	1.77	1.79	1.80	1.86	1.77	1.74	1.76	1.70	1.78
HAT + DDM	Accuracy	99.83	99.84	99.85	99.84	99.84	99.83	99.84	99.84	99.85	99.85	99.84	99.84	99.84	99.84	99.84
	Kappa	99.54	99.65	99.66	99.68	99.64	99.63	99.61	99.65	99.69	99.67	99.60	99.68	99.57	99.60	99.70
	Time	60.66	50.56	47.04	38.88	131.88	44.34	107.68	95.40	105.95	102.29	60.19	63.46	50.96	40.81	57.01
	Size	219.11	219.11	219.30	643.12	302.82	303.37	219.19	219.45	303.20	637.85	302.98	219.19	219.27	219.19	224.85
	Cost	1.16	1.43	1.45	1.39	1.59	1.45	1.06	2.01	1.24	1.29	1.77	1.56	1.70	1.51	1.29
HT + DDM	Accuracy	99.82	99.83	99.85	99.84	99.83	99.83	99.84	99.84	99.83	99.83	99.84	99.84	99.84	99.84	99.82
	Kappa	99.52	99.65	99.64	99.68	99.64	99.62	99.61	99.63	99.69	99.65	99.60	99.68	99.59	99.61	99.69
	Time	39.80	39.65	47.98	49.67	40.65	49.98	60.93	54.88	53.51	58.89	33.26	42.12	40.00	40.12	47.59
	Size	590.28	921.81	716.03	585.08	509.08	1000.06	801.64	316.01	428.12	1000.14	224.06	636.65	717.90	513.86	980.01
	Cost	0.85	0.83	1.01	1.13	0.87	0.03	1.19	0.72	1.04	0.61	0.67	0.85	0.82	0.83	1.02

TABLE 7. The mean μ, standard deviation σ, minimum and maximum values of the performance metrics tested on the 15 sets of the three-class dataset in the case study II.

Learner	Metrics	Statistics			
		μ	σ	min	max
HAT + SIS	Accuracy	99.92	0.004	99.91	99.93
	Kappa	99.81	0.023	99.77	99.85
	Time	58.56	21.35	45.49	107.90
	Size	219.23	0.099	219.15	219.43
	Cost	1.77	0.072	1.64	1.89
HAT + DDM	Accuracy	99.84	0.005	99.83	99.85
	Kappa	99.63	0.046	99.54	99.70
	Time	70.47	29.72	38.88	131.88
	Size	298.13	143.85	219.11	643.12
	Cost	1.46	0.246	1.06	2.01
HT + DDM	Accuracy	99.83	0.005	99.82	99.85
	Kappa	99.61	0.046	99.52	99.69
	Time	46.60	8.062	33.26	60.93
	Size	662.71	245.82	224.06	1000.14
	Cost	0.831	0.275	0.03	1.19

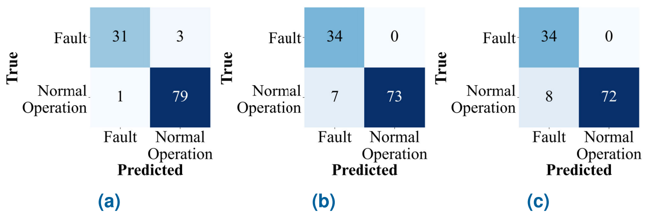

FIGURE 10. Confusion matrices of the three learners in case study IV. (a) HAT+SIS, (b) HAT+DDM, (c) HT+DDM.

drift, which may occur owing to a communication bottleneck, malfunctioning, or physical attack on the device. We simulate this scenario using a data stream consisting of 1450 instances with line maintenance, remote tripping commands, and fault events on both lines and different locations within the lines. Fig. 11 shows the results for the three learners in case study V with PMU's disappearance at instance 500. According to the results, HAT+SIS exhibits the best performance.

F. CASE STUDY VI: MEASUREMENTS OVERLAPPING

Some cyber-attacks may exhibit similar class conditional measurements distribution $P(X|Y)$ as fault disturbances. In other words, cyber-attacks and faults may fall into the same region of the measurements (features) space. This situation makes it difficult for learners to discriminate between similar events. In this scenario, we study one fault disturbance and two cyber-attacks: (i) fault from $80-90\%$ on line 1, and (ii) a data injection attack that mimics a fault from $80-90\%$ on line 1 with a remote tripping command; and (iii) a fault from $80-90\%$ on line 1 with relay #2 disabled. First, we force the learner to process a data stream from disturbance (i), followed by an abrupt concept change in the data distribution corresponding to cyber-attacks (ii) or (iii). Then, we make the learner learn oppositely, a cyber-attack (ii) or (iii), followed by the fault disturbancen in (i). Fig. 12 shows the accuracy of the three learners under the case study VI of measurements overlapping. HAT+SIS is less vulnerable to abrupt changes in the data distribution. In Figs. 12(a) and 12(c), the three learners exhibit similar accuracies when they first process instances from the fault distribution. However, their accuracy performance differs if they start processing instances from the cyber-attack distribution, as seen in Figs. 12(b) and 12(d). The results indicate that HAT+SIS performs best in this case study because it can correctly classify instances from data injection and remote tripping attacks.

G. PRICE FORECASTING DATASET

This section uses a dataset outside the cyber-attack and disturbance domain to further assess the merits of our proposed approach. We carry out the following experiments with the price forecasting Elec2 dataset based on the electricity price market in the Australian state of New South Wales [47]. The Elec2 dataset contains 45312 instances drawn between May 7th, 1996, and December 5th, 1998, with a sampling resolution of one instance for each half-hour. The market prices are set by matching the electricity demand with the cheapest combination of energy generation from all power stations. Electricity market data is subject to concept drifts because the market is affected by different factors such as weather, time of the day, season, and other factors, that is, the energy prices are not stationary. The dataset has eight features including electricity demand and price schedules. The class labels are set as UP or DOWN depending on

TABLE 8. Performance of the base learners for the 15 sets of the two-class dataset in the case study III.

Base Learner	Metric	Dataset														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HAT + SIS	Accuracy	99.92	99.92	99.93	99.92	99.92	99.92	99.92	99.92	99.91	99.93	99.92	99.92	99.92	99.92	99.92
	Kappa	99.76	99.81	99.82	99.83	99.81	99.80	99.79	99.81	99.78	99.82	99.79	99.82	99.77	99.79	99.83
	Time	47.93	61.16	73.72	58.23	59.80	63.59	66.99	66.31	87.75	68.56	49.86	47.41	47.75	46.43	47.70
	Size	219.15	219.15	219.33	219.33	219.15	219.38	219.19	219.15	219.33	219.43	219.29	219.15	219.19	219.15	219.19
	Cost	1.78	1.94	2.05	1.90	1.93	1.72	1.89	2.09	2.05	2.17	1.84	1.77	1.78	1.73	1.78
HAT + DDM	Accuracy	99.83	99.84	99.85	99.84	99.84	99.83	99.84	99.84	99.85	99.85	99.84	99.84	99.84	99.84	99.84
	Kappa	99.53	99.62	99.64	99.66	99.62	99.61	99.59	99.63	99.66	99.65	99.58	99.65	99.55	99.59	99.66
	Time	47.18	45.35	44.38	39.27	45.37	40.77	55.28	63.65	63.16	42.18	47.60	44.93	49.09	44.93	42.40
	Size	230.15	219.72	219.88	646.82	303.4	303.95	219.79	250.07	303.77	641.55	303.56	219.8	219.84	219.77	226.45
	Cost	1.16	1.46	1.16	1.18	1.39	1.30	0.95	1.78	1.17	1.22	1.57	1.35	1.41	1.45	1.26
HT + DDM	Accuracy	99.83	99.84	99.85	99.84	99.84	99.83	99.84	99.84	99.85	99.85	99.84	99.84	99.84	99.84	99.84
	Kappa	99.53	99.62	99.64	99.66	99.62	99.61	99.59	99.63	99.66	99.65	99.58	99.65	99.55	99.59	99.66
	Time	47.20	72.72	145.43	61.02	53.41	60.75	94.13	61.34	69.95	93.46	39.75	45.31	62.78	211.67	93.56
	Size	713.02	921.78	716.03	875.39	717.87	814.12	801.64	735.29	842.32	333.45	224.06	636.65	717.99	921.76	432.83
	Cost	1.03	0.88	1.01	1.22	0.96	1.29	1.24	0.79	1.16	0.81	0.78	0.92	0.82	0.81	0.88

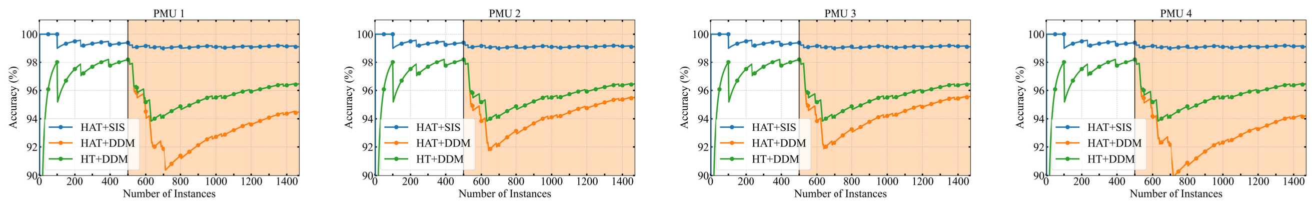

FIGURE 11. Accuracy of the three learners in case study V. The vertical black bar indicates the instance at which one of the PMUs disappears in the data stream. The light orange shaded area corresponds to the portion of the data stream where the PMU is inactive, whereas the uncolored area corresponds to the data stream where the PMU is active.

TABLE 9. The mean μ, standard deviation σ, minimum and maximum values of the performance metrics tested on the 15 sets of the two-class dataset in the case study III.

Learner	Metrics	Statistics			
		μ	σ	min	max
HAT + SIS	Accuracy	99.92	0.004	99.91	99.93
	Kappa	99.80	0.021	99.76	99.83
	Time	59.54	12.03	46.43	87.75
	Size	219.23	0.099	219.15	219.43
	Cost	1.89	0.141	1.72	2.17
HAT + DDM	Accuracy	99.84	0.005	99.83	99.85
	Kappa	99.61	0.040	99.53	99.66
	Time	47.70	7.40	39.27	63.65
	Size	301.90	143.50	219.72	646.82
	Cost	1.32	0.201	0.95	1.78
HT + DDM	Accuracy	99.84	0.005	99.83	99.85
	Kappa	99.61	0.040	99.53	99.66
	Time	80.83	44.96	39.75	211.67
	Size	693.61	208.85	224.06	921.78
	Cost	0.973	0.177	0.78	1.29

whether the current electricity price is higher or lower than the average price over the previous 48 instances (or the previous 24 hours).

The hyperparameters of SIS, r and k, are tuned using a grid search, as shown in Table 10. The best performance is obtained by setting $r = 10$ and $k = 2$. A finer grid search is left for future studies. Fig. 13 shows the performance of the three learners across the entire dataset. HAT+SIS is the best performer overall, whereas HT+DDM has the worst performance. HAT+DDM and HT+DDM exhibit a nonlinear increasing time complexity of approximately half of the dataset, whereas HAT+SIS shows a linear time complexity. HAT+SIS has the smallest model size during the entire simulation, whereas the model size of HT+DDM increases linearly. HT+DDM shows nonlinear behavior for the model cost, similar to its time complexity. HAT+SIS and HAT+DDM exhibit linearly increasing model cost. From Table 11, we observe that HAT+SIS outperforms HAT+DDM by 6% and HT+DDM by 12% in terms of accuracy. HAT+SIS requires at least half the processing time and model size of the other two learners. The model cost of HAT+SIS is similar to that of HAT+DDM but less than twice the cost of HT+DDM.

VI. DISCUSSION

The case studies from section V show that HAT+SIS adapts to online monitoring environments and is robust against abrupt concept drifts. Both aspects suggest that HAT+SIS is suitable for real-time power systems events and intrusion detection classification systems. In addition to the notable performance in terms of accuracy and Kappa, one meritorious upshot of HAT+SIS is that its running time and model cost are considerably lower than those of HAT+DDM and HT+DDM, as reported in Tables 4 and 5. On the one hand, HAT+DDM and HT+DDM are very sensitive to concept drifts. Under certain conditions, their performance can be worsen than that of a random classifier, as shown in Fig. 7. However, HAT+SIS remains almost unaltered by such a concept drift, which avoids misleading insights about the current state of the system. Similarly, HAT+SIS shows a stable rate of correct predictions for classifying events

TABLE 10. Hyperparameter tuning of the HAT+SIS learner using the price forecasting Elec2 dataset.

Base Learner	Metrics	$r = 10$				$r = 5$				$r = 2$			
		$k = 4$	$k = 3$	$k = 2$	$k = 1$	$k = 4$	$k = 3$	$k = 2$	$k = 1$	$k = 4$	$k = 3$	$k = 2$	$k = 1$
HAT + SIS‘	Accuracy	85.840	85.88	86.130	85.42	85.71	85.76	85.95	85.41	85.66	77.61	81.25	85.440
	Kappa	71.051	71.12	71.629	70.15	70.78	70.88	71.27	70.14	70.68	54.03	61.64	70.191
	Time	197.31	178.96	155.11	128.02	193.80	176.87	155.15	131.57	188.79	165.12	147.54	131.83
	Size	23.45	23.45	23.45	23.03	23.45	23.45	23.45	23.03	23.45	23.45	23.38	23.03
	Cost	1.01	0.89	0.74	0.57	0.98	0.87	0.73	0.58	0.95	0.80	0.68	0.588

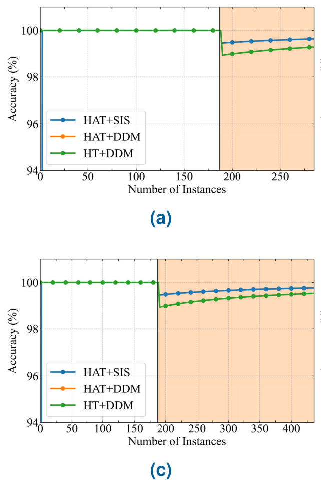

FIGURE 12. Accuracy of the three learners in case study VI. The vertical black bar indicates the instance at which the abrupt change in the data stream occurs. The white-shaded area corresponds to the portion of the stream under a fault disturbance distribution, whereas the light orange-shaded area corresponds to a cyber-attack distribution. (a) and (b) for fault disturbance and remote tripping command events; (c) and (d) for fault disturbance and relay #2 disabled. The performance of HAT+DDM and HT+DDM is the same in this scenario.

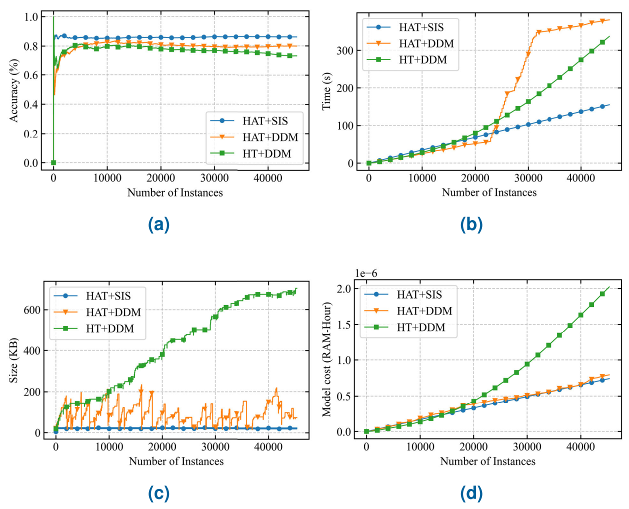

FIGURE 13. Performance comparison of the three learners using the price forecasting Elec2 dataset. The comparison is shown for the following metrics: (a) Accuracy, (b) Time, (c) Size, and (d) Model cost.

TABLE 11. Performance of the base learners using the price forecasting Elec2 dataset.

Base Learner	Metric	
HAT + SIS	Accuracy	86.13
	Kappa	71.62
	Time	155.11
	Size	23.45
	Cost	0.74
HAT + DDM	Accuracy	79.87
	Kappa	58.63
	Time	380.82
	Size	73.62
	Cost	0.71
HT + DDM	Accuracy	73.20
	Kappa	43.95
	Time	339.18
	Size	702.62
	Cost	2.05

with different loading conditions or the sudden disappearance of a PMU, as shown in Figs. 10 and 11, respectively. HAT+SIS rapidly adapts its tree-based structure to discriminate similar events. Fig. 12 shows the high accuracy recovery rate.

In case studies V-B and V-C, HAT+SIS exhibits a slightly higher accuracy and Kappa metric for binary and ternary events than HAT+DDM and HT+DDM. However, the three classifiers have similar accuracy and Kappa performance across the entire simulation. The major advantage of the HAT+SIS classifier in these case studies is that HAT+SIS maintains a flat model size and a linear running time and model cost. Based on these results, we observe that the major advantage of HAT+SIS over its competitors occurs when the target space is multiclass, as seen in case study V-A. Regarding the price forecasting dataset, HAT+SIS outperforms the other two classifiers in all the metrics. The HAT+SIS classifier maintains the lowest model cost, model size, and running time while exhibiting the highest accuracy and Kappa metrics across the 45312 instances.

The 37 events allow us to judge the efficacy of any classifier, and a classifier that exhibits a remarkable performance using such events can be judged as a noteworthy classifier. Although the numerical results show that our proposed classifier exhibits a higher accuracy than other existing classifiers, as shown in Table 12, we know that the set of events is not exhaustive. We acknowledge that other events can be considered to strengthen our proposed approach. For instance, we can include denial of service attacks such as data flooding, mutation of MODBUS protocol, or aurora attacks that refers to opening and closing a breaker near a generator in a rapid sequence.

A. HAT+SIS LIMITATIONS

Despite the merits of our proposed approach, as shown in Section V, the limitations of the HAT+SIS classifier need to be discussed. The proposed instance selection algorithm consists of three algorithmic stages. The first stage relies on reordering the instances based on their spatiotemporal distance to the target. Although instance reordering is an essential component of the first stage, it may degrade the original temporal distribution of the data stream, as shown in [48]. The temporal distribution of the data is crucial because it embeds the actual unknown concept of the data. Reordering the data may make it acquire a different concept over time; such a situation represents a different learning problem than the original one. In addition, SIS performs a window size search with a warm restart and stops the search only if the accuracy on a trial set is below a specified threshold. Such a strategy restricts the search around the previous best window size, leading to suboptimal window sizes. The search strategy imposes a trade-off between reducing the searching space and decreasing the complexity of the method to avoid time and memory restrictions. Finally, the selection algorithm introduces four additional hyperparameters that must be tuned according to the CPPS application. Consequently, the researcher either tunes the hyperparameters using an existing preprocessed dataset or tunes the hyperparameters on the fly.

B. PMU PLACEMENT STRATEGY

In this work, we use a testbed architecture of three buses with PMUs placed on all buses. If the test bed is a more extensive network, we may need to consider a specific PMU placement strategy for the proposed approach. Such a strategy will place PMUs in areas defined by clusters of buses that share the same dynamic behavior reducing the overall number of PMUs [49]. The dataset considered in this work contains events from dynamic transients such as three-phase short circuits, line outages, load variations, and breaker tripping. For instance, placing a PMU in a region where a subset of the buses of the system exhibits similar behavior under a short circuit and line outages is sound. A drawback of this strategy is that it may be challenging for the classifier to identify where an event occurs among the buses of the same cluster. Moreover, a tradeoff between the number of clusters and the cost of PMU deployment must be considered.

C. APPLICABILITY OF THE HAT+SIS LEARNER

The learner must exhibit a fast time response for the event and intrusion detection task. As mentioned in section IV-B, ICS datasets are built using high-speed networking and PMU data. PMUs have a very high data rate because they transmit tens or hundreds of synchrophasors per second through the networking architecture. Such a situation forces the learner to make predictions within a concise amount of time. In addition, the available working memory is not abundant, particularly for PMUs or phasor data concentrators (PDC). From Table 5, we can see that HAT+SIS processes approximately 5,000 instances from the ICS dataset in about 43 seconds. In other words, the learner processes about 112 instances per second. We observe that the learner's model size is almost 196 KB. Existing PMUs or PDCs on the market can handle such memory demands. Hence, HAT+SIS is a suitable classifier for PMU data-based contingency detection.

TABLE 12. Accuracies of HAT+SIS and other adaptive classifiers on the Attack dataset reported in the literature.

Algorithm	# of classes	Accuracy
Common Path Mining [50]	7	93.00
NNGE+STEM [17]	41	93.00
HAT+DDM [28]	41	92.00
Weights Voting Algorithm [51]	37	92.40
SSHAD [30]	37	96.84
Tree-based GBFS [32]	37	92.46
HAT+DDM	37	98.10
HT+DDM	37	98.28
HAT+SIS	37	**99.30**

D. COMPARISON WITH EXISTING WORKS

In Table 12, we compare the classification accuracy of the HAT+SIS learner and other algorithms based on the ICS dataset. Noticeably, we can observe that the HAT+SIS method outperforms existing algorithms for classifying disturbances and cyber-attacks. The reported results for the HAT+SIS, HAT+DDM and HT+DDM learners are based on the average accuracy among the fifteen sets from the multiclass dataset presented in Table 5.

VII. CONCLUSION

This paper proposes a novel combination of a stream learning classifier, the Hoeffding adaptive tree, with an instance selection algorithm for the real-time classification of cyber and non-cyber CPPS contingencies. HAT is a decision tree classifier that incorporates mechanisms to learn with continuous data streams and retrain its model as soon as a concept drift is detected. The selection algorithm improves HAT capabilities and comprises three algorithmic stages: reordering, resetting, and searching. The first stage uses a spatiotemporal distance function to measure the similarity of a set of observations with the target instance. The distance function uses an adaptive feature weight for the PMU measurements with different scales. The second stage reinitializes the parameters of HAT to allow the adaptation of the tree to the current concept underlying the data distribution. The third stage greedily searches the optimal size of a sliding window by identifying the instances most similar to the target instance. We simulate six case studies using the ICS datasets that consider various real scenarios with different cyber and non-cyber contingencies. The contingencies alter the physics of the system or its monitoring architecture. Extensive numerical results demonstrate that the novel combination of HAT with the SIS algorithm outperforms existing classifiers in the literature, especially for multiclass classification. In addition, the proposed classifier, HAT+SIS, also outperforms its competitors when evaluated using a price forecasting dataset. Such results indicate the robustness of HAT+SIS outside the event detection domain. This study demonstrates that the proposed classifier applies to real-time scenarios in CPPS, is sensitive to cyber and non-cyber contingencies, and shows superior performance over its competitors.

REFERENCES

[1] H. Akhavan-Hejazi and H. Mohsenian-Rad, "Power systems big data analytics: An assessment of paradigm shift barriers and prospects," *Energy Rep.*, vol. 4, pp. 91–100, Nov. 2018, doi: 10.1016/j.egyr.2017.11.002.

[2] R. V. Yohanandhan, R. M. Elavarasan, P. Manoharan, and L. Mihet-Popa, "Cyber-physical power system (CPPS): A review on modeling, simulation, and analysis with cyber security applications," *IEEE Access*, vol. 8, pp. 151019–151064, 2020.

[3] T. L. Hardy, *Software and System Safety*. Bloomington, MN, USA: AuthorHouse, Apr. 2012.

[4] D. Kushner, "The real story of Stuxnet," *IEEE Spectr.*, vol. 50, no. 3, pp. 48–53, Mar. 2013.

[5] D. U. Case, "Analysis of the cyber attack on the Ukrainian power grid," in *Proc. Electr. Inf. Sharing Anal. Center (E-ISAC)*, vol. 388, 2016, pp. 1–29.

[6] T. Starks. (2020). *The Cybersecurity 202*. Accessed: Aug. 11, 2021. [Online]. Available: https://www.washingtonpost.com/politics/the-202-newsletters/the-cybersecurity-202/

[7] R. Moxley and D. Dolezilek, "Case studies: Synchrophasor WAMPAC maximizes stability and economy," in *Proc. IET Conf. Rel. Transmiss. Distribution Netw. (RTDN)*, 2011, pp. 1–6.

[8] A. R. Metke and R. L. Ekl, "Security technology for smart grid networks," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 99–107, Jun. 2010.

[9] V. Terzija, G. Valverde, D. Cai, P. Regulski, V. Madani, J. Fitch, S. Skok, M. M. Begovic, and A. Phadke, "Wide-area monitoring, protection, and control of future electric power networks," *Proc. IEEE*, vol. 99, no. 1, pp. 80–93, Jan. 2011.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–37, Apr. 2014, doi: 10.1145/2523813.

[11] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence—SBIA*. Berlin, Germany: Springer, 2004, pp. 286–295.

[12] J. Giraldo, M. E. Hariri, and M. Parvania, "Decentralized moving target defense for microgrid protection against false-data injection attacks," *IEEE Trans. Smart Grid*, vol. 13, no. 5, pp. 3700–3710, Sep. 2022.

[13] M. M. S. Khan, J. A. Giraldo, and M. Parvania, "Attack detection in power distribution systems using a cyber-physical real-time reference model," *IEEE Trans. Smart Grid*, vol. 13, no. 2, pp. 1490–1499, Mar. 2022.

[14] I. Zografopoulos and C. Konstantinou, "Detection of malicious attacks in autonomous cyber-physical inverter-based microgrids," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 5815–5826, Sep. 2022.

[15] S. Lakshminarayana, S. Adhikari, and C. Maple, "Analysis of IoT-based load altering attacks against power grids using the theory of second-order dynamical systems," *IEEE Trans. Smart Grid*, vol. 12, no. 5, pp. 4415–4425, Sep. 2021.

[16] K.-D. Lu and Z.-G. Wu, "Constrained-differential-evolution-based stealthy sparse cyber-attack and countermeasure in an AC smart grid," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5275–5285, Aug. 2022.

[17] U. Adhikari, T. H. Morris, and S. Pan, "Applying non-nested generalized exemplars classification for cyber-power event and intrusion detection," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 3928–3941, Sep. 2018, doi: 10.1109/TSG.2016.2642787.

[18] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *Proc. 7th Int. Symp. Resilient Control Syst. (ISRCS)*, Aug. 2014, pp. 1–8.

[19] Y. An and D. Liu, "Multivariate Gaussian-based false data detection against cyber-attacks," *IEEE Access*, vol. 7, pp. 119804–119812, 2019.

[20] A. Gholami, A. Vosughi, and A. K. Srivastava, "Denoising and detection of bad data in distribution phasor measurements using filtering, clustering, and Koopman mode analysis," *IEEE Trans. Ind. Appl.*, vol. 58, no. 2, pp. 1602–1610, Mar. 2022.

[21] A. Parizad and C. J. Hatziadoniu, "Cyber-attack detection using principal component analysis and noisy clustering algorithms: A collaborative machine learning-based framework," *IEEE Trans. Smart Grid*, vol. 13, no. 6, pp. 4848–4861, Nov. 2022.

[22] M. Ganjkhani, M. Gilanifar, J. Giraldo, and M. Parvania, "Integrated cyber and physical anomaly location and classification in power distribution systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 7040–7049, Oct. 2021.

[23] R. Yadav and A. K. Pradhan, "PCA-LSTM learning networks with Markov chain models for online classification of cyber-induced outages in power system," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3948–3957, Sep. 2021.

[24] M. R. Habibi, H. R. Baghaee, T. Dragicevic, and F. Blaabjerg, "Detection of false data injection cyber-attacks in DC microgrids based on recurrent neural networks," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 9, no. 5, pp. 5294–5310, Oct. 2021.

[25] F. Li, Q. Li, J. Zhang, J. Kou, J. Ye, W. Song, and H. A. Mantooth, "Detection and diagnosis of data integrity attacks in solar farms based on multilayer long short-term memory network," *IEEE Trans. Power Electron.*, vol. 36, no. 3, pp. 2495–2498, Mar. 2021.

[26] M. Abdel-Basset, N. Moustafa, and H. Hawash, "Privacy-preserved generative network for trustworthy anomaly detection in smart grids: A federated semisupervised approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 995–1005, Jan. 2023.

[27] N. Dahal, O. Abuomar, R. King, and V. Madani, "Event stream processing for improved situational awareness in the smart grid," *Exp. Syst. Appl.*, vol. 42, no. 20, pp. 6853–6863, Nov. 2015, doi: 10.1016/j.eswa.2015.05.003.

[28] U. Adhikari, T. H. Morris, and S. Pan, "Applying Hoeffding adaptive trees for real-time cyber-power event and intrusion classification," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4049–4060, Sep. 2018, doi: 10.1109/tsg.2017.2647778.

[29] Z. E. Mrabet, D. F. Selvaraj, and P. Ranganathan, "Adaptive Hoeffding tree with transfer learning for streaming synchrophasor data sets," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5697–5704.

[30] G. Intriago and Y. Zhang, "Online dictionary learning based fault and cyber attack detection for power systems," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2021.

[31] E. Hallaji, R. Razavi-Far, M. Wang, M. Saif, and B. Fardanesh, "A stream learning approach for real-time identification of false data injection attacks in cyber-physical power systems," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3934–3945, 2022.

[32] D. Upadhyay, J. Manero, M. Zaman, and S. Sampalli, "Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 1104–1116, Mar. 2021, doi: 10.1109/TNSM.2020.3032618.

[33] A. Bifet and R. Kirkby. (May 2011). *Data Stream Mining a Practical Approach*. Accessed: Sep. 23, 2020. [Online]. Available: https://www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf

[34] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama, "Machine learning for streaming data: State of the art, challenges, and opportunities," *ACM SIGKDD Explor. Newslett.*, vol. 21, no. 2, pp. 6–22, Nov. 2019.

[35] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, Oct. 2012, doi: 10.1007/s10994-012-5320-9.

[36] M. Kordos, *Instance Selection in Machine Learning: New Directions in Similarity-based, Evolutionary and Embedded Methods*, I. Adamiec-Wójcik, Ed. Bielsko, Poland: Univ. Bielsko-Biala, 2019.

[37] I. Žliobaitė, "Combining similarity in time and space for training set formation under concept drift," *Intell. Data Anal.*, vol. 15, no. 4, pp. 589–611, Jun. 2011.

[38] *Ten Little Algorithms, Part 3: Welford's Method (and Friends)*. Accessed: Oct. 17, 2021. [Online]. Available: https://www.embeddedrelated.com/showarticle/785.php

[39] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 1–11.

[40] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," in *Proc. 1st Workshop Appl. Pattern Anal.*, vol. 11, Sep. 2010, pp. 44–50.

[41] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: Machine learning for streaming data in Python," 2020, *arXiv:2012.04740*.

[42] U. Adhikari, T. H. Morris, and S. Pan, "A cyber-physical power system test bed for intrusion detection systems," in *Proc. IEEE PES Gen. Meeting Conf. Expo.*, Jul. 2014, pp. 1–5.

[43] U. Adhikari, S. Pan, T. H. Morris, and J. Beave. *Industrial Control System (ICS) Cyber Attack Datasets, Dataset 1: Power System Datasets*. Accessed: Aug. 11, 2020. [Online]. Available: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

[44] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Mach. Learn.*, vol. 98, no. 3, pp. 455–482, Apr. 2014, doi: 10.1007/s10994-014-5441-4.

[45] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2000, pp. 1–11, doi: 10.1145/347090.347107.

[46] A. Bifet and R. Gavaldà, *Adaptive Learning From Evolving Data Streams*. Berlin, Germany: Springer, 2009, pp. 249–260, doi: 10.1007/978-3-642-03915-7_22.

[47] M. Harries and N. S. Wales, "SPLICE-2 comparative evaluation: Electricity pricing," Univ. New South Wales, Tech. Rep., 1999.

[48] I. Žliobaitė, "Controlled permutations for testing adaptive classifiers," in *Discovery Science*, T. Elomaa, J. Hollmén, and H. Mannila, Eds. Berlin, Germany: Springer, 2011, pp. 365–379.

[49] J. C. Cepeda, J. L. Rueda, I. Erlich, and D. G. Colomé, "Probabilistic approach-based PMU placement for real-time power system vulnerability assessment," in *Proc. 3rd IEEE PES Innov. Smart Grid Technol. Eur. (ISGT Europe)*, Oct. 2012, pp. 1–8.

[50] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 3104–3113, Nov. 2015, doi: 10.1109/TSG.2015.2409775.

[51] D. Wang, X. Wang, Y. Zhang, and L. Jin, "Detection of power grid disturbances and cyber-attacks based on machine learning," *J. Inf. Secur. Appl.*, vol. 46, pp. 42–52, Jun. 2019.

GABRIEL INTRIAGO (Student Member, IEEE) received the B.E. degree in electrical and computer engineering from Escuela Superior Politécnica del Litoral (ESPOL), Ecuador, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Santa Cruz. He is also a Research Assistant with the Mathematics Department and the MODEMAT Research Center, Escuela Politécnica Nacional, Ecuador. His research interests include machine learning, data stream mining, control, optimization, and security with applications to power systems.

YU ZHANG (Member, IEEE) received the B.Eng. degree (Hons.) in electrical engineering from the Wuhan University of Technology, Wuhan, China, in 2006, the M.Sc. degree (Hons.) in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010, and the Ph.D. degree (Hons.) in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2015. During the summer of 2014, he was a Research Intern with ABB US Corporate Research Center, Raleigh, NC. He is currently an Assistant Professor with the ECE Department, University of California at Santa Cruz (UCSC). Before joining UCSC, he was a Postdoctoral Researcher with UC Berkeley and Lawrence Berkeley National Laboratory. His research interests include cyber-physical systems, smart power grids, optimization theory, machine learning, and big data analytics. He received the Huawei Scholarship and the Infineon Scholarship from Shanghai Jiao Tong University, in 2009, the ECE Department Fellowship from the University of Minnesota, in 2010, and the Student Travel Awards from the SIAM and the IEEE Signal Processing Society, in 2014.