Tenet: Benchmarking Data Stream Classifiers in Presence of Temporal Dependence

Giacomo Ziffer
DEIB
Politecnico di Milano
Milano, Italy
giacomo.ziffer@polimi.it

Federico Giannini
DEIB
Politecnico di Milano
Milano, Italy
federico.giannini@polimi.it

Emanuele Della Valle
DEIB
Politecnico di Milano
Milano, Italy
emanuele.dellavalle@polimi.it

***Abstract*—In batch learning, it is commonly assumed that samples are independent and identically distributed (i.i.d.). However, this assumption does not hold in dynamic environments where data streams are not identically distributed due to concept drifts. Furthermore, while most Streaming Machine Learning (SML) literature assumes independence among examples, data streams often have important temporal components that learning should adequately consider. Neglecting this temporal dependence can lead to significant misguidance in designing and evaluating SML models. To support our thesis, we propose TENET, a novel benchmarking framework designed to evaluate data stream classifiers in non-i.i.d. scenarios comparatively. TENET consists of a data stream generator and a baseline. The data stream generator introduces temporal dependence into the data streams commonly used for evaluating SML algorithms. The baseline is a continuous version of the Long Short-Term Memory algorithm called cLSTM. Extensive experiments using TENET demonstrate that cLSTM consistently outperforms state-of-the-art SML classifiers when learning from data streams with temporal dependence. This result is a call to action for the SML and the Deep Learning communities to investigate classifiers in the time-dependent streaming scenario and makes TENET the first publicly available benchmark to support this research.**

***Index Terms*—Benchmark, Data Stream, Temporal Dependence, Concept Drift**

I. INTRODUCTION

Streaming Machine Learning (SML) [1] presents several challenges compared to the traditional Machine Learning (ML) setting. Data streams are formally defined as an unbounded sequence of data points, each with a timestamp, so a temporal order. This temporal property is essential as it distinguishes a data stream from non-streaming data. Nevertheless, a fundamental principle is assumed by most data stream classifiers: that data points are independent and identically distributed (i.i.d. data). Although this assumption may hold for typical ML data, it is admittedly dubious that it does for data streams. In many situations, the underlying distribution evolves, originating the so-called concept drift [2], and the data point received at time t is strongly correlated to those at previous time steps.

A significant body of research within SML focused on detecting concept drifts and developing techniques to adapt to such changes [1], [3], [4]. Despite this focus, the lack of independence received much less attention, with only a few works investigating this problem [5]–[10]. The foundational equivalence between ML and SML methods is contingent upon the assumption of i.i.d. data within individual concepts. Most SML approaches, therefore, assume the absence of temporal dependence in the data. However, this assumption is frequently unrealistic, as numerous data streams exhibit dependencies on their historical values [5], [6]. A given attribute value may result from an auto-regressive transformation applied to preceding instances, as in the fluctuation in commodity prices like electricity [7].

The existence of temporal dependence underscores the blurred boundary between data streams and time series. In Time Series Analysis [11], temporal dependence can manifest in various forms within a data stream, such as trends and seasonalities. Modeling these temporal patterns as they evolve can be challenging, and traditional methods assuming independence between observations may not be suitable. Consequently, addressing this complex problem necessitates the development of specialized methods capable of capturing the dependencies present in the data. One effective model is the Long Short-Term Memory (LSTM) Neural Network [12], a Recurrent Neural Network (RNN) designed to learn long sequences in the data. LSTM has demonstrated superior performance for forecasting time series data compared to traditional algorithms like the ARIMA model [13]. Although RNNs can be naturally applied for prediction in streaming stationary scenarios, their application in the presence of concept drifts still needs to be explored.

The objective of this work is to propose a systematic way to explore the impact of temporal dependence on data stream classifiers. We focus on the strong similarities between data streams and time series and how temporal dependence can influence or limit learning from data streams. Specifically, this study aims to address the following **research question**: *how can a comprehensive benchmarking framework be established, comprising baselines, evaluation metrics, datasets, and a rigorous comparative analysis methodology, to facilitate and standardize the exploration of SML approaches for stream classification tasks exhibiting temporal dependence?*

The main contributions of this work are summarized as follows.

- We introduce TENET, a novel benchmarking framework for the comparative evaluation of data stream classifiers. It features a meta-generator that creates time-dependent data streams, offering new synthetic benchmarks to investigate existing SML algorithms' guarantees. TENET is the first publicly available benchmark to support developing new SML classifiers for time-dependent streaming scenarios.

- We define a framing of the benchmarking problem for data streams exhibiting temporal dependence through TENET, leveraging decision theory as formalized in [6]. This theory underpins TENET's benchmarking approach, tailored for scenarios involving temporal dynamics.
- We propose as a baseline a continuous version of the Long Short-Term Memory, cLSTM, as a mini-batch classifier capable of handling concept drift and temporal dependence. cLSTM provides insights into the capabilities of sequential models in improving classification performance when learning from not i.i.d. data streams.
- We provide empirical evidence of the effectiveness of cLSTM and the TENET framework in evaluating data stream classifiers with not i.i.d. data streams. Our comprehensive experiments demonstrate the insufficiency of augmenting the feature space with past labels, i.e., Temporal Augmentation [5], and confirm that cLSTM consistently outperforms the state-of-the-art SML classifiers.

All data streams and algorithms used in this paper are available online[1] as a publicly available benchmark to help other researchers in their work to reproduce the results shown in this study or the development of new algorithms.

The remainder of this paper is organized as follows. Section II presents relevant works about handling temporal dependence. In Section III, we propose TENET, while Section IV explains cLSTM. Section V presents the experimental settings and the results of the evaluation campaign. Finally, Section VI discusses our conclusions based on the experiments and outlines directions for future research.

II. BACKGROUND

Temporal dependence in data streams has been approached through estimation techniques with a forgetting factor, where the significance of a data point in a stream is inversely proportional to its age [14], [15]. However, these methods do not fully address how data points are related over time. Despite the powerful and easy-to-deploy nature of many Streaming Machine Learning (SML) algorithms [1], they cannot capture temporal dependence. Only a few investigations have been conducted on this topic within the SML literature, with only one change detector being proposed in this scenario [16]. In [5]–[7], the authors propose an evaluation method and decision theory to consider temporal dependence. [8] discusses the similarities between time series and data streams, while [9] investigates temporal dependence problem for the regression task. The effects of time on data stream classifiers, thus, still need to be exhaustively explored.

The Temporally Augmented classifier [5] is the most widely used SML approach for addressing temporal dependence. Denoting by k the order of the temporal augmentation, this procedure adds to the feature space of each data point the labels of the previous k data points. In a data stream classification problem, each data point d_i is a tuple denoted by $< X_i, y_i >$. X_i represents the vector containing the features of the data point, while y_i corresponds to its associated label. The temporal augmentation augments the feature space of X_i with previous class labels $y_{t-1}, \dots, y_{t-k}$ as expressed in Equation 1 to obtain a new feature vector X_t^{SWT}.

$$X_t^{SWT} = X_i \cup \{y_{t-1}, y_{t-2}, ..., y_{t-k}\} \tag{1}$$

The prediction for observation X_t becomes a function h_t of the original input attributes and recent class labels $\hat{y_t} = h_t(X_t, y_{t-1}, \dots, y_{t-k})$. This approach can account for dependence among input features and past labels and dependence among past labels at different times. It is worth noting that the Temporally Augmented classifier is not a novel approach specific to data streams. It is common in time series forecasting, where past values of the target variable are the primary predictive information. By reconstructing the input space to represent all dependencies in a new set of dimensions, the Temporally Augmented classifier exploits the Takens' embedding theorem [17], which guarantees finding the best embedding that transforms a univariate time series into a high-dimensional space, where temporal relationships unfold through necessary time delays representing all dependencies.

Recurrent Neural Network (RNN) models are sequential architectures capable of capturing temporal correlations. The training involves feeding back the layer's hidden state from time step t as input to the next step $t + 1$, along with the input for that step, and updating the weights using the back-propagation through time optimisation algorithm [18]. RNNs struggle when learning sequence problems with long-term dependencies. Long Short-Term Memory neural networks (LSTMs) [12] were proposed to address this issue. LSTMs not only track dependencies at time step t with the hidden state h_{t-1} from the previous step $t - 1$ but also possess a memory cell c_{t-1} that does not vanish during the learning process. LSTMs are widely used to learn sequence problems by tracking dependencies when making predictions over time-correlated data. In forecasting time series data, they usually perform better than traditional-based algorithms such as ARIMA [13]. However, despite their suitability for this scenario, their application to data streams has received limited attention until recently. Notably, emerging studies have begun to explore the potential of Artificial Neural Networks in streaming contexts [19].

III. TENET BENCHMARK

Let a data stream $S = \{d_{-\infty}, \dots, d_{-1}, d_0\}$ be an open-ended stream of data points d collected over time, where each data point d_i is a tuple denoted by $< X_i, y_i >$, containing input examples in which the feature vector $X \in \mathbb{R}^n$ and $n \geq 1$ and the corresponding label is y. The set of possible values for y is finite, denoted as $y \in l_1, \dots, l_L$ with $L \geq 2$, indicating a classification task. In this case, we assume a

[1]https://github.com/gziffer/tenet

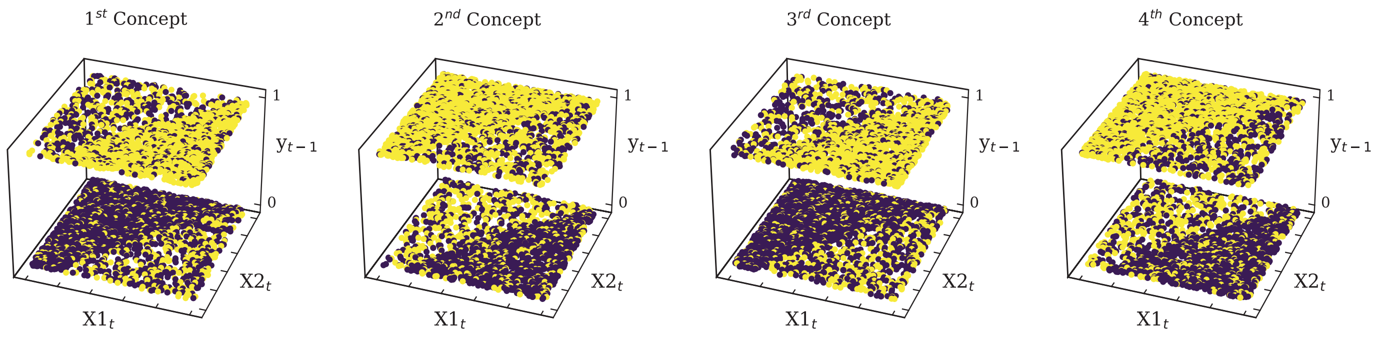


Fig. 1: Visualization of the 4 concepts of the Sine1 dataset (yellow for class 0 and purple for class 1) with a temporal injection of order 2. Z-axis represents the past class label y_{t-1}. Each concept drift inverts the two classes from the previous concept, i.e., the purple points in the highest part of the plane $y_{t-1} = 0$ of 1^{st} concept move to the lowest part in the 2^{nd} concept and back to the highest part in the 3^{rd} concept.

problem setting where new input examples X are presented every time units to the learning model for prediction, such that X_t represents a vector of features available at time t. The true class label y_t, corresponding to instance X_t, is available before the next instance X_{t+1} appears. Thus, it can be used for training immediately after it has been used for prediction.

An important characteristic of data stream classification is the presence of temporal dependence within the data stream. In this work, we focus on studying this temporal dependence and its impact in influencing the performance and accuracy of classification models. Temporal dependence reflects the fact that data points are not independent but are influenced by preceding data. Moreover, we also consider the occurrence of concept drifts, which are changes in the data distribution that can significantly alter decision boundaries over time. Evolving data distributions are common in many real-world applications, leading to concept drifts that pose challenges for maintaining accurate and robust models. By addressing both temporal dependence and concept drift, TENET aims to develop a comprehensive understanding for data stream classification in dynamic environments.

A. Theoretical Grounding

Temporal dependence in data streams means that observations are not independent from each other with respect to time of arrival. To establish the theoretical foundations of our approach, we refer to the following formal definitions [6].

Definition III.1. *First order temporal dependence is present when an observation is not independent from the previous observation, i.e.* $P(y_t, y_{t-1}) \neq P(y_t)P(y_{t-1})$, *where* t *is the time index,* y_t, $y_{t-1} \in \{1, \dots, k\}$, *where* k *is the number of classes. An* l^{th} *order temporal dependence is present if* $P(y_t|y_{t-1}, \dots, y_{t-l}) \neq P(y_t|y_{t-1}, \dots, y_{t-1-l})$.

The temporal dependence for class i is positive if the joint probability of the current and previous labels, $P(y_t, y_{t-1})$, exceeds the product of their individual probabilities, $P(y_t)P(y_{t-1})$. In this scenario, labels tend to exhibit a higher propensity to follow the same sequence compared to their prior probabilities. Conversely, a negative temporal dependence, where $P(y_t, y_{t-1}) < P(y_t)P(y_{t-1})$, incentivizes label alternation. This study focuses on combining these two types of temporal dependence.

Let's consider the scenario where we need to make a prediction $\hat{y}_t$ at time t. By this point, we will have already observed the feature vectors $(X_1, \dots, X_{t-1})$ and, assuming the standard SML setting of immediate label availability after making predictions, we will also have access to the corresponding true labels $(y_1, \dots, y_{t-1})$. Upon observing the current feature vector X_t, the optimal strategy is to leverage all the available evidence and predict:

$$\hat{y}_t = \operatorname*{argmax}_i P(y_t = i|X_t, y_{t-1}, \dots, y_1). \tag{2}$$

If there is no temporal dependence in the data, then Eq. 2 reduces to $\hat{y}_t = \operatorname{argmax}_i P(y_t = i|X_t)$, since because of the Bayesian Rule $P(y_t = i|X_t, y_{t-1}, \dots, y_1) = \frac{P(y_t=i|X_t)P(y_{t-1})\cdots P(y_1)}{P(X_t)P(y_{t-1})\cdots P(y_1)} = P(y_t = i|X_t)$. In practice, the order of temporal dependence considered is often manually restricted to the l^{th} order. In this case, the prediction becomes $\hat{y}_t = \operatorname{argmax}_i P(y_t = i|X_t, y_{t-1}, \dots, y_{t-l})$, where l represents the length of the historical sequence taken into account. This study primarily focuses on second-order temporal dependence.

B. Temporal Injection

Temporal dependence is closely related to concept drift, as the data stream can resemble a time series when considered chronologically. Comparing data stream classifiers poses a challenge due to variations in recommendations, evaluation procedures, datasets, and assumptions among different authors [20]. Additionally, the limited availability of real-world datasets restricts comparative studies on new proposals [21]. This scarcity of benchmark data necessitates using approaches that simulate changes in static data or generate synthetic data, such as feature switching, class swapping, class merging, and

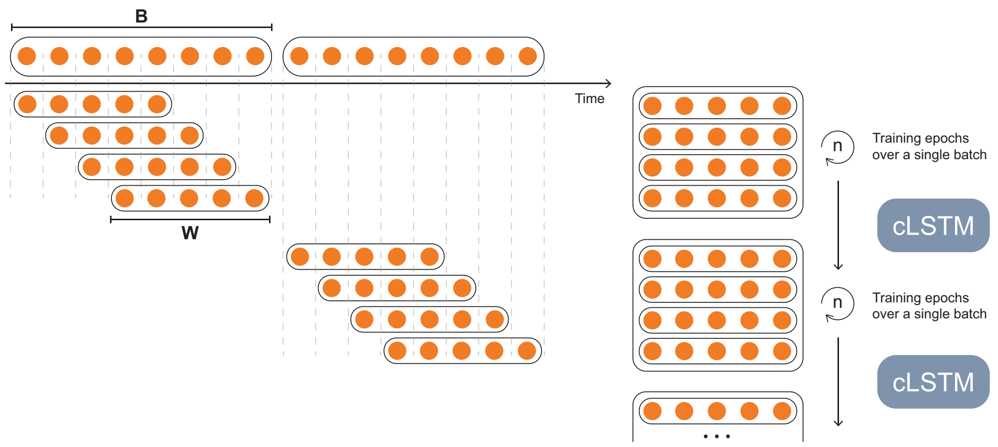

B
Time
W
n
Training epochs over a single batch
cLSTM
n
Training epochs over a single batch
cLSTM
...

Fig. 2: Proposed incremental sub-batching procedure for cLSTM, utilizing a batch size B and a sliding window size W.

data reordering based on a hidden feature [22]. The challenge is more pronounced when dealing with time-dependent data streams due to the lack of relevant tailored data.

To address this issue, we propose TENET, a novel systematic way to test the underlying guarantees of data stream classifiers and explore the impact of temporal dependence. By generating specific correlations in the past, the analysis of models that handle both concept drift and temporal dependence becomes more robust. TENET aims to mitigate the problems associated with selecting data streams in the experimental evaluation of stream classifiers and drift detectors.

The main component is a meta-generator that creates time-dependent data streams to define novel synthetic, not i.i.d. benchmarks. This meta-generator can be applied to any SML data generator, e.g., Sine [3], SEA [23], STAGGER [3], which are limited to generating concept drift. We propose introducing *temporal injection* to address this limitation. The temporal injection operates as follows: starting with a data stream containing instances (x_t, y_t), where x_t represents a feature vector and y_t represents its corresponding label, at each time step, the label y_t is replaced by y_t^* using Equation 3.

$$y_t^* = \text{mode}\,(y_t, y_{t-1}, \dots, y_{t-k}) \tag{3}$$

The order k is chosen to be even to avoid ties, as the label y_t is also considered. Furthermore, it is assumed that the original class label vector y contains sequences of the same class label and does not consist solely of alternating class labels. With an order $k = 2$, $y_t^* = \text{mode}\,(y_t, y_{t-1}, y_{t-2})$. In this scenario, there are two main cases to consider. If y_{t-1} and y_{t-2} are discordant, the single instance x_t classification should proceed as normal. However, if y_{t-1} and y_{t-2} have the same value, the information of x_t is not relevant for classification purposes, and it is sufficient to use the label of the two previous instances. If the classifier only relies on the instance at time t, it will lose important information for correctly classifying incoming data. The new classification problem, as shown in Fig. 1 for Sine1 data stream, introduces a novel challenge due to the correlations between the label of both attributes and past labels at different lags. The original classification problem contains 4 concept drifts, i.e., $x_1 - sin(x_2) > 0$ for 1^{st} and 3^{rd} concepts, and $x_1 - sin(x_2) < 0$ for 2^{nd} and 4^{th} concepts. Compared to this, the data stream obtained with TENET using temporal injection has a much higher correlation of features and past labels. For all 4 concepts, depending on the value of the past label, we have 2 different classification problems, which are not linearly separable as in the original setting.

IV. cLSTM BASELINE

The second component of TENET is a baseline capable of handling concept drift and temporal dependence in the data stream. In this perspective, we define a continuous version of the Long Short-Term Memory, cLSTM. cLSTM architecture is the minimum adaptation of LSTM for non-i.i.d streaming scenarios due to its ability to learn sequences, handle incrementally past correlations, and adapt to concept drift.

A. *Incremental Sub-batching*

The first step is organizing data to support incremental learning for cLSTM. Samples are accumulated in batches via a tumbling window [24] as data items arrive in the stream and are provided to the model for prediction and training. However, LSTM requires batches of sequences as input for training. We implement a fixed-size tumbling window to recreate the sequence concept within the incoming stream, converting a batch of samples into a batch of sequences. Data points are ordered by arrival time and divided into sub-batches using a sliding window method. We adopt the sub-batching procedure in Figure 2 to buffer data points in a batch of size B and build sequences using a window of size W, producing B-W+1 sequences per batch while preserving temporal order.

B. *Continuous LSTM*

The training method for LSTM networks typically involves several steps on batch data, allowing for deep structures and multiple epochs. Since Stochastic Gradient Descent (SGD) is

Algorithm 1 cLSTM training

```
Input: Data stream $S$, Batch size $B$,
Epochs $E$, Window size $W$.
1: $batch \leftarrow list(), model \leftarrow new\ cLSTM()$
2: for all $(X_t, y_t)$ in $S$ do
3:     Append $(X_t, y_t)$ to $batch$
4:     if $|batch| = B$ then
5:         $X, Y \leftarrow BuildSequences(batch, W)$
6:         $pred \leftarrow model.predict(X)$ //many to many(B, W)
7:         $Ypred \leftarrow average(pred)$ //from (B, W) to (B, 1)
8:         $Evaluate(Ypred, Y)$
9:         cLSTM $\leftarrow model.update(X, Y, E)$ //training
10:        $batch \leftarrow list()$
11:    end if
12: end for
```

an iterative method, we can exploit it to train cLSTM. Instead of being iterated on the whole dataset, the same approach is applied to the sub-batches, as shown in Fig. 2. The weights of cLSTM are updated with SGD using new data each time a new batch arrives. Algorithm 1 details the cLSTM learning process. We buffer the data stream in a batch with size B (Line 3) and create the B sub-batches (Line 5) through the process in Section IV-A. Then, using the prequential evaluation [25], cLSTM generates a probability distribution for each data point in the B sub-batches (Line 6). The probability distribution for each data point's target classes is determined by averaging the probability distributions associated with all the sub-batches the data point is a part of. (Line 7). Finally, it back-propagates the error, calculated by comparing the current value with the prediction made for that specific instant (Line 9). This way, the error is back-propagated through the network once per batch, allowing the system to handle data streams.

Our approach reduces the risk of overfitting for two main reasons. Notably, cLSTM has a single hidden layer. The challenge in this context is finding an appropriate network depth. If the model is too complex, the learning process will converge slowly and thus lose the desired property of online learning. On the other hand, if the model is too simple, the learning capacity will be too restricted, and it will be difficult to learn complex patterns. Secondly, propagating the sub-batches and back-propagating the error through the network is effective with a reduced configuration. In this way, cLSTM can handle evolving data streams. Indeed, a consequent advantage is the smooth adaptation in dynamic data streams where the target function evolves. In fact, by preventing the SGD from overfitting on a specific concept, we enable SGD to continue to learn and thus manage any changes in subsequent data.

We argue that sequential models like cLSTM have great potential in not i.i.d. streaming environments with temporal dependence and concept drift. In addition to using a single hidden layer to simplify the architecture and avoid overfitting scenarios, the hidden state and memory cell are reset to 0 after each sample sequence is received. This configuration allows better adaptation to new data in case of concept drift. Furthermore, there is no pre-training in the execution, thus allowing the continuous adaptation capability of the SGD to be fully utilized. cLSTM is trained using a reduced number of epochs on each batch to prevent overfitting during training. The learning rate is fixed, and there is no decay, which is common and effective practice in batch scenarios where the objective is to converge to a fixed point. However, under a streaming scenario, this could cause the SGD to react more slowly to concept drift and eventually get stuck in one concept.

V. Experiments

This section presents a comprehensive empirical evaluation of state-of-the-art SML classifier and cLSTM using Tenet. Initially, we analyze the impact of the order of temporal augmentation. Subsequently, we employ well-established synthetic and real-world data streams to rigorously evaluate the proposed benchmarking framework.

A. *Data streams from SML's Literature*

Simulating drift in data can potentially introduce bias into experimental evaluations. To mitigate this risk, we used real-world data streams without temporal injection. Table I summarizes their main characteristics. It is worth noting that, apart from the Insects dataset, the real-world SML data streams exhibit dependencies; Elec2 and Weather also demonstrate first-order temporal dependence, indicated by $P(T) > P(M)$ [6]. We employed published data streams or generators from the River library [26] with original parameterization. Concept drift was simulated every 25,000 samples using different concepts from each stream, with three abrupt or three gradual drifts, each having amplitudes of 1 or 5k instances, respectively. Exceptions include the Hyperplane and RBF streams, where we adjusted the magnitude of change and speed of change using 0.2 and 1.0 to create abrupt or gradual drifts.

The SEA Concepts data stream [23] includes three attributes, two of which are relevant for classification, with the two-class decision boundary defined by $x_1 + x_2 = \theta$, where x_1 and x_2 are the two relevant features, and θ is a predefined threshold. SINE1 and SINE2 data streams [3] each have two concepts and four dimensions, with classification functions defined by $x_1 = sin(x_2)$ and $x_1 = 0.5 + 0.3sin(3\pi x_2)$, respectively. The STAGGER data simulates drifts with boolean functions based on three features (size, shape, color) [3]. Agrawal data stream generator [27] produces a stream with nine features and has 10 functions for generating binary class labels from those features. The Random Radial Basis Function (RBF) generator creates a stream by generating random centroids with positions, deviations, labels, and weights, then producing new samples by randomly selecting centroids and adding Gaussian noise. The Hyperplane generator produces a stream for binary classification, with a decision boundary defined by a hyperplane that can be smoothly rotated to simulate drift, adjusting feature weights with a probability of reversing drift direction.

The Electricity dataset [28] predicts electricity price changes based on consumption and prices in the same and nearby

							Original		Temp. Inj.	
Data stream	#Samples	#Feat.	#Class	Type	I.	I.D.	$P(M)$	$P(T)$	$P(M)$	$P(T)$
SEA	100K	3	2	*Synth*	✓	✗	0.64	0.56	0.73	0.81
STAGGER	100K	2	2	*Synth*	✓	✗	0.64	0.54	0.71	0.79
Sine1	100K	2	2	*Synth*	✓	✗	0.50	0.50	0.51	0.75
Sine2	100K	2	2	*Synth*	✓	✗	0.50	0.51	0.51	0.76
Hyperplane	100K	10	2	*Synth*	✓	✗	0.51	0.51	0.50	0.75
RBF	100K	10	2	*Synth*	✓	✗	0.51	0.50	0.51	0.74
Agrawal	100K	9	2	*Synth*	✓	✗	0.50	0.53	0.51	0.79
Elec2	45312	5	2	*Real*	✗	✗	0.58	0.85	-	-
Insects	57018	33	2	*Real*	✓	✗	0.51	0.50	0.50	0.73
PowerSupply	29928	2	24	*Real*	✗	✗	0.04	0	-	-
Rialto	82250	27	10	*Real*	✗	✗	0.10	0	-	-
Weather	18159	8	2	*Real*	✗	✗	0.68	0.70	-	-

TABLE I: Characteristics of evaluated data streams. I: Independent, I.D.: Identically Distributed. ✗ for I.D. implies the presence of concept drifts. $P(M)$ is the prior probability of the majority class, $P(T) = \sum_{i=1}^{k} P(i,i)$ characterizes first order temporal dependence.

regions. The Power supply dataset [20] spans three years (1995-1998), aiming to predict the specific hour of the day (1 out of 24 possibilities) based on power supply, influenced by seasonal, weather, and time-of-day variations, and distinctions between weekdays and weekends. The Rialto Bridge Timelapse dataset [29] derives from time-lapse videos using a fixed webcam, involving classifying buildings under changing weather and lighting conditions. The Weather dataset [30] covers 50 years of weather data with features including temperature, dew point, pressure, visibility, and wind speed, predicting rain occurrence. The Insects dataset [20] originates from a real-world streaming application utilizing optical sensors to recognize flying insect species in real time. In our experiments, we employed the *'incremental-balanced'* setting, considering only two classes, i.e., *'ae-aegypti-female'* and *'ae-aegypti-male'*. As the only real dataset without temporal dependence, we introduced temporal patterns through temporal injection to facilitate comparisons in time-dependent scenarios.

B. Set-up

TENET benchmarking framework facilitates the formulation of precise research questions guiding our experimental campaign. Specifically, the questions we aimed to address were:

RQ1: How does the temporal dependence within data streams impact the learning process of state-of-the-art SML classifiers?

RQ2: To what extent do cLSTM enhance classification performance when learning from evolving data streams with temporal dependence?

We employ a combination of statistical measures to evaluate the compared models thoroughly. Specifically, we utilise Kappa statistics and Kappa-Temporal statistics [5] to assess the accuracy of the models. Kappa-Temporal statistics is a modified version of Kappa that considers the difference between a classifier's prequential accuracy and the accuracy of a "naive" classifier that always predicts the last observed class label, known as the No-Change or Persistent classifier. The Persistent classifier aligns with principles commonly used as a baseline in time series forecasting, equivalent to an ARMA(1,0) model in autoregressive time series analysis [11].

We utilise the prequential evaluation methodology [25] to evaluate the models. Unlike a periodic holdout [31], prequential evaluation observations first to test and then update the model. It is important to note that we did not apply cross-validation strategies as it would break the temporal dependence within the stream. To avoid distorting the instance-based nature of the HAT and ARF models, we applied prequential evaluation for both instance-based SML approaches and the cLSTM batch-based algorithm, comparing every *batch size* data points the cLSTM predictions with the last *batch size* predictions of SML models. To ensure the robustness of our results, we conducted 10 runs for each test and took the average performance. In addition, we employed non-parametric tests using the methodology from [32] to verify statistically significant differences between algorithms. We utilised the Friedman test with $\alpha = 0.05$ and the Nemenyi post-hoc test.

We compared cLSTM with Hoeffding Adaptive Tree (HAT) [33] and Adaptive Random Forest (ARF) [34], two widely used SML approaches. Additionally, we included their temporally augmented versions, namely SWT HAT and SWT ARF, to handle temporal dependence. For all SML models, the configuration used is defined on the default parameters, which are fine-tuned on analyses that considered a large portion of the datasets also used in this analysis. This selection process ensures fairness in comparison. Furthermore, Section V-C investigates the optimal order for Temporally Augmented models, determining that with a temporal injection of order 3 in the data streams, SWT HAT and SWT ARF achieve optimal

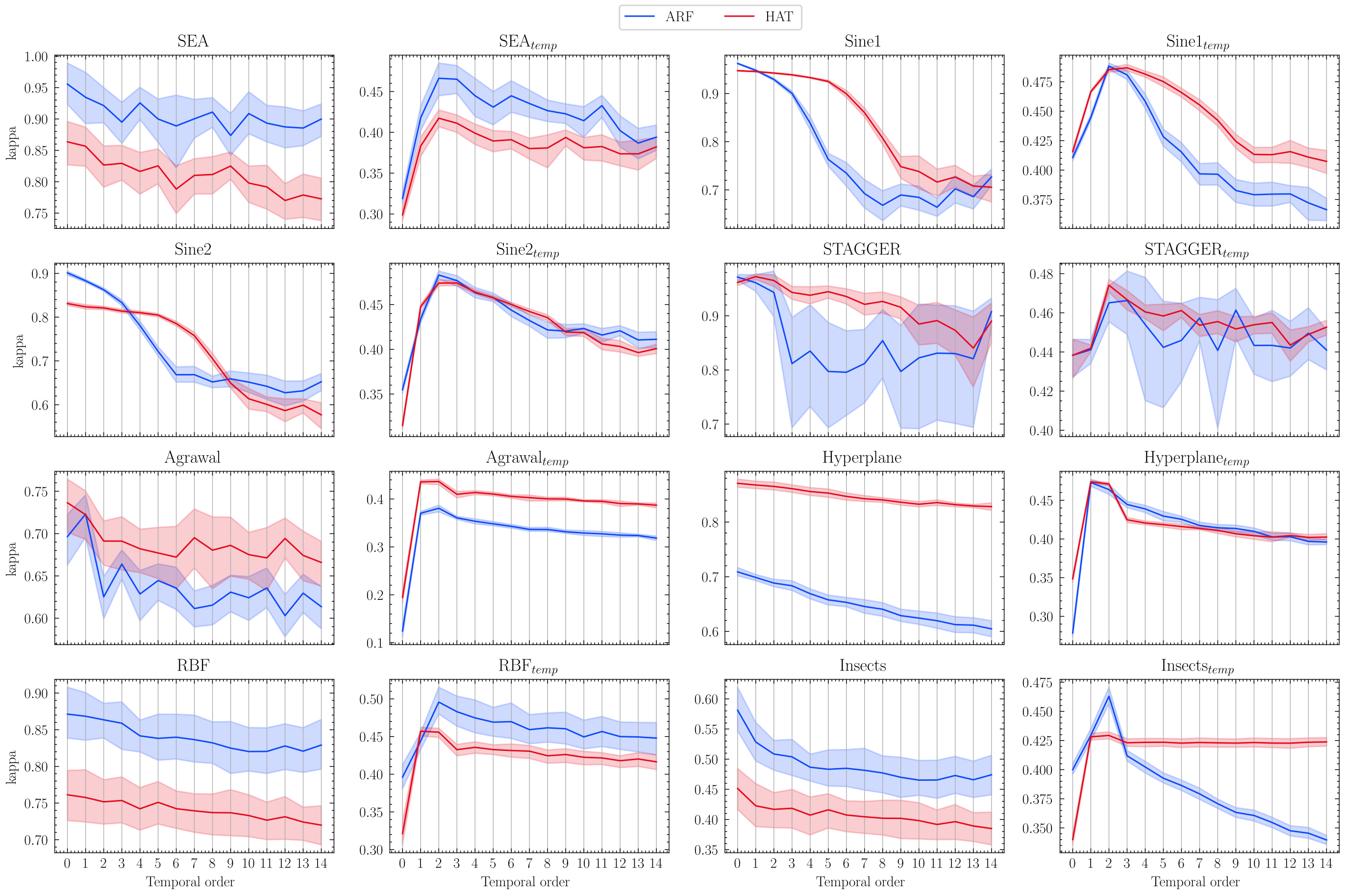


Fig. 3: Kappa Statistics with different orders of Temporal Augmentation for the Hoeffding Adaptive Tree and the Adaptive Random Forest models.

augmentation with two past labels. Concerning cLSTM configuration, the hyperparameters were selected from a predefined set of values, favouring the best average performance. The choice of sub-batch size was made to have sequences at the input of the cLSTM long enough to favour its training. The configuration parameters of the cLSTM are the following; *training epochs = 10*, *learning rate = 0.1*, *sub-batch size = 10*, *batch size = 128*, *neurons = 50*, *1 hidden layer*.

In all tables, *temp* denotes the application of the temporal order injection, bold values indicate the best result for a given experiment, while *Avg* summarizes the scores as the averages of each model's values.

C. Analysis of Temporal Augmentation Effectiveness

In this section, we investigate the impact of temporal augmentation on the Hoeffding Adaptive Tree (HAT) and Adaptive Random Forest (ARF) models. We conducted an analysis across various temporal orders, ranging from 0 to 14, using all the data streams. Temporal injection from TENET was applied to evaluate how incorporating past embeddings affects their performance. Each configuration underwent 20 runs to maximize variability across streams and to highlight potential changes in confidence intervals based on the order used for temporal augmentation. This assessment aims to determine the optimal temporal embedding order that improves model accuracy without adding unnecessary complexity.

The results depicted in Fig. 3 highlight significant performance improvements in streams with temporal injection, demonstrating effective utilization of the inherent temporal structure for enhanced classification accuracy and partially addressing *RQ1*. Our findings reveal that increasing the order of past embeddings leads to improvements in classification accuracy up to order 2. However, beyond this threshold, the incremental benefits diminish, suggesting an optimal range for embedding historical features. This trend is clearly illustrated for data with temporal injection, where performance gains stabilize after reaching a specific order of past embeddings. Conversely, with data lacking temporal dependence, the performance increment is negligible, and for sufficiently large temporal orders, it significantly degrades performance.

It is noteworthy that as the order of temporal embedding increases, the confidence interval around the performance metric also widens. This reflects the growing uncertainty of

	Data stream	Drift type	HAT	ARF	SWT HAT	SWT ARF	cLSTM	No-Change
Original	SEA	A	84.59 (4)	**96.77 (1)**	83.19 (5)	92.94 (3)	94.02 (2)	-0.92
		G	83.04 (4)	**95.08 (1)**	78.5 (5)	91.86 (3)	92.44 (2)	-0.48
	STAGGER	A	96.22 (5)	97.86 (4)	98.24 (3)	**99.76 (1)**	99.21 (2)	-0.45
		G	93.37 (5)	95.58 (4)	96.36 (3)	**97.34 (1)**	96.83 (2)	-0.42
	Sine1	A	94.73 (2)	**96.25 (1)**	94.36 (3)	92.9 (4)	92.61 (5)	-0.99
		G	83.79 (2)	**84.51 (1)**	83.28 (3)	81.34 (5)	83.04 (4)	-0.47
	Sine2	A	83.04 (4)	**90.87 (1)**	82.43 (5)	86.87 (2)	83.69 (3)	-0.24
		G	70.37 (4)	**79.3 (1)**	70.18 (5)	78.69 (2)	73.55 (3)	-0.5
	Hyperplane	A	79.75 (2)	71.84 (4)	78.19 (3)	70.28 (5)	**84.09 (1)**	-0.29
		G	78.23 (2)	71.31 (4)	77.94 (3)	70.16 (5)	**83.01 (1)**	-0.78
	RBF	A	75.39 (5)	87.41 (2)	75.46 (4)	86.74 (3)	**88.54 (1)**	-0.49
		G	72.87 (5)	**85.21 (1)**	74.92 (4)	83.71 (3)	84.21 (2)	-0.53
	Agrawal	A	**72.9 (1)**	69.01 (3)	68.73 (4)	61.36 (5)	71.88 (2)	-0.9
		G	**57.32 (1)**	54.69 (3)	51.33 (4)	50.5 (5)	56.34 (2)	-0.85
	Original avg		*80.4 (3.29)*	*83.98 (2.21)*	*79.51 (3.86)*	*81.75 (3.36)*	***84.53 (2.29)***	-0.59
Temporal Injection	SEA_{temp}	A	29.48 (5)	32.04 (4)	41.99 (3)	46.34 (2)	**80.61 (1)**	47.36
		G	28.98 (5)	32.0 (4)	42.91 (3)	46.52 (2)	**74.56 (1)**	47.53
	STAGGER_{temp}	A	43.84 (4)	43.82 (5)	46.66 (2)	46.34 (3)	**85.89 (1)**	48.37
		G	42.24 (5)	43.31 (4)	45.58 (3)	45.65 (2)	**78.76 (1)**	47.82
	Sine1_{temp}	A	41.79 (4)	41.01 (5)	48.49 (2)	47.25 (3)	**75.38 (1)**	48.77
		G	37.07 (4)	36.55 (5)	47.84 (2)	46.9 (3)	**69.35 (1)**	49.26
	Sine2_{temp}	A	31.87 (5)	36.03 (4)	46.34 (3)	47.45 (2)	**64.59 (1)**	49.19
		G	28.24 (5)	31.91 (4)	46.68 (3)	47.7 (2)	**62.1 (1)**	48.98
	Hyperplane_{temp}	A	34.16 (4)	27.99 (5)	46.98 (3)	47.92 (2)	**66.59 (1)**	46.59
		G	34.5 (4)	28.19 (5)	47.81 (3)	51.58 (2)	**58.73 (1)**	45.84
	RBF_{temp}	A	34.13 (5)	39.79 (4)	44.95 (3)	48.16 (2)	**58.32 (1)**	49.16
		G	26.43 (5)	28.76 (4)	45.6 (2)	43.28 (3)	**56.01 (1)**	47.86
	Agrawal_{temp}	A	19.59 (4)	12.85 (5)	42.96 (2)	37.47 (3)	**53.05 (1)**	48.05
		G	15.64 (4)	10.95 (5)	44.25 (2)	36.7 (3)	**51.05 (1)**	49.19
	Temp inj avg		*32.0 (4.5)*	*31.8 (4.5)*	*45.65 (2.57)*	*45.66 (2.43)*	***66.78 (1.0)***	48.14
	Overall avg		*56.2 (3.89)*	*57.89 (3.36)*	*62.58 (3.21)*	*63.7 (2.89)*	***75.66 (1.64)***	23.78

TABLE II: Prequential Kappa-Statistics for the original and the TENET's temporal injected versions of the *synthetic data streams*. Drift type: A: Abrupt; G: Gradual. Bold values indicate the best result for a given experiment, and rankings are shown in brackets. *Avg* summarizes the scores as the averages of each model's values.

the model as more potentially noisy past data is incorporated. While temporal augmentation can improve performance, it simultaneously introduces more variability into the model's predictions. This underscores the importance of carefully balancing the amount of historical data used. The analysis highlights the need to select an optimal temporal window for feature augmentation. Including too few past features may underutilize temporal dependencies, while incorporating too many can introduce noise and redundancy, leading to diminishing returns. Therefore, finding the right balance is crucial for maximizing the effectiveness of temporal augmentation in data stream classification.

D. Positioning SML against a cLSTM baseline

In the analysis, only Kappa Statistics is presented for conciseness. The results achieved with Kappa Temporal align with those depicted in this section using standard Kappa Statistics. Table II displays the Kappa Statistics results for the synthetic data stream. To facilitate comparison, we included the performance of the No-Change classifier, though not considered in the ranking. For the *original* data stream, ARF emerges as the top-performing algorithm, confirming the assumption of data independence. Both versions of SWT underperform compared to their base models, except for the STAGGER dataset, as confirmed also by the previous analysis in Section V-C. The latter case suggests that past information can also be valuable for inferencing with data without explicit temporal dependence. The performance of cLSTM further underscores this, demonstrating performance at least comparable to the SWT approaches.

The best-performing algorithm with the synthetic data streams with temporal injection is cLSTM. SWT models show improvements compared to their base models but remain primarily below the cLSTM. This suggests that temporal dependence significantly impacts a data stream's classification performance, and existing solutions, such as Temporal Augmentation, are insufficient to exploit complex relations with past data. This point is particularly evident in Figure 4, where the difference becomes evident in both Kappa Statistics and Kappa Temporal Statistics. It is also worth noting that the Temporally Augmented models struggle to outperform the No-Change classifier, obtaining almost null Kappa Temporal Statistics. Under these circumstances, augmenting the input vector with past labels is insufficient for correct classifications. It is also relevant that the results are similar when comparing abrupt and gradual drift settings. Notably, gradual drift further complicates the classification.

Referring to the first research question *RQ1*, we empirically

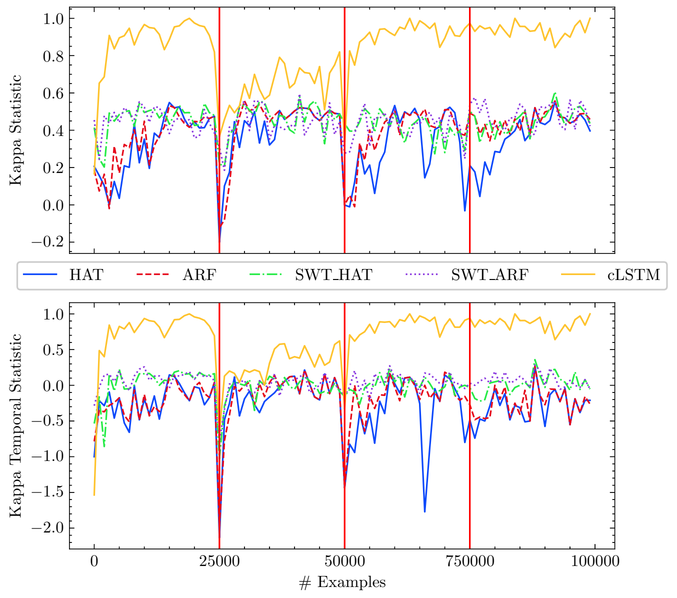


Fig. 4: Kappa Statistics and Kappa Temporal Statistics over the STAGGER data stream with abrupt drift. The red lines highlight the concept drifts, occurring every 25,000 samples.

Data stream	HAT	ARF	Swt HAT	Swt ARF	cLSTM
Elec2	60.9 (5)	73.8 (2)	66.3 (4)	**75.1 (1)**	68.6 (3)
Insects	45.6 (4)	**58.3 (1)**	42.2 (5)	50.8 (3)	52.0 (2)
Insects$_{temp}$	33.7 (5)	40.9 (4)	43.1 (3)	46.6 (2)	**75.3 (1)**
PowerSupply	12.5 (4)	11.2 (5)	14.0 (3)	67.7 (2)	**78.7 (1)**
Rialto	23.4 (5)	71.3 (3)	25.8 (4)	**92.6 (1)**	90.2 (2)
Weather	29.7 (3)	30.2 (2)	27.3 (4)	26.4 (5)	**42.2 (1)**
Overall avg	*34.3 (4.3)*	*47.6 (2.8)*	*36.5 (3.8)*	*59.9 (2.3)*	***67.8 (1.7)***

TABLE III: Prequential Kappa-Statistics with *real data streams*. Bold values indicate the best result for a given experiment, and rankings are shown in brackets. *Avg* summarizes the scores as the averages of each model's values.

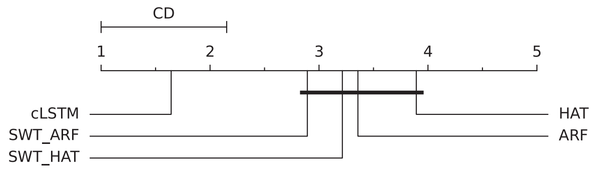
confirm that the temporal dependence significantly affects the learning process of SML classifiers. Furthermore, the cLSTM demonstrates superior drift adaptation while effectively managing temporal dependencies, thereby affirmatively addressing the second research question *(RQ2)*. This assertion is supported by the results of the post-hoc Nemenyi test, detailed in Fig. 5a, which validate the performance improvements achieved using cLSTM. Notably, among the compared approaches, cLSTM emerges as statistically superior, whereas the other methods, i.e., ARF and HAT and their respective Swt versions, do not exhibit significant differences.

Table III presents the results for the real data. In this case, the performance is more balanced, with cLSTM performing better overall. Similar to the synthetic data streams, temporal augmentation achieves improvements in the data, whether it contains temporal dependence or not. Overall the cLSTM is the best or very close to the best-performing algorithm. We want to emphasize that this result is due to the ability to learn the sequence within the stream with the sub-batching procedure. It then becomes critical to apply these models to streams to understand when treating them as a time series and achieving better performance is possible. Opposite to the synthetic data streams, the post-hoc Nemenyi test discussed in Fig. 5b does not identify significant small groups but consistently ranks cLSTM before SML's algorithms. The difference between the number of models compared and the number of data streams used is not large enough to obtain a critical distance to separate the approaches into small groups. An extended version of this study will use more data streams. Nonetheless, it is still worth noting that the compared models are ordered not by the type of approach used (Temporal Augmentation better than standard base models) but rather by the model (ARF better than HAT). This result further confirms that Temporal Augmentation is a partial solution and does not achieve a tangible improvement in the case of complex problems, such as the real data used.

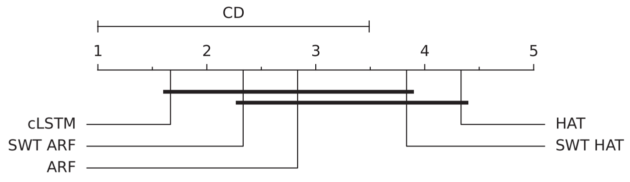
VI. Conclusion

This paper pioneers Tenet, the first benchmarking framework for evaluating data stream classifiers with temporal dependence. Its main component is a novel meta-generator for time-dependent data streams. Moreover, we proposed as a baseline a continuous version of LSTM, namely cLSTM. cLSTM is the smallest adaptation of an LSTM, which is meant to learn sequences suited for data streams that present concept drifts. Instead, this work focuses on analysing the sequential models' behaviours using a simplified scenario in the presence of temporal dependence. We argue that incremental sequential models, such as cLSTM, are promising techniques for a streaming scenario to take temporal dependence into account. In all experiments using Tenet, the temporal order impacts the learning process of streaming algorithms regardless of the drift type, positively answering *RQ1*. Moreover, it demonstrates the insufficiency of augmenting feature space with past labels, i.e., Temporal Augmentation. More advanced techniques are needed when learning from evolving data streams with temporal dependence. We also provided empirical evidence of the effectiveness of cLSTM in this scenario, positively answering *RQ2*, showing better performance, especially with temporal dependence. This work paves the way for more robust data stream classifiers, able to tame concept drift and temporal dependence and potentially exploit during learning.

The main limitations of cLSTM can be identified by its high computational complexity and sensitivity to hyperparameters configuration, requiring a significant amount of data. Data stream classifiers are instead constrained by limited memory and processing time constraints. However, Temporal Augmentation also has some drawbacks in this context. Its time complexity increases with the number of past values added to the input vector (more oversized windows capture longer horizons of dependence), it does not model directly the temporal structure, and overly large temporal orders degrade model performance. Therefore, we intend to explore more advanced (e.g. non-linear) methods for incorporating



(a) Synthetic data, CD = 1.15.
cLSTM; SWT ARF $\succeq$ SWT HAT $\succeq$ HAT.



(b) Real data, CD = 2.49.
cLSTM $\succeq$ SWT ARF $\succeq$ ARF $\succeq$ SWTHAT;
SWT ARF $\succeq$ ARF $\succeq$ SWT HAT $\succeq$ HAT.

Fig. 5: Nemenyi post-hoc test with 95% confidence level.

temporal information into data stream classification in future works. Ideas from time series analysis could be adapted to improve the performance of these models. For example, Echo State Networks presents a promising approach for embedding temporal signals into vectors, effectively converting time series data into i.i.d. data streams that traditional SML methods can process. Finally, we intend to extend TENET temporal injection to multiclass problems and to accommodate the delayed and weakly supervised settings, which were left outside the scope of this work and have much importance for more real-world not i.i.d. streaming scenarios.

REFERENCES

[1] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine learning for data streams: with practical examples in MOA*. MIT press, 2018.

[2] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[3] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues, "Learning with drift detection," in *SBIA*, ser. Lecture Notes in Computer Science, vol. 3171. Springer, 2004, pp. 286–295.

[4] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, 2019.

[5] A. Bifet, J. Read, I. Zliobaite, B. Pfahringer, and G. Holmes, "Pitfalls in benchmarking data stream classification and how to avoid them," in *ECML/PKDD (1)*, ser. LNCS, vol. 8188. Springer, 2013, pp. 465–479.

[6] I. Zliobaite, 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, 2015.

[7] A. Bifet, "Classifier concept drift detection and the illusion of progress," in *ICAISC (2)*, ser. LNCS, vol. 10246. Springer, 2017, pp. 715–725.

[8] J. Read, R. A. Rios, T. Nogueira, and R. F. de Mello, "Data streams are time series: Challenging assumptions," in *BRACIS (2)*, ser. LNCS, vol. 12320. Springer, 2020, pp. 529–543.

[9] Y. Song, J. Lu, H. Lu, and G. Zhang, "Learning data streams with changing distributions and temporal dependency," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[10] G. Ziffer, A. Bernardo, E. Della Valle, V. Cerqueira, and A. Bifet, "Towards time-evolving analytics: Online learning for time-dependent evolving data streams," *Data Science*, vol. 6, no. 1-2, pp. 1–16, 2023.

[11] G. E. P. Box and G. M. Jenkins, "Time series analysis: Forecasting and control," 1994.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *ICMLA*. IEEE, 2018, pp. 1394–1401.

[14] C. Anagnostopoulos, D. K. Tasoulis, N. M. Adams, N. G. Pavlidis, and D. J. Hand, "Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification," *Stat. Anal. Data Min.*, vol. 5, no. 2, pp. 139–166, 2012.

[15] D. A. Bodenham and N. M. Adams, "Continuous monitoring for changepoints in data streams using adaptive estimation," *Stat. Comput.*, vol. 27, no. 5, pp. 1257–1270, 2017.

[16] Q. Duong, H. Ramampiaro, and K. Nørvåg, "Applying temporal dependence to detect changes in streaming data," *Appl. Intell.*, vol. 48, no. 12, pp. 4805–4823, 2018.

[17] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pp. 366–381.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[19] N. Gunasekara, B. Pfahringer, H. M. Gomes, and A. Bifet, "Survey on online streaming continual learning," in *IJCAI*. ijcai.org, 2023, pp. 6628–6637.

[20] V. M. A. de Souza, D. M. dos Reis, A. G. Maletzke, and G. E. A. P. A. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Min. Knowl. Discov.*, vol. 34, no. 6, pp. 1805–1858, 2020.

[21] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, 2017.

[22] P. Sobolewski and M. Wozniak, "Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors," *J. Univers. Comput. Sci.*, vol. 19, no. 4, pp. 462–483, 2013.

[23] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *KDD*. ACM, 2001, pp. 377–382.

[24] A. Arasu, S. Babu, and J. Widom, "The CQL continuous query language: semantic foundations and query execution," *VLDB J.*, vol. 15, no. 2, pp. 121–142, 2006.

[25] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, 2013.

[26] 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," *J. Mach. Learn. Res.*, vol. 22, pp. 110:1–110:8, 2021.

[27] R. Agrawal, T. Imielinski, and A. N. Swami, "Database mining: A performance perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 6, pp. 914–925, 1993.

[28] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," University of New South Wales, School of Computer Science and Engineering, Tech. Rep., 1999.

[29] V. Losing, B. Hammer, and H. Wersing, "KNN classifier with self adjusting memory for heterogeneous concept drift," in *ICDM*. IEEE Computer Society, 2016, pp. 291–300.

[30] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, 2013.

[31] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *KDD*. ACM, 2009, pp. 329–338.

[32] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[33] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *IDA*, ser. LNCS, vol. 5772. Springer, 2009, pp. 249–260.

[34] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfahringer, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Mach. Learn.*, vol. 106, no. 9-10, pp. 1469–1495, 2017.