Unsupervised Concept Drift Detection From Deep Learning Representations in Real-Time

Unsupervised Concept Drift Detection From Deep Learning Representations in Real-Time

Salvatore Greco, Bartolomeo Vacchetti, Daniele Apiletti, and Tania Cerquitelli, *Member, IEEE*

***Abstract*—Concept drift is the phenomenon in which the underlying data distributions and statistical properties of a target domain change over time, leading to a degradation in model performance. Consequently, production models require continuous drift detection monitoring. Most drift detection methods to date are supervised, relying on ground-truth labels. However, they are inapplicable in many real-world scenarios, as true labels are often unavailable. Although recent efforts have proposed unsupervised drift detectors, many lack the accuracy required for reliable detection or are too computationally intensive for real-time use in high-dimensional, large-scale production environments. Moreover, they often fail to characterize or explain drift effectively. To address these limitations, we propose DRIFTLENS, an unsupervised framework for real-time concept drift detection and characterization. Designed for deep learning classifiers handling unstructured data, DRIFTLENS leverages distribution distances in deep learning representations to enable efficient and accurate detection. Additionally, it characterizes drift by analyzing and explaining its impact on each label. Our evaluation across classifiers and data-types demonstrates that DRIFTLENS (i) outperforms previous methods in detecting drift in 15/17 use cases; (ii) runs at least 5 times faster; (iii) produces drift curves that align closely with actual drift (correlation ≥ 0.85); (iv) effectively identifies representative drift samples as explanations.**

***Index Terms*—Concept drift, data drift, drift detection, drift explanation, deep learning, NLP, computer vision, audio.**

I. INTRODUCTION

THE basic assumption in deep learning is that training data mimics the real world. Yet, deep learning models are typically trained and evaluated on static datasets. However, the world is dynamic, and what the model learned during training may no longer be valid. The underlying data distribution and statistical properties of the target domain may change over time, leading to a degradation of the model's performance (or decay).

Received 29 July 2024; revised 4 April 2025; accepted 18 July 2025. Date of publication 29 July 2025; date of current version 15 September 2025. This work was supported in part by the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" through - program "RESTART" under Grant PE00000001 and in part by the "National Centre for HPC, Big Data and Quantum Computing," under Grant CN000013 (approved under the M42C Call for Proposals - Investment 1.4 - Notice "National Centers" - D.D. No. 3138 of 16.12.2021, admitted for funding by MUR Decree No. 1031 of 17.06.2022). Recommended for acceptance by R. Akbarinia. *(Corresponding author: Salvatore Greco.)*

The authors are with the Department of Control and Computer Engineering, Politecnico di Torino, Turin 10129, Italy (e-mail: salvatore_greco@polito.it).

Code: https://github.com/grecosalvatore/drift-lens

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2025.3593123, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2025.3593123

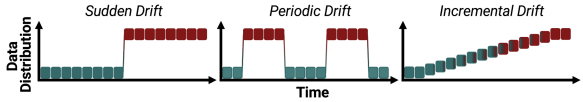


Fig. 1. Examples of drift patterns.

As we will formally define in Section II, changes in data distribution—known as *concept drift* [1]—can compromise the reliability of deep learning models in real-world applications [2]. To mitigate this, continuous monitoring of production models is crucial [3]. Effective monitoring should not only provide *early warnings* when drift occurs but also *characterize* and *explain* drift to assist humans in implementing *adaptive measures* to maintain model performance on evolving data.

As we will discuss in Section III, a large body of research has focused on concept drift detection through *supervised* methods, which rely on error rates or performance-based measures computed from ground-truth labels [1]. However, they exhibit limited applicability in many real-world applications where true labels are unavailable for newly processed data.

A parallel research effort has been devoted to *unsupervised* concept drift detection [4], [5], [6], [7]. Most methods compare the data stream with reference data using distribution distances, divergence measures, or statistical hypothesis tests. However, they are computationally intensive and ineffective for real-time concept drift detection in deep learning models processing large-scale, high-dimensional, unstructured, and unlabeled data streams [8]. In addition, they typically fail to characterize and explain drift, which is crucial to facilitate drift adaptation.

In tackling this, the contribution of this paper is twofold:

(1) We propose DRIFTLENS (Section IV), a novel *unsupervised* drift detection framework designed to detect *whether* and *when* drift occurs by exploiting distribution distances of deep learning representations from unstructured data, modeled as Gaussian distributions. In addition, DRIFTLENS performs *drift characterization* by determining and *explaining* the drift impact on each label. Thanks to its low complexity, it enables *real-time* drift detection, regardless of data dimensionality or volume.

(2) We conduct a comprehensive evaluation on several deep learning classifiers for text, image, and audio (Section V), and we attempt to answer the following four research questions (RQs):

(RQ1) *To what extent can it detect drift of varying severity without relying on true labels?*

(RQ2) *How can it be applied broadly and effectively across various data types, models, and classification tasks?*

(RQ3) *How efficient is it at detecting drift in near real-time?*

(RQ4) *To what extent can it accurately model, characterize, and explain the presence of drift over time?*

We found that DRIFTLENS (i) outperforms previous drift detectors in 15/17 use cases; (ii) is extremely fast, enabling real-time drift detection independent of data volumes (≤ 0.2 seconds, at least 5 times faster than other detectors); (iii) produces drift curves highly correlated with drift severity; (iv) effectively identifies representative drift samples as explanations.

II. PROBLEM FORMULATION

A. Concept Drift Definition

Concept drift can be defined as the phenomenon in which the underlying data distributions and statistical properties of a target data domain change over time. Drift can significantly affect the performance of deep learning models deployed in real-world applications. Various terms have been proposed to refer to *"concept drift"* [9], such as data drift, dataset shift, covariate shift, prior probability shift, and concept shift. Although each definition emphasizes a particular facet of drift, most works, similar to ours, broadly refer to all subcategories under the term *"concept drift"*. Formally, concept drift is defined as a change in the joint distribution between a time period $[0, t]$ and a time window $t+w$. Drift occurs in $t+w$ if:

$$P_{[0,t]}(X, y) \neq P_{t+w}(X, y) \tag{1}$$

Where X and y are the feature vectors and the target variable of each data instance (x_i, y_i), and $P_t(X, y)$ is the joint probability. The time window $t + w$ can be defined as a period or instant based on how the data stream is processed. The joint probability can be further decomposed as:

$$P_t(X, y) = P_t(y/X)P_t(X) = P_t(X/y)P_t(y) \tag{2}$$

Where, $P_t(X/y)$ is the class-conditional probability, $P_t(y/X)$ is the target labels posterior probability, $P_t(X)$ is the input data prior probability, and $P_t(y)$ is target labels prior probability. In classification tasks, concept drift can occur as a change in any of these terms: (1) $P(X)$: A drift in the input data. The marginal probability of the input features X changes (also known as data drift, covariance drift, or virtual drift). (2) $P(y/X)$: The relationships or conditional probabilities of target labels given input features change, but the input features do not necessarily change (usually referred to as concept or real drift). (3) $P(y)$: A change in the output data—labels and their probabilities change (usually referred to as label drift).

This work focuses on detecting drift in scenarios where no ground truth labels are available for new data. Therefore, drift must be detected in an *unsupervised* manner. Due to the lack of actual labels, the only possible drift that can be considered is the change in the prior probability of the features $P(X)$ [4].

B. Drift Patterns

Concept drift can occur in several patterns (Fig. 1):

Sudden/Abrupt Pattern: Drift can occur suddenly if the distribution of new samples changes rapidly. An example of sudden drift is a tweet topic classifier during the outbreak of the COVID-19 pandemic. At that point, the model was suddenly exposed to numerous text samples containing a new topic. Recognizing this scenario is crucial for initiating the retraining process and restoring the classifier's performance.

Periodic/Recurrent Pattern: Drift occurs repeatedly after the first observed event, with a seasonality unknown during training. For example, in an election year, the language and topics of discussion on social media can change significantly, affecting the sentiment and context in which certain words or phrases are used. After the election, discussions return to normal, but during another major event, such as another election, they may change again. Another example of recurring distribution changes is sporting events like the Olympics.

Incremental Pattern: The transition between concepts occurs gradually over time. For example, in image or object classification for autonomous vehicles, the model may encounter new vehicle types, such as scooters (i.e., Personal Light Electric Vehicles), that were not part of the original training data. Initially, these new vehicles may appear rarely and sporadically. However, they gradually become more commonplace, eventually becoming a permanent feature of the streetscape.

C. Application Scenario

We design DRIFTLENS so that it can be effectively exploited in a variety of application scenarios characterized by:

1) Unavailability of ground truth labels for new incoming data: Many applications involve classifying large-scale data productions (e.g., social media content) through deep learning models. Usually, actual labels for newly processed samples are unavailable. Consequently, both drift detection and adaptation must be performed in an unsupervised manner.

2) High data complexity and dimensionality: Most applications use unstructured data like texts, images, and audio, which are characterized by high dimensionality and sparsity, complexity, and the lack of a fixed structure, such as columns. These factors can undermine the effectiveness and increase the complexity of drift detection and characterization techniques.

In such a target scenario, our desiderata for the drift detection method are the following: (1) *Fast detection*. Drift should be detected as soon as possible and not only when it occurs with high severity; (2) *Real-time detection*. Drift detection should have low complexity to be performed in real-time; (3) *Drift characterization*. Information on drift patterns, the labels most affected by changes, and an explanation should be provided. DRIFTLENS addresses all these desiderata.

III. RELATED WORKS

There has been a significant effort in the field of concept drift detection [1], [6], [9], [10], [11], [12]. Drift detectors can be classified into two macro-categories based on the true labels availability assumption: (1) *supervised* and (2) *unsupervised*.

1) Supervised Concept Drift Methods: Most of the previous drift detection techniques are *supervised* [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26]. These methods typically rely on error rate-based measures or ensemble

models to assess degradation in the error rate over time (e.g., decrease in accuracy) [27]. However, they assume that the true labels are available with the new data or within a short period of time. In practice, the labels for new data are usually unavailable, and labeling them is very costly and time-consuming, limiting their applicability in real-world scenarios.

2) Unsupervised Concept Drift Methods: In contrast, *unsupervised* techniques do not require true labels [4], [6], [7], [8]. Our method falls into this category. As outlined by [4], unsupervised techniques can be further divided into: (2.i) *statistical-based*, (2.ii) *loss-based*, and (2.iii) *virtual classifier-based*.

2.i) Statistical-based Methods: Most of the unsupervised techniques [28], [29], [30], [31], [32], [33], [34], [35], [36], [37] rely on statistical hypothesis tests, two-sample tests, or divergence metrics between distributions to detect drift, such as Maximum Mean Discrepancy [28], Kolmogorov-Smirnov [38], Least-Squares Density Difference [30], or Cramér–von Mises [39]. These techniques have two main advantages. First, they can be applied similarly independently of the data type. Second, they do not require external models or resources. Their main limitation is that they are usually computationally intensive and ineffective in detecting drift affecting deep learning models applied to large-scale, high-dimensional, and unstructured data stream. This can affect their runtime and drift prediction performance [8].

2.ii) Loss-based Methods: These techniques exploit model loss functions to evaluate the similarity of new data points with previous ones, such as [40], [41], [42], [43]. Usually, they rely on autoencoders or are used in conjunction with supervised drift detectors. They are based on the assumption of a correlation between the increase in model losses and concept drift [44]. However, they have two main limitations. First, they require an external resource or model to perform drift detection. Second, being usually based on autoencoders, they are mainly suitable for computer vision models. Therefore, they are not agnostic to the data domain, and their usage on other data types requires specific implementations and design choices.

2.iii) Virtual Classifier-based Methods: Similarly, these techniques [45], [46], [47] use classifiers to detect drift. The general idea is to divide data into two sets, before and after a certain moment in time. If the classifier achieves an accuracy higher than random in classifying samples into the two classes, it implies divergent data properties between the two class distributions, suggesting the presence of drift. The main limitation is the need to train and maintain another model to detect drift, whose implementation is domain- and task-specific.

Drift Explanation: Some techniques attempt to provide human-readable explanations for drift rather than only detecting when it occurs [5], [48], [49], [50], [51]. For instance, [48] proposes a framework that trains a proxy model to classify or segment drifted samples [52], [53] and then leverages Explainable AI (XAI) [54], [55] to interpret drift through the proxy model. Other techniques provide feature-wise representations to interpret drift [56], [57], [58]. However, they usually struggle with high-dimensional or non-semantic features, like unstructured data. Despite its importance, research on drift explanations remains limited, especially for unstructured data where inputs lack clear semantics and are high-dimensional [48].

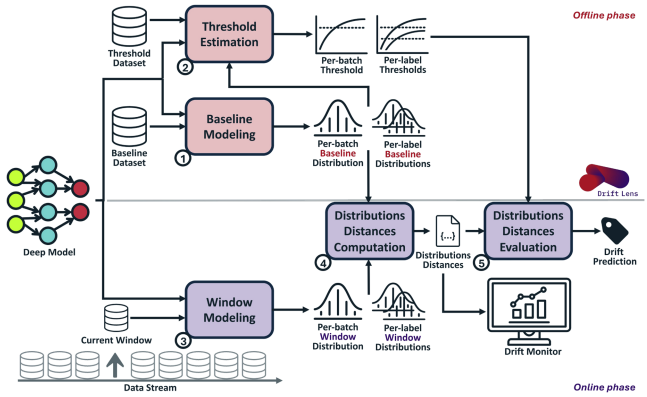


Fig. 2. DRIFTLENS Framework. In the *offline* phase, it estimates the reference distributions and distance thresholds from historical (training) data. Distributions are modeled as multivariate normal and are computed: (i) for the entire batch (*per-batch*), and (ii) conditioned on the predicted label (*per-label*). In the *online* phase, it analyzes data streams in fixed windows, comparing new and reference distributions, and using thresholds to identify drift, visualized in a drift monitor.

Drift Adaptation and Incremental Learning: Some works are related to incremental learning for drift detection and adaptation [59]. However, in unsupervised settings, the adaptation to drift is challenging due to the lack of annotated samples [4]. Drift adaptation is out of the scope of this paper. Instead, we focus only on the drift detection problem (*monitoring*).

This paper presents DRIFTLENS, an unsupervised statistical-based drift detection technique. It significantly extends the preliminary idea in [60] in two key aspects: (1) *Methodology:* It enhances the original approach by complementing the *per-label* with a *per-batch* analysis, improving its general applicability across tasks. Additionally, it redefines the threshold estimation and provides drift explanations to better characterize drift; (2) *Evaluation:* It demonstrates its general applicability and effectiveness through a comprehensive evaluation across data types, models, and baseline detectors. DRIFTLENS is also available in a tool with a user-friendly graphical interface [61].

DRIFTLENS differs from previous work in three key aspects. (1) Like other statistical-based methods, it is completely *unsupervised*. Thus, it does not require any external model to detect drift and is *data type agnostic*, unlike supervised, loss-based, and virtual classifier methods. (2) It exploits statistical distances that better scale with data dimensionality than other statistical-based methods, thereby enabling efficient *real-time* drift detection in large data volumes. (3) It *characterizes* drift by assessing its impact on each label and providing prototype-based *explanations* to improve human understanding of drift.

IV. DRIFTLENS

DRIFTLENS is an *unsupervised* drift detection technique based on distribution distances within embeddings—internal dense representations generated by deep learning models. Designed specifically for unstructured data, it captures subtle patterns and relationships in embeddings to identify data drifts over time. It operates in an *unsupervised* way as it does not require true labels in the data stream. However, it is tailored to detect concept drift on supervised models, such as classifiers.

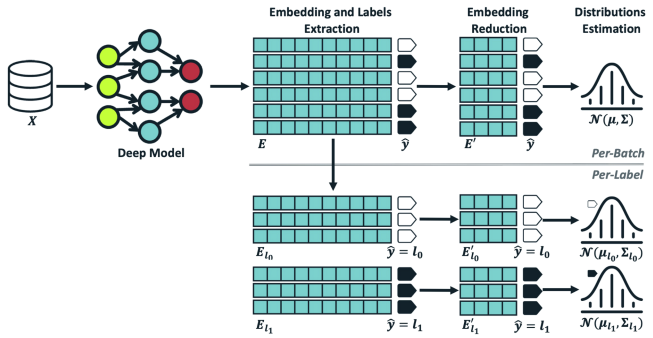


Fig. 3. DRIFTLENS Data Modeling. Given a deep learning model and a set of data X, it estimates the multivariate normal embedding distributions. It first extracts data embeddings and predicted labels from the model and reduces the embedding dimensionality. It then estimates (i) the *per-batch* distribution by computing the mean vector μ and covariance matrix Σ over all embeddings, and (ii) label-specific distributions by grouping embeddings by label (e.g., l_0 and l_1), reducing the dimensionality, and computing the *per-label* μ_l and Σ_l, $\forall l \in L$.

DRIFTLENS consists of an *offline* and an *online* phase (Fig. 2). In the *offline* phase, it estimates the *reference distributions* and *threshold values* from the historical dataset (e.g., training data). These distributions, called *baseline*, represent the distribution of features (embeddings) of the concepts learned by the model during training, thus representing the absence of drift. In the *online* phase, it processes the new data stream in fixed-size windows. First, the distributions of the new data windows are estimated. Second, the distribution distances are computed with respect to the reference distributions. If the distance exceeds the threshold, a drift is predicted.

We first detail the data modeling (Section IV-A) and the distribution distance (Section IV-B), performed similarly in both phases. We then describe the offline (Section IV-C) and online (Section IV-D) phases. Finally, we discuss drift explanations and adaptation (Section IV-E).

A. Data Modeling: Embedding Distributions Estimation

Consider a deep learning classifier designed to distinguish between a set of class labels L. The classifier typically consists of an encoder $\phi(X)$ that transforms sparse, complex, and high-dimensional raw input data into a dense, lower-dimensional latent representation (embedding) through a series of nonlinear transformations parameterized by the learned weights W. The embedding representation is then processed by one or more fully connected layers to predict the final class label.

To simplify data distribution modeling, DRIFTLENS exploits the internal representations (embeddings) generated by the deep learning model instead of raw inputs. This reduces noise and complexity, especially for unstructured data types like images, text, and audio, which are often complex, noisy, and non-Gaussian. In contrast, embeddings are lower-dimensional and capture more structured information.

As shown in Fig. 3, DRIFTLENS estimates two types of embedding distributions: (1) the overall distribution of the entire data (*per-batch*), and (2) $|L|$ separate distributions, one for each predicted class (*per-label*). It assumes that embeddings follow multivariate normal distributions, making Gaussianity a more plausible approximation given the simpler relationships in embedding space [62]. This approach is data type-independent, ensuring broad applicability in diverse domains.

Formally, during data modeling, a batch of data X (historical data or a data stream window) is fed into the deep learning model to extract the embedding matrix $E = \phi(X) \in \mathbb{R}^{m \times d}$ and predicted labels $\hat{y} \in \mathbb{R}^m$, where m is the number of samples in the set and d is the dimensionality of the embedding layer. Embeddings are also grouped by predicted label, resulting in $|L|$ matrices $E_l \in \mathbb{R}^{m_l \times d}, \forall l \in L$, where m_l is the number of samples predicted with label l.

In deep learning classifiers, the embedding space dimensionality d is usually large. To estimate the multivariate normal distributions, we need to compute the mean vector μ and the covariance matrix Σ of the embedding vectors. However, the covariance matrix requires at least d linearly independent vectors to be full rank. Otherwise, it may contain complex numbers that could affect the distribution distance computation (see Section IV-B). This issue is more pronounced when estimating the multivariate normal distribution for each label, as we need d linearly independent vectors predicted with each label. This is usually not an issue when modeling the baseline, which is computed over a usually large historical dataset. In contrast, in the online phase, estimating the per-batch distribution requires d while the per-label distributions $d \times |L|$ linearly independent vectors in each data stream window. Therefore, the fixed size used to divide the data stream into windows must be very large. To solve this, DRIFTLENS performs an embedding dimensionality reduction applying a principal component analysis (PCA).

$$E' = PCA(E); \quad E'_l = PCA(E_l), \forall l \in L \tag{3}$$

This leads to the reduced embedding matrices $E' \in \mathbb{R}^{m \times d'}$ and $E'_l \in \mathbb{R}^{m_l \times d'_l}, \forall l \in L$. d' and d'_l are user-defined parameters that determine the number of principal components for the per-batch and per-label reductions, with $0 < d', d'_l \leq d$. For the per-batch, d' can be set to a value up to $d' = \min(m_w, d)$, where m_w is the window size used in the online phase and d is the embedding dimensionality. Instead, d'_l also depends on the number of labels. A reasonable value for a balanced data stream is close to $d'_l = \min(m_w/|L|, d)$.

Finally, the reduced embedding matrices E' and E'_l are used to estimate the *per-batch* and the $|L|$ *per-label* multivariate normal distributions (4) and (5).

$$P(\phi(x); W) \sim \mathcal{N}(\mu, \Sigma) \tag{4}$$

$$P(\phi(x) \mid \hat{y} = l; W) \sim \mathcal{N}(\mu_l, \Sigma_l), \forall l \in L \tag{5}$$

Where $\mu \in \mathbb{R}^{d'}$ and $\Sigma \in \mathbb{R}^{d' \times d'}$ are the mean vector and the covariance matrix of the *per-batch* multivariate normal distribution, and each $\mu_l \in \mathbb{R}^{d'_l}$ and $\Sigma_l \in \mathbb{R}^{d'_l \times d'_l}$ represents the *per-label* multivariate normal distribution of each label $l \in L$, separately. The multivariate normal distributions are straightforward and fast to estimate because they can be fully characterized by the mean vector and the covariance matrix. The main advantage of estimating the distributions as multivariate normal is that they can be represented with dimensionality d' and d'_l,

regardless of the number of samples in the reference and new windows. This allows the method to scale well even with large amounts of data, enabling drift detection in real-time regardless of the data volumes.

B. Distributions Distance Computation: Frechét Distance

DRIFTLENS uses the Frechét distance [63] to calculate the distance between two multivariate normal distributions. The Frechét distance, also known as the Wasserstein-2 distance [64], has been widely used in deep learning to measure the distances between the distributions of models' features but in very different scenarios [65]. DRIFTLENS uses the Frechét distance to measure the distances between the embedding distributions of a baseline (historical) and the new windows in the data stream, and we call it *Frechét Drift Distance* (FDD).

Given a multivariate reference normal distribution b (e.g., the baseline computed in the *offline* phase) characterized by a mean vector μ_b and a covariance matrix Σ_b, and the multivariate normal distribution of a new data window w, characterized by μ_w and Σ_w, the FDD is calculated as:

$$FDD(b,w) = ||\mu_b - \mu_w||_2^2 + Tr\left(\Sigma_b + \Sigma_w - 2\sqrt{\Sigma_b\Sigma_w}\right) \tag{6}$$

$FDD \in [0, \infty]$. The higher the FDD, the greater the distance and the more likely the drift. It captures changes in the mean (center) and diagonal elements of the covariance (spread) between two distributions. The former results from the L2 norm of the difference between the mean vectors $||\mu_b - \mu_w||_2^2$. The latter is computed by $Tr(\Sigma_b + \Sigma_w - 2\sqrt{\Sigma_b\Sigma_w})$, which is a generalization of the squared difference between the standard deviations in a one-dimensional space. As a result, FDD can detect subtle drifts that affect both the center and spread of distributions. Although DRIFTLENS uses the FDD as default metric, it allows flexibility in its choice [66], [67], [68], [69].

DRIFTLENS computes a single *per-batch* distance by computing the FDD score between the baseline (μ_b, Σ_b) and the new window (μ_w, Σ_w) *per-batch* distributions, and $|L|$ *per-label* distances by computing the distance of the distributions ($\mu_{b,l}, \Sigma_{b,l}$) and ($\mu_{w,l}, \Sigma_{w,l}$) for each label $l \in L$ separately.

C. Offline Phase: Baseline and Threshold Estimation

In the *offline* phase (steps ①–② in Fig. 2), DRIFTLENS estimates: ① the probability distribution of a reference (historical) dataset representing the concepts learned by the model during training (*baseline*), and ② distance *thresholds* to distinguish between normal and drift-indicative distances, which will be used during the *online* phase. This phase is executed once to instantiate DRIFTLENS, after the monitored model has been trained, evaluated, and is ready for deployment.

Baseline Estimation: It estimates the reference distributions (distributions in the absence of drift) by performing the data modeling (Section IV-A) of the baseline (historical) dataset.

Formally, given a baseline dataset X_b, containing m_b samples, the entire baseline dataset X_b is fed into the deep learning model to extract the embedding vectors $E_b = \phi(X_b) \in \mathbb{R}^{m_b \times d}$ and estimate the predicted labels $\hat{y}_b$. Then, the *per-batch* PCA is fitted over the entire set of vectors, and $|L|$ different PCAs are fitted, grouping the embedding vectors according to the predicted labels. The embedding vectors are then reduced both for the *per-batch* and *per-label*, to obtain the reduced embedding matrices $E'_b \in \mathbb{R}^{m_b \times d'}$, and $E'_{b,l} \in \mathbb{R}^{m_{b,l} \times d'_l}, \forall l \in L$. These matrices are used to estimate the baseline *per-batch* and *per-label* multivariate normal distributions. The *per-batch* distribution is fully characterized by the baseline *per-batch* mean vector $\mu_b \in \mathbb{R}^{d'}$ and the covariance matrix $\Sigma_b \in \mathbb{R}^{d' \times d'}$, which are obtained by calculating the mean and covariance over the entire set of reduced embedding vectors in the baseline dataset. The *per-label* distributions are obtained by computing the $|L|$ mean vectors $\mu_{b,l} \in \mathbb{R}^{d'_l}$ and the covariance matrices $\Sigma_{b,l} \in \mathbb{R}^{d'_l \times d'_l}$ on the reduced embedding vectors, grouped by predicted labels. Note that regardless of the dimensionality of the reference set (baseline) m_b, the estimated reference distributions are characterized by vectors of size d' and d'_l. Thus, drift detection in new windows is not influenced by m_b.

Threshold Estimation: It estimates the maximum possible distance (FDD) a window without drift can reach. It takes in input a threshold dataset X_{th}, the window size m_w (equal to the one used in the online phase), the baseline, and a parameter n_{th} defining the number of windows randomly sampled from the threshold dataset. n_{th} should be large to better estimate the maximum distance considered without drift. In our experiments (Section V-A), we set $n_{th} = 10{,}000$, and show that its variations have a small impact (see Appendix C, available online).

Specifically, DRIFTLENS randomly samples, with replacement, n_{th} windows from the threshold dataset X_{th}, each containing m_w inputs. For each window, it performs the data modeling and computes the *per-batch* and *per-label* distribution distances from the baseline distributions (6). Therefore, n_{th} distribution distances for the entire batch and each label are computed. Finally, the distribution distances are sorted in descending order. The first element contains the maximum distance that a window of data considered without drift can achieve. Distances that exceed this value are potential warnings of drift. However, there are potentially outlier distances due to the large number of randomly sampled windows. Therefore, DRIFTLENS provides a parameter to define the threshold sensitivity $T_\alpha \in [0, 1]$. This parameter removes the $T_\alpha\%$ left tail of the sorted distances (in descending order) to remove outliers. The final thresholds T and $T_l, \forall l \in L$, are set to the maximum distance after removing the $T_\alpha\%$ of the largest distances. The higher the value of T_α, the lower the threshold values and the higher the sensitivity to possible drift and false alarm. In our experiments (see Section V-A), we use $T_\alpha = 0.01$ (removing 1%) as the default threshold sensitivity.

Choice of the Reference Dataset: The reference dataset must consist of *historical* data representing what the model learned during training. Since we operate after training, we assume the availability of historical data used for training and testing, typically split into train/val/test or train/test sets. Training data can be used as the baseline to represent the concepts learned by the model, modeling the absence of drift. For the threshold

estimation—maximum distribution distance without compromising model performance—a reasonable choice is the data used for evaluation (e.g., test or validation sets, or their combination). Alternatively, windows from the data stream can be used. Due to the absence of true labels, human experts must ensure that these windows represent distributions without drift.

D. Online Phase: Drift Detection Over Time

The *online* phase (steps ③–⑤ in Fig. 2) continuously detects drift in the monitored model when deployed and applied to a data stream. DRIFTLENS splits the data stream into fixed-size windows, whose size is defined by the parameter m_w, and applies the same process to each window.

Drift Detection in a Window: Given a new data window X_w containing m_w samples, ③ the data modeling is processed by (i) extracting the embedding $E_w = \phi(X_w) \in \mathbb{R}^{m_w \times d}$ and predicted labels $\hat{y}_w \in \mathbb{R}^{m_w}$, (ii) performing the embedding dimensionality reduction to obtain $E'_w \in \mathbb{R}^{m_w \times d'}$ and $E'_{w,l} \in \mathbb{R}^{m_{w,l} \times d'_l}, \forall l \in L$, and (iii) estimating the *per-batch* and *per-label* multivariate normal distributions. The PCA models fitted during the offline phase are reused in this step. The *per-batch* multivariate normal distribution is obtained by computing the mean vector $\mu_w \in \mathbb{R}^{d'}$ and the covariance matrix $\Sigma_w \in \mathbb{R}^{d' \times d'}$ over the entire set of reduced embeddings in the window. For the *per-label*, the reduced embedding vectors are grouped by predicted labels, and the mean and covariance are computed separately on each subset, resulting in $|L|$ multivariate normal distributions characterized by $\mu_{w,l} \in \mathbb{R}^{d'_l}$ and $\Sigma_{w,l} \in \mathbb{R}^{d'_l \times d'_l}$. Next, ④ the *per-batch* and the *per-label* FDD distances between the window and baseline distributions are computed using (6). If these distances exceed the thresholds, drift is predicted ⑤. The *per-batch* distance detects if the entire window is affected by drift, while the *per-label* distance characterizes drift and identifies the labels most affected.

Drift Monitoring Over Time: Once DRIFTLENS processes a new window, the FDD distribution distances are added to the drift monitor, which shows them separately for the *per-batch* and *per-label*. A warning symbol is added to the plot when distances exceed the corresponding threshold. The drift monitor provides valuable insights to understand (i) *when*, *whether* drift occurs, and its *severity*, (ii) *which labels* are the most affected by drift, and (iii) the drift *pattern* over time.

Fig. 4 shows a drift monitor example. The top chart shows the *per-batch*, whereas the bottom chart shows the *per-label* distribution distances over time. The *x-axis* reports timestamps and window identifiers. The *y-axis* shows the FDD distances. When a drift is detected, the area under the curve is filled, and a warning is displayed on the x-tick. This monitor reveals that drift first occurs after 50 windows with high severity and reoccurs periodically. The label *World* is most affected, followed by *Business*, while *Sports* is minimally affected.

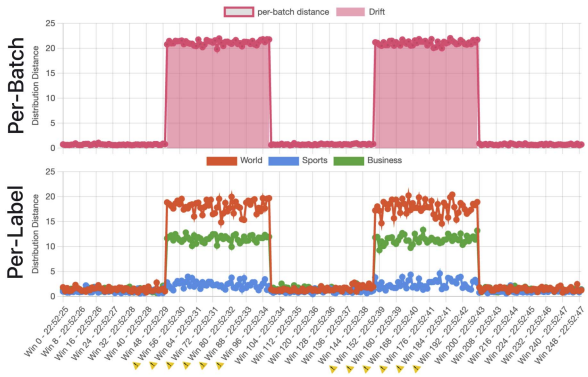


Fig. 4. Drift monitor example.

E. Drift Explanation, Characterization, and Adaptation

Drift Explanation: DRIFTLENS produces drift explanations to enhance human understanding of drift. Although the objective is related to XAI [54], [55], it differs in focus. XAI aims to explain classifier predictions, whereas drift explainability aims to explain the underlying causes of drift [48]. Since DRIFTLENS operates on classifiers for high-dimensional, high-volume unstructured data streams, without true labels, the explanation method must meet these requirements. (1) *Post-hoc:* It should apply to models already trained, without modifying the training process. (2) *Model-agnostic:* It should work with any model, regardless of its architecture. (3) *Unsupervised:* It should not rely on true labels or predefined concepts. (4) *Data type-independent:* It should work uniformly across data types.

To meet these requirements, DRIFTLENS produces example-based explanations [55], which are intuitive to humans [70], [71], including to understand drift [48]. Drift explanations are generated on request for one or more drifted windows rather than continuously. Drift explanations identify representative examples (*prototypes*) of inputs processed by the model to illustrate the *"concepts"* associated with each label and a batch of data. In the absence of drift, we expect these prototypes to closely resemble those in the historical data. In contrast, when drift is present, the drifting samples introduce new concepts specific to a label or the entire batch.

Given one or multiple concatenated windows, DRIFTLENS first exploits a clustering algorithm to group samples by embedding representations, considering the entire batch and each label separately. The objective of clustering is to group samples to isolate drifted samples into one or more pure clusters. To this end, it executes the K-Means algorithm multiple times, for $k \in [2, K_{\max}]$ on the embedding matrices E and $E_l, \forall l \in L$, where $K_{\max}$ is a user-defined parameter. The optimal clustering partition—the best k—is determined by maximizing the Silhouette metric, and may differ between the per-batch and each label. For a given label or batch, the K-Means produces k clusters, each representing a distinct concept relevant to classification. The samples nearest to the centroids of these clusters serve as representative examples (*prototypes*), effectively summarizing the primary characteristics of the group. To improve prototype interpretability, a user-defined parameter, top-n, is introduced, specifying the n closest samples to each centroid based on the euclidean distance of the embeddings. This enhances interpretability by allowing users to observe multiple representative samples per cluster. Examining a single example may not be sufficient to

understand the common features, whereas visualizing multiple similar instances provides more comprehensive insight into the shared characteristics within each cluster. Prototypes provide valuable insight to interpret the causes of drift (see Section V-E).

Drift Characterization and Adaptation: In unsupervised data streams, the absence of annotated samples makes automatic drift adaptation challenging [4]. As a result, as with most unsupervised detectors, DRIFTLENS does not automatically adapt the model when drift is detected. Instead, human-in-the-loop is required for adaptation, necessitating the annotation of new data, or a subset, by domain experts (or an oracle). However, drift characterization enables experts to gain deeper insight into drift and take informed adaptive measures.

1) The *per-label drift analysis* helps users focus on the labels most affected by drift, simplifying both the understanding and annotation process. By narrowing the scope of analysis, it reduces complexity and enables users to direct their efforts where they are most needed. This enhances the interpretability of drift patterns and reduces the annotation workload, making it easier to adapt models to evolving data distributions.

2) The *prototype-based drift explanations* help users better understand the nature of drift and how it affects each label and the entire batch. These prototype examples can reveal whether a new class label should be introduced due to the emergence of concepts that differ significantly from those in the training data. Alternatively, they may suggest that the meaning of an existing label is evolving over time, indicating the need for further training to maintain accuracy.

3) The *drift monitor over time* can reveal whether the drift is stabilizing, progressively increasing, or recurring. (i) If drift occurs *suddenly and remains constant*, it indicates that the deployed model is outdated, with new distributions consistently replacing the previous ones. Therefore, the model should be updated or replaced by fine-tuning it on the new distributions or retraining it from scratch, incorporating historical and new data. (ii) If drift occurs with a *periodic* pattern, two strategies are possible. The model can be retrained to perform well across both distributions (similar to sudden drift), or an additional model can be trained and deployed alongside the original. When drift occurs, the outdated model can be swapped out for the one best suited to the new distribution. Once data return to their normal state, the system can switch back to the original model. (iii) If drift increases *incrementally*, the model can be gradually fine-tuned to the new distributions, or retraining can be postponed until drift stabilizes.

Once the model is adapted (or retrained) to the new incoming distributions, DRIFTLENS can be updated by simply re-executing the offline phase to reset the baseline and threshold.

V. EVALUATION

We assess DRIFTLENS' effectiveness in detecting drift across text, image, and audio classifiers (introduced in Section V-A) by evaluating drift detection performance (Section V-B) and execution time (Section V-C). We also examine its ability to characterize drift trends over time (Section V-D) and provide explanations (Section V-E). Parameter sensitivity is analyzed in Appendix C, available online, and a distribution distance ablation study is conducted in Appendix D, available online.

A. Experimental Settings

The 17 experimental use cases are summarized in Table I.

Deep learning Classifiers: We experiment with BERT [80], DistilBERT [81], and RoBERTa [82] deep learning models for NLP classification, VGG16 [83] and Vision Transformer (ViT) [84] for computer vision, and Wav2Vec [85] for audio.

Datasets: We train NLP models for topic detection using Ag News [72] and 20 Newsgroups [73], and for occupation classification with Bias in Bios [74] datasets. In computer vision, we use MNIST [75], Intel Image [76], STL-10 [77], and FairFace [78] for image classification. In the audio domain, we use Common Voice [79] for speaker gender classification.

Embedding Extraction: For BERT, DistilBERT, RoBERTa, and ViT models, we extract the embedding of the [CLS] token from the last hidden layer (dimensionality $d=768$). For VGG16, we extract and flatten the last convolutional layer, resulting in $d=512$, $d=8{,}192$, and $d=4{,}608$ for use cases 5.2, 6.2, and 7.2, respectively. For Wav2Vec, we extract and average all the embeddings of the last hidden layer ($d = 768$).

Data Split: Each dataset has been split into four non-overlapping subsets, as described in the last column of Table I. Two subsets are used to fine-tune and evaluate the classifiers (train and test splits). These two subsets represent *historical* data that can be used by drift detectors as reference data. The third subset is reserved for generating windows in the *data stream*, simulating new, unseen data with a similar distribution to the training set (without drift). The fourth subset is designated for simulating drift. Together, these last two subsets form the data stream for drift detection experiments. Importantly, the historical and data stream subsets are disjoint.

Drift Simulation: We simulate different drift sources, as described in Table I. In use cases 1-3 and 5-7, drift is simulated by introducing a new unknown class label, obtained by removing samples from a label during training and by presenting them in the data stream. For instance, in use case 1, all $31{,}900$ samples for the class *Science/Tech* are used to simulate drift. The remaining samples from classes *World*, *Business*, and *Sport* are divided into $59{,}480$ and $5{,}700$ to fine-tune and evaluate the model (historical data), while $30{,}520$ to simulate data stream samples without drift. In use case 3, the same dataset is used as in use case 2, but drift is more subtle, as only a subset of a class is used to simulate the drift by exploiting the hierarchical categorization of labels. In use case 8, we simulate data drift by altering the input features through image blurring. Gaussian blur is applied with a radius of 2 to a circular patch covering $D_{\%}$ (drift percentage) of pixels. In use cases 4, 9, and 10, drift is simulated with biased classifiers trained on underrepresented protected features. Drift occurs by presenting samples from subgroups with the same labels but different features, exposing the classifier's bias. In use case 4, we build upon previous research showing that occupation classifiers can exhibit gender bias [86], [87], and we trained a classifier on training bios where all nurses were female while

TABLE I
OVERVIEW OF THE EXPERIMENTAL USE CASES

Data Type	Dataset	Task	Use Case	Models	F1	Description
Text	Ag News [72]	Topic Detection	1.1 1.2* 1.3*	BERT DistilBERT RoBERTa	0.98 0.97 0.98	**Training labels**: *World*, *Business*, and *Sport* **Drift**: Simulated with one new class label: *Science/Technology* **Data split**: HISTORICAL {59,480 train; 5,700 test} - DATA STREAM {30,520 without drift; 31,900 drifted}
Text	20 Newsgroups [73]	Topic Detection	2.1 2.2* 2.3*	BERT DistilBERT RoBERTa	0.88 0.87 0.88	**Training labels**: *Technology*, *Sale-Ads*, *Politics*, *Religion*, *Science* (5 macro-labels) **Drift**: Simulated with one new macro-label *Recreation*: composed of *baseball*, *hockey*, *motorcycles*, *autos* subtopics **Data split**: HISTORICAL {5,080 train; 3,387 test} - DATA STREAM {5,560 without drift; 3,655 drifted}
Text	20 Newsgroups [73]	Topic Detection	3	BERT	0.87	**Training labels**: Training on 6 labels. The 5 macro-labels: *Technology*, *Sale-Ads*, *Politics*, *Religion*, *Science*, and a subset of the macro-label *Recreation*, composed of subtopics: *baseball* and *hockey* **Drift**: Simulated with another subset of the macro-label *Recreation*: *motorcycles* and *autos* subtopics **Data split**: HISTORICAL {5,744 train; 3,829 test} - DATA STREAM {6,304 without drift; 1,805 drifted}
Text	Bias in Bios [74]	Occupation Classification	4	BERT	0.94	**Training labels**: *Professor*, *Physician*, *Attorney*, *Photographer*, *Journalist* (male gender); *Nurse* (female gender) **Drift**: Simulated with bios from same labels but opposite gender (male nurses and female for all others occupations) **Data split**: HISTORICAL {39,270 train; 15,108 test} - DATA STREAM {94,184 without drift; 106,242 drifted}
Image	MNIST [75]	Image Classification	5.1 5.2*	ViT VGG16	0.98 0.95	**Training labels**: Digits from 0 to 7 **Drift**: Simulated with two new digits: Digits 8 and 9 **Data split**: HISTORICAL {35,000 train; 8,000 test} - DATA STREAM {12,000 without drift; 13,000 drifted}
Image	Intel Image [76]	Image Classification	6.1 6.2*	ViT VGG16	0.90 0.89	**Training labels**: *Forest*, *Glacier*, *Mountain*, *Building*, *Street* **Drift**: Simulated with one new class label: *Sea* **Data split**: HISTORICAL {6,000 train; 4,000 test} - DATA STREAM {4,256 without drift; 2,780 drifted}
Image	STL-10 [77]	Image Classification	7.1 7.2*	ViT VGG16	0.96 0.82	**Training labels**: *Airplane*, *Bird*, *Car*, *Cat*, *Deer*, *Dog*, *Horse*, *Monkey*, *Ship* **Drift**: Simulated with one new class label: *Truck* **Data split**: HISTORICAL {5,850 train; 2,925 test} - DATA STREAM {2,925 without drift; 1,300 drifted}
Image	STL-10 [77]	Image Classification	8	ViT	0.90	**Training labels**: *Airplane*, *Bird*, *Car*, *Cat*, *Deer*, *Dog*, *Horse*, *Monkey*, *Ship*, *Truck* **Drift**: Simulated by introducing *blur* in the images within the same labels **Data split**: HISTORICAL {6,500 train; 3,250 test} - DATA STREAM {3,250 without drift; 3,250 drifted}
Image	FairFace [78]	Gender Identification	9	ViT	0.95	**Training labels**: *Male*, *Female* - (White, Black, Middle Eastern, Latino-Hispanic, Indian races) **Drift**: Introduced with images from same labels but different race (East Asian and Southeast Asian races) **Data split**: HISTORICAL {27,984 train; 7,313 test} - DATA STREAM {30,000 without drift; 32,401 drifted}
Audio	Common Voice [79]	Gender Identification	10	Wav2Vec	0.91	**Training labels**: *Male*, *Female* - (US and UK accent) **Drift**: Introduced with audios from same labels but different accent (Australian, Canadian, Scottish) **Data split**: HISTORICAL {70,578 train; 9,951 test} - DATA STREAM {29,556 without drift; 42,697 drifted}

Use cases are partitioned into groups based on the dataset and task. The training labels, the way the drift is simulated, and the split of the dataset are given in the description for each group. Within each group, different deep learning models are considered, and the corresponding F1 scores obtained on the test set are given.

individuals in other occupations were male. In the data stream, drift is introduced by mixing non-drift with drifted samples, where nurses are male and individuals in other occupations are female. In use cases 9 and 10, drift is simulated as racial bias by presenting images from different races or audio with English accents other than those used during training.

Windows Generation: When generating windows of the data stream, we use samples—either with or without drift—that the model has never encountered during training or testing, whose dimensionalities are specified in the last column of Table I (DATA STREAM). Windows are sampled with replacement due to dimensionality constraints (samples may belong to more windows). When generating a window without drift, we randomly select a balanced sample by class labels from the new data without drift. These samples are previously unseen by the model but share the same distribution as the training data. When generating windows containing $D_{\%}$ of drift, we randomly select $D_{\%}$ of the window size (m_w) from the drifted samples, while the remaining $(100 - D)_{\%}$ are balanced samples from the new unseen data without drift. Use case 8 is the only exception where a $D_{\%}$ of pixels over all images are blurred.

DriftLens Configuration: In the *offline* phase, the entire historical training and test sets are used for baseline and threshold estimation, respectively, with sizes shown in the last column of Table I (HISTORICAL). The number of principal components for *per-batch* and *per-label* embedding are $d' = 150$ and $d'_l = 75$, except for use case 10, where $d' = d'_l = 25$ ensures full-rank matrices. The number of windows for threshold estimation is $n_{th} = 10{,}000$, with sensitivity $T_\alpha = 0.01$.

Baseline Detectors: We compare DRIFTLENS with four unsupervised statistical-based detectors from previous work: Maximum Mean Discrepancy (MMD) [28], [37], Kolmogorov-Smirnov (KS) [29], [37], Least-Squares Density Difference (LSDD) [30], and Cramér-von Mises (CVM) [39]. We use the implementation provided by the Alibi Detect library [88]. We keep the default parameters configuration, and we use embedding vectors as input. The *p-value* to discriminate drifted and non-drifted distributions is set to 0.05 as the default. However, as we will discuss in Section V-C, the running time of some detectors is significantly influenced by the reference set dimensionality m_b. Given our focus on real-time drift detection, and to ensure a fair comparison, we constrain the reference window dimensionality. Specifically, we limit the execution time for each detector to predict drift in a window to a maximum of 30 seconds, estimated using $m_w = 1{,}000$ and $d = 1{,}000$.[1] To this end, we set the maximum reference window size $m_b \approx 14{,}000$ for MMD and $m_b \approx 8{,}500$ for LSDD. In contrast, we used the full training set for KS and CVM due to their faster execution time. When the training set size exceeds these constraints, the reference set is created by randomly sampling a balanced subset equal to the maximum size. The only exception is the 20 Newsgroups dataset, where we used $m_b \approx 1{,}700$ for use case 2 and $m_b \approx 2{,}050$ for use case 3 across all detectors to ensure a balanced reference set.

[1]Executed on an Apple M1 MacBook Pro 13 2020 with 16GB of RAM.

B. Drift Detection Performance Evaluation

This evaluation assesses the effectiveness and general applicability of DRIFTLENS in detecting drifted windows of varying severity across models, data types, and tasks (RQ1 and RQ2 in Section I). We approach drift detection as a binary classification task—predict whether a window contains drift.

Evaluation Metrics: We measure the *accuracy* in predicting drift ($A_{D_\%}$) with different degrees of severity $D_\% \in \{0\%, 5\%, 10\%, 15\%, 20\%\}$. Windows without drift ($D_\% = 0\%$) are used to measure type I errors (false alarm)—no drift, but has been detected. If a window contains any percentage of drifted examples, the ground truth is set to 1; otherwise, it is set to 0. For each $D_\%$, the accuracy is computed as the mean over 5 independent runs of the accuracy computed on 100 windows with a fixed size of m_w.[2] For instance, $A_{20\%}$ is computed by averaging five accuracies, each computed over 100 windows containing m_w samples of which 20% are drifted. However, since the predictions of drifted and non-drifted windows are closely related, a technique that always predicts drift is unreliable. Therefore, we compute the *Harmonic Drift Detection* (H_{DD}), as the harmonic mean between the accuracy for non-drifted windows ($A_{0\%}$) and the average accuracy across windows with different drift severities ($\bar{A}_{\text{drift}}$):

$$H_{DD} = \frac{2}{\frac{1}{A_{0\%}} + \frac{1}{\bar{A}_{\text{drift}}}} \tag{7}$$

Where:

$$\bar{A}_{\text{drift}} = \frac{A_{5\%} + A_{10\%} + A_{15\%} + A_{20\%}}{4} \tag{8}$$

Results: Table II reports: (i) the mean *accuracy* broken down by drift percentage ($D_\%$), and (ii) the H_{DD} score, for each use case, drift detector, and window size (m_w). Results within the same use case but with a different classifier (marked with $*$ in Table I) are reported in Appendix A, available online. Table II(a) uses larger data windows than Table II(b) as datasets contain more samples.

For *larger* data volumes (Table II(a)), DRIFTLENS achieves better drift prediction performance on all use cases independently of the data type and window size, except for use cases 9 with $m_w = 500$ and 4 with $m_w = 2000$. For use cases 1.1 and 4, it is particularly effective in detecting drift, with $H_{DD} \geq 0.93$. DRIFTLENS is the only detector that consistently achieves reliable results across all use cases ($H_{DD} \geq 0.38$) and that performs reliably in the audio domain (use case 10). Notably, it is capable of detecting drift and outperforming other detectors, even when drift manifests as classifier bias (use cases 4, 9, 10).

In these use cases with larger datasets and window sizes, DRIFTLENS achieves an average H_{DD} score of 0.82, outperforming MMD (0.57), KS (0.56), LSDD (0.57), and CVM (0.55). Therefore, it significantly outperforms the best detector by an average margin of 0.25 in H_{DD}. Results are consistent across deep learning classifiers (Table Va in Appendix A, available online). Overall, DRIFTLENS is the best detector on 8 out of 8 experimental use cases with larger datasets and window sizes.

For *smaller* data volume (Table II(b)), DRIFTLENS is still the most effective overall, except for use case 8, in which MDD and LSDD achieve slightly better H_{DD}. Interestingly, all detectors detect drift simulated with blur (use case 8). However, KS and CVM exhibit a large number of false positives, especially for larger window sizes. Notably, DRIFTLENS ($H_{DD} \geq 0.81$) and LSDD ($H_{DD} \geq 0.17$) are the only detectors consistently achieving reliable performance across all use cases and window sizes. In contrast, the other detectors exhibit some use cases with unreliable performance ($H_{DD} = 0$).

In these use cases with smaller datasets and window sizes, DRIFTLENS achieves an average H_{DD} score of 0.92, outperforming MMD (0.69), KS (0.42), LSDD (0.66), and CVM (0.40). Therefore, it significantly outperforms the best detector by 0.23 in H_{DD}. Results are consistent across deep learning classifiers (Table Vb in Appendix A, available online). Overall, DRIFTLENS is the best detector on 7 out of 9 experimental use cases with smaller datasets and window sizes.

Summary of findings: DRIFTLENS is highly effective in detecting drift (RQ1 in Section I), outperforming other detectors in 15/17 use cases. It is the only reliable method across all use cases, making it the most effective and widely applicable across models, data types, and data volumes (RQ2 in Section I).

C. Drift Detection Complexity Evaluation

This evaluation assesses the efficiency of DRIFTLENS in performing near real-time drift detection (RQ3 in Section I).

Evaluation Metrics: We measure the mean running time in seconds required to predict drift, given the embeddings extracted. Experiments are executed on an Apple M1 MacBook Pro 13 2020 with 16 GB of RAM. Due to the high dimensionalities tested, we use synthetically generated embeddings.

Results: Fig. 5 shows the running time in seconds on a logarithmic scale of each detector by varying (a) the reference window size m_b, (b) the data stream window size m_w, and (c) the embedding dimensionality d. One dimension at a time is varied while keeping the other fixed at: $m_w = 1{,}000$, $d = 1{,}000$, and $m_b = 5{,}000$. DRIFTLENS outperforms all detectors in terms of efficiency, running at least 5 times faster. As the dimensionalities increase, its execution time increases negligibly, unlike other techniques that are significantly affected by the window sizes and embedding dimensionality.

Fig. 6 shows DRIFTLENS' running time when dealing with large data volumes. The reference window is increased up to $500{,}000$ samples, and the data stream window to $10,000$. The other detectors do not work with these dimensionalities on the experimental hardware. When varying the reference window size m_b, the data stream window size is kept fixed to $m_w = 5{,}000$. When varying the data stream window size m_w, the reference window size is fixed to $m_b = 500{,}000$. Fig. 6(a) confirms that its running time is almost independent of the reference window size, as in the online phase, only the pre-computed reference distributions are used (μ_b, $\mu_{b,l}$ and Σ_b, $\Sigma_{b,l}$ with dimensionalities d', d'_l). Fig. 6(b) shows that its running time increases almost

[2] At each run the threshold of DRIFTLENS is re-estimated and the reference set of the other detectors is re-sampled (if exceeds the maximum size).

TABLE II
DRIFT DETECTION PERFORMANCE EVALUATION

(a) **Larger data volumes:** Data stream window sizes $m_w \in \{500, 1000, 2000\}$

Use Case	Drift Detector	$m_w = 500$						$m_w = 1000$						$m_w = 2000$					
		Drift Percentage $D_\%$						Drift Percentage $D_\%$						Drift Percentage $D_\%$					
		0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}
1.1 Ag News BERT	MMD	1.00	0.00	0.10	0.96	1.00	*0.68*	1.00	0.00	0.83	1.00	1.00	*0.83*	1.00	0.06	1.00	1.00	1.00	*0.87*
	KS	1.00	0.01	0.21	0.98	1.00	*0.71*	0.99	0.03	0.95	1.00	1.00	*0.85*	1.00	0.38	1.00	1.00	1.00	*0.91*
	LSDD	1.00	0.00	0.13	0.93	1.00	*0.68*	1.00	0.00	0.83	1.00	1.00	*0.83*	1.00	0.12	1.00	1.00	1.00	*0.88*
	CVM	1.00	0.01	0.22	0.99	1.00	*0.71*	0.99	0.02	0.99	1.00	1.00	*0.86*	1.00	0.44	1.00	1.00	1.00	*0.92*
	DRIFTLENS	0.99	0.83	1.00	1.00	1.00	***0.97***	1.00	0.98	1.00	1.00	1.00	***1.00***	0.97	1.00	1.00	1.00	1.00	***0.98***
4 Bias in Bios BERT	MMD	1.00	0.00	0.94	1.00	1.00	*0.85*	1.00	0.00	1.00	1.00	1.00	*0.86*	1.00	0.89	1.00	1.00	1.00	*0.99*
	KS	1.00	0.01	0.70	1.00	1.00	*0.81*	1.00	0.22	1.00	1.00	1.00	*0.89*	0.97	0.99	1.00	1.00	1.00	*0.98*
	LSDD	1.00	0.00	0.95	1.00	1.00	*0.85*	1.00	0.01	1.00	1.00	1.00	*0.86*	1.00	0.99	1.00	1.00	1.00	***1.00***
	CVM	1.00	0.00	0.47	1.00	1.00	*0.77*	1.00	0.13	1.00	1.00	1.00	*0.88*	0.98	0.96	1.00	1.00	1.00	*0.98*
	DRIFTLENS	0.97	0.53	1.00	1.00	1.00	***0.93***	0.98	0.96	1.00	1.00	1.00	***0.99***	0.98	1.00	1.00	1.00	1.00	*0.99*
5.1 MNIST ViT	MMD	1.00	0.00	0.01	0.90	1.00	*0.65*	1.00	0.00	0.64	1.00	1.00	*0.80*	1.00	0.00	1.00	1.00	1.00	*0.86*
	KS	1.00	0.01	0.31	0.99	1.00	*0.73*	1.00	0.07	1.00	1.00	1.00	*0.87*	1.00	0.84	1.00	1.00	1.00	***0.98***
	LSDD	1.00	0.00	0.00	0.77	1.00	*0.61*	1.00	0.00	0.50	1.00	1.00	*0.77*	1.00	0.00	1.00	1.00	1.00	*0.86*
	CVM	1.00	0.00	0.29	1.00	1.00	*0.73*	1.00	0.03	1.00	1.00	1.00	*0.86*	1.00	0.83	1.00	1.00	1.00	***0.98***
	DRIFTLENS	1.00	0.13	1.00	1.00	1.00	***0.88***	1.00	0.43	1.00	0.58	0.85	***0.92***	1.00	0.85	1.00	1.00	1.00	***0.98***
9 FairFace ViT	MMD	0.75	0.38	0.61	0.75	0.92	**0.71**	0.15	0.95	0.99	1.00	1.00	*0.26*	0.01	1.00	1.00	1.00	1.00	*0.01*
	KS	0.39	0.67	0.77	0.89	0.94	*0.52*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	0.85	0.30	0.46	0.71	0.89	*0.70*	0.31	0.89	0.98	1.00	1.00	*0.47*	0.00	1.00	1.00	1.00	1.00	*0.00*
	CVM	0.42	0.64	0.78	0.88	0.96	*0.56*	0.03	0.99	1.00	1.00	1.00	*0.05*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	1.00	0.02	0.07	0.27	0.64	*0.40*	1.00	0.04	0.19	0.65	0.95	***0.63***	1.00	0.06	0.48	0.96	1.00	***0.77***
10 CommonVoice Wav2Vec	MMD	0.14	0.93	0.99	1.00	1.00	*0.24*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	KS	0.10	0.95	0.97	0.99	1.00	*0.17*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	0.02	0.98	1.00	1.00	1.00	*0.05*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	CVM	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	0.97	0.07	0.17	0.27	0.42	***0.38***	0.93	0.19	0.38	0.58	0.85	***0.65***	0.84	0.42	0.75	0.95	0.99	***0.81***

(b) **Smaller data volumes:** Data stream window sizes $m_w \in \{250, 500, 1000\}$

Use Case	Drift Detector	$m_w = 250$						$m_w = 500$						$m_w = 1000$					
		Drift Percentage $D_\%$						Drift Percentage $D_\%$						Drift Percentage $D_\%$					
		0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}
2.1 20Newsgroup BERT	MMD	0.83	0.33	0.74	0.98	1.00	*0.80*	0.06	1.00	1.00	1.00	1.00	*0.11*	0.00	1.00	1.00	1.00	1.00	*0.00*
	KS	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	1.00	0.01	0.04	0.19	0.48	*0.31*	0.98	0.14	0.41	0.76	0.96	*0.72*	0.61	0.65	0.93	1.00	1.00	*0.72*
	CVM	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	0.92	0.28	0.68	0.96	1.00	***0.81***	0.89	0.42	0.92	1.00	1.00	***0.86***	0.84	0.78	1.00	1.00	1.00	***0.89***
3 20Newsgroup BERT	MMD	1.00	0.00	0.04	0.59	0.98	*0.57*	0.99	0.08	0.76	1.00	1.00	*0.83*	0.48	0.89	1.00	1.00	1.00	*0.61*
	KS	0.01	1.00	1.00	1.00	1.00	*0.02*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	1.00	0.00	0.00	0.03	0.35	*0.17*	1.00	0.00	0.02	0.55	0.99	*0.56*	1.00	0.01	0.41	1.00	1.00	*0.75*
	CVM	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	0.97	0.21	0.65	0.96	1.00	***0.82***	0.98	0.35	0.94	1.00	1.00	***0.89***	0.98	0.63	1.00	1.00	1.00	***0.94***
6.1 Intel Image ViT	MMD	1.00	0.00	0.00	0.41	0.99	*0.52*	1.00	0.00	0.21	1.00	1.00	*0.71*	1.00	0.01	0.97	1.00	1.00	*0.85*
	KS	1.00	0.01	0.04	0.53	1.00	*0.56*	1.00	0.03	0.43	1.00	1.00	*0.76*	0.99	0.15	0.99	1.00	1.00	*0.88*
	LSDD	1.00	0.00	0.00	0.08	0.69	*0.32*	1.00	0.00	0.02	0.71	1.00	*0.60*	1.00	0.01	0.49	1.00	1.00	*0.77*
	CVM	1.00	0.01	0.04	0.53	1.00	*0.56*	1.00	0.03	0.47	1.00	1.00	*0.77*	1.00	0.20	0.99	1.00	1.00	*0.89*
	DRIFTLENS	0.95	0.37	1.00	1.00	1.00	***0.88***	0.96	0.57	1.00	1.00	1.00	***0.93***	0.96	0.75	1.00	1.00	1.00	**0.95**
7.1 STL-10 ViT	MMD	1.00	0.00	0.06	1.00	1.00	*0.68*	1.00	0.00	1.00	1.00	1.00	*0.86*	1.00	0.10	1.00	1.00	1.00	*0.87*
	KS	0.97	0.12	0.37	0.98	1.00	*0.76*	0.82	0.53	0.98	1.00	1.00	*0.85*	0.40	0.99	1.00	1.00	1.00	*0.57*
	LSDD	1.00	0.00	0.01	1.00	1.00	*0.67*	1.00	0.00	0.99	1.00	1.00	*0.86*	1.00	0.09	1.00	1.00	1.00	*0.87*
	CVM	0.96	0.18	0.43	0.98	1.00	*0.77*	0.74	0.65	0.99	1.00	1.00	*0.82*	0.24	1.00	1.00	1.00	1.00	*0.38*
	DRIFTLENS	0.96	0.82	1.00	1.00	1.00	***0.96***	0.96	1.00	1.00	1.00	1.00	***0.98***	0.98	1.00	1.00	1.00	1.00	***0.99***
8 STL-10 ViT	MMD	1.00	1.00	1.00	1.00	1.00	***1.00***	1.00	1.00	1.00	1.00	1.00	***1.00***	1.00	1.00	1.00	1.00	1.00	***1.00***
	KS	0.95	1.00	1.00	1.00	1.00	*0.97*	0.65	1.00	1.00	1.00	1.00	*0.79*	0.12	1.00	1.00	1.00	1.00	*0.21*
	LSDD	1.00	0.04	0.16	0.48	0.80	*0.54*	1.00	1.00	1.00	1.00	1.00	***1.00***	1.00	1.00	1.00	1.00	1.00	***1.00***
	CVM	0.94	1.00	1.00	1.00	1.00	*0.97*	0.60	1.00	1.00	1.00	1.00	*0.75*	0.07	1.00	1.00	1.00	1.00	*0.13*
	DRIFTLENS	0.95	1.00	1.00	1.00	1.00	*0.97*	0.93	1.00	1.00	1.00	1.00	*0.96*	0.92	1.00	1.00	1.00	1.00	*0.96*

For DRIFTLENS, MMD [37], KS [37], LSDD [30], CVM [39] detectors [88], and each window size m_w are reported (i) the *accuracy* separately per drift percentage $D_\% \in \{0\%, 5\%, 10\%, 15\%, 20\%\}$, and (ii) the *Harmonic Drift Detection* mean H_{DD}. Accuracy is computed over 100 windows and averaged by repeating 5 runs. The best-performing detector for each use case based on the overall H_{DD} is highlighted, and per window size is in bold.

negligibly with data stream window size, enabling real-time detection in high-throughput data streams. Notably, the running time is always $\leq 0.2\,s$.

Summary of findings: DRIFTLENS is the fastest detector, enabling real-time drift detection regardless of data volumes in the reference set or data stream (answering RQ3 in Section I).

D. Drift Curves Evaluation

This evaluation assesses the ability of DRIFTLENS to represent and characterize the drift curve over time (RQ4 in Section I).

Evaluation Metrics: We use the Spearman coefficient [89] to measure the correlation between the *per-batch* (FDD) distances over time and the curve of injected drift. This coefficient evaluates the monotonic relationship between two variables—they tend to move in the same direction—and is suitable for non-linear patterns (e.g., sudden or periodic changes). It ranges from -1 (perfect negative) to +1 (perfect positive), with 0 indicating no monotonic relationship. The injected drift curve is composed of 0 in windows without drift, and the percentage of drift ($D_\%$) in windows containing some drift. We also qualitatively evaluate examples of drift curves.

Quantitative Results: Table III reports the mean ($\pm$ std) Spearman correlation, computed across all use cases and averaged over 5 runs. Data streams are generated by randomly sampling 100 windows, each with $m_w = 1{,}000$ samples. In the sudden pattern, drift occurs after 50 windows and remains constant

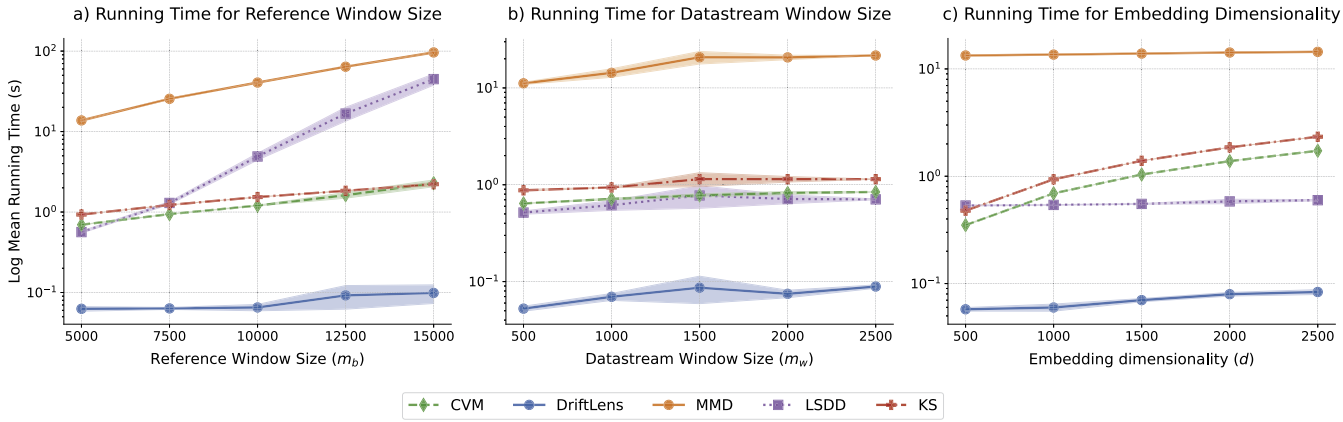


Fig. 5. Running time evaluation. Mean ($\pm$ std) of running time (log scale) for predicting drift with each detector, while varying one dimension at a time: (a) reference window size m_b, (b) data stream window size m_w, and (c) embedding dimensionality d. The other parameters are fixed at $m_b = 5{,}000$, $m_w = 1{,}000$, and $d = 1{,}000$. Mean and std are computed over five runs.

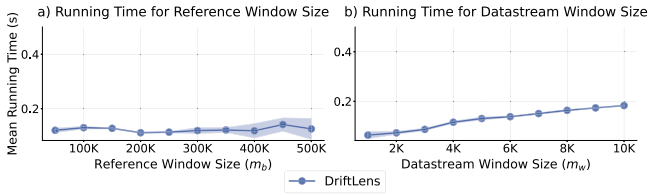


Fig. 6. DRIFTLENS mean running time in seconds as the reference m_b (a) and data stream m_w (b) window sizes change.

TABLE III
DRIFT CURVE EVALUATION

Drift Pattern	*Sudden*	*Incremental*	*Recurrent*
Avg. Spearman Corr.	$0.88 \pm .02$	$0.99 \pm .01$	$0.85 \pm .00$

Spearman correlation between drift severity and *per-batch* distribution distance (*FDD*), per drift pattern, averaged across all use cases.

with a percentage of $D_{\%} = 40\%$. In the incremental pattern, drift occurs after 50 windows with $D_{\%} = 20\%$ and increases by $\Delta D_{\%} = 1\%$ after each window. In the periodic pattern, 20 windows without drift and 20 windows containing $D_{\%} = 40\%$ of drift reoccur periodically. Table III reveals that the *per-batch* (FDD) distances are highly correlated with the generated drift patterns. The correlation exceeds 0.85 for all drift patterns, demonstrating the ability of DRIFTLENS to correctly characterize and model the drift trend over time.

Qualitative Results: Fig. 7 shows the generated monitors for the (a) *sudden*, (b) *incremental*, and (c) *periodic* patterns in use case 6.1. The *per-batch* and *per-label* drift curves are coherent with the generated patterns. The labels *Glacier* (green) and *Mountain* (orange) are the most affected by drift, likely due to the newly injected class (*Sea*) being classified under these labels. In contrast, *Building* (blue) and *Street* (purple) show moderate impact, and *Forest* (red) is nearly unaffected. The drift curves provide valuable insights to characterize drift by showing the most impacted labels, and the drift pattern. Additional examples are reported in Appendix B, available online.

Summary of findings: DRIFTLENS generates drift curves over time that closely align with drift severity, and the *per-label* curves help characterize drift (answering RQ4 in Section I).

E. Drift Explanation Evaluation

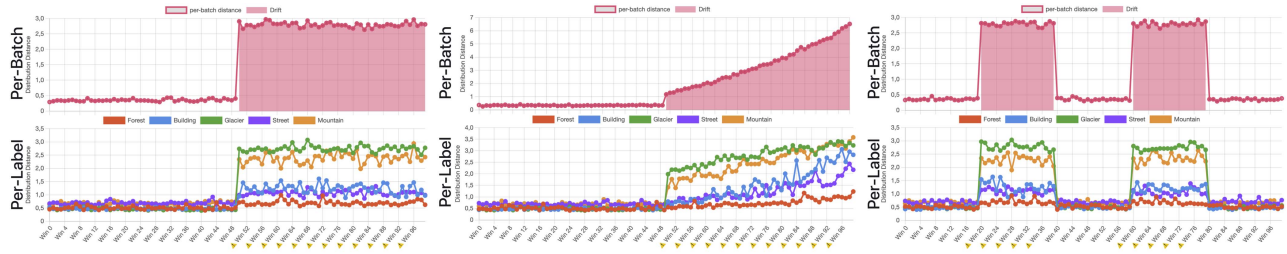
This evaluation assesses the effectiveness of DRIFTLENS in generating *per-label* drift explanations (RQ4 in Section I).

Evaluation Metrics: The *per-label* drift explanations aim to identify prototype samples that caused drift in a label. To extract drifted prototypes from clusters' centroids, clustering must separate non-drifted samples into distinct groups while forming one or more predominantly drifted clusters. Therefore, we measure the *purity* of clustering, which quantifies the extent to which each cluster contains samples from a single class. In our context, instead of the class label, we use two categories: (1) drifted and (2) non-drifted samples. Given a set $X_l = X_l^D \cup X_l^N$ of m_l samples predicted with the label l, where X_l^D are the drifted and X_l^N the non-drifted samples, the K-Means divides X_l into k clusters. Purity is computed as:

$$\text{Purity} = \frac{1}{m_l} \sum_{i=1}^{k} \max\left(|C_i \cap X_l^D|, |C_i \cap X_l^N|\right) \tag{9}$$

Where C_i is the set of samples in cluster i, the terms $|C_i \cap X_l^D|$ and $|C_i \cap X_l^N|$ represent the number of drifted and non-drifted samples in cluster i, and $\max(|C_i \cap X_l^D|, |C_i \cap X_l^N|)$ selects the majority category (drift or no-drift) in each cluster. High purity suggests that clustering successfully isolates drifted samples into one or more clusters, ensuring that prototypes extracted from cluster centroids provide meaningful drifted examples. We also qualitatively discuss examples of drift explanations.

Quantitative Results: Table IV reports the clustering purity for the *per-label* drift explanations. We focus on the label most affected by drift—where drifted samples are more likely to be classified. Specifically, we analyze use cases (label) 1.1 (*World*), 2.1 (*Science*), 4 (*Nurse*), 6.1 (*Glacier*), and 7.1 (*Ship*), based on findings in Appendix B, available online. For each drift severity $D_{\%} \in \{10\%, 15\%, 20\%\}$ and window size $m_w \in \{1000, 2000, 4000\}$, purity is calculated and averaged over 100



(a) Sudden drift. (b) Incremental drift. (c) Periodic drift.

Fig. 7. Drift patterns qualitative evaluation for *use case 6.1* (ViT - Intel Image). Drift is simulated with a new class (*Sea*).

TABLE IV
DRIFT EXPLANATION EVALUATION

Use case	Label	$m_w = 1000$			$m_w = 2000$			$m_w = 4000$		
		10%	15%	20%	10%	15%	20%	10%	15%	20%
1.1	*World*	0.92	0.88	0.85	0.92	0.88	0.84	0.92	0.88	0.84
2.1	*Science*	0.84	0.82	0.80	0.83	0.82	0.81	-	-	-
4	*Nurse*	0.80	0.76	0.73	0.80	0.76	0.73	0.81	0.77	0.74
6.1	*Glacier*	0.80	0.79	0.78	0.79	0.79	0.78	0.79	0.79	0.78
7.1	*Ship*	0.87	0.82	0.79	0.87	0.83	0.79	-	-	-

Purity score for *per-label* clustering of the most affected label, separately per window size m_w and drift percentage $D \in \{10\%, 15\%, 20\%\}$.

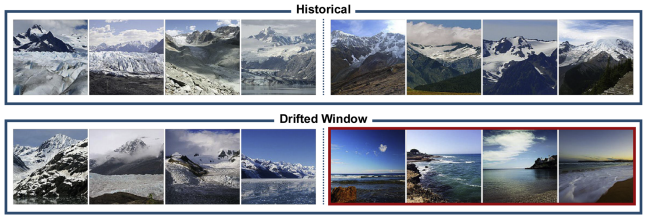


Fig. 8. Drift explanation for class *Glacier* in use case 6.1.

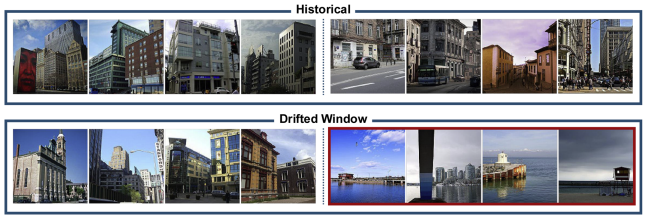


Fig. 9. Drift explanation for class *Building* in use case 6.1.

windows. In use cases 2.1 and 7.1, there are not enough samples for $m_w = 4000$. K-Means is executed multiple times for $k \in [2, 10]$ with the best k selected based on the Silhouette. Table IV shows good clustering performance in all use cases and drift levels, with purity ranging from 0.73 to 0.92. These results highlight that clustering effectively isolates normal and drifted samples, which is crucial to identifying drifted prototypes.

Qualitative Results: Figs. 8 and 9 show the *per-label* drift explanations for use case 6.1, for the classes *Glacier* (highly affected) and *Building* (moderately affected), generated with $m_w = 2000$ and 20% of drift. The closest 4 samples to the centroids are extracted as prototypes. In Fig. 8, historical prototypes for *Glacier* are represented by two clusters: winter glaciers (top-left) and summer glaciers (top-right). When drift occurs, the drifted window is characterized by two clusters: glacier during winter and summer (bottom-left) and general sea images, which are responsible for drift (bottom-right). Similarly, in Fig. 9, *historical* prototypes for the class *Building* are characterized by two clusters of images: buildings only (top-left) and over streets (top-right). When drift occurs, the drifted window is characterized by two clusters: general buildings (bottom-left) and buildings over the sea, which are responsible for drift (bottom-right). These prototypes can help humans understand the drift nature, facilitating the adaptation. Additional qualitative examples are discussed in Appendix B, available online.

Summary of findings: DRIFTLENS provides explanations useful to characterize drift (answering RQ4 in Section I).

VI. CONCLUSION

This paper presents DRIFTLENS, an unsupervised drift detection and characterization framework to continuously monitor drift in deep learning classifiers for unstructured, unlabeled data streams. Our evaluation shows that (i) it is more effective and efficient in detecting drift than state-of-the-art techniques, (ii) it has a short execution time, enabling real-time detection, and (iii) it effectively represents the drift trend over time while characterizing and explaining drifting labels. We release DRIFTLENS as an open-source framework broadly applicable across classifiers and data types. Its *per-batch* analysis is versatile and can extend beyond classification, while the *per-label* analysis is specific for classification or can be adapted to other label-based tasks, such as clustering and object detection.

Limitations: DRIFTLENS uses the Frechét distance and it can inherit some of its limitations. (1) *Noise and selection bias with small datasets.* FDD can be affected by noise and selection bias in distance calculation with small datasets. However, we empirically show that DRIFTLENS performs well even with smaller reference data and window sizes. (2) *Limited number of statistics.* The FDD relies on a limited set of statistics—mean and covariance—for distance calculation, covering only the first two moments of the distribution and not considering higher moments like skewness and kurtosis. (3) *Distance interpretability.* The FDD distances are in the $[0, \infty]$ range, but would be more interpretable in the [0,1] range. This can be addressed by expressing the distance relative to the threshold. (4) DRIFTLENS *assumes embeddings follow Gaussian distributions*, simplifying mathematical modeling but potentially overlooking nuances in

the actual distributions. However, our experiments show that this assumption does not affect drift detection performance while significantly enhancing efficiency. (5) We experimented with drift in windows with nearly balanced labels, but this assumption may not always hold.

In **future work**, we plan to: (i) integrate drift explanations with data type-specific XAI techniques based on embedding representations or concepts [90], [91], [92], (ii) address automatic drift adaptation in unsupervised or supervised with limited label, (iii) test drift detectors in scenarios with unbalanced data, and (iv) extend the approach to tasks other than classification.

References

[1] 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, Dec. 2019.

[2] X. Wang, H. Wang, and D. Yang, "Measure and improve robustness in NLP models: A survey," in *Proc. 2022 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2022, pp. 4569–4586.

[3] S. Shankar and A. G. Parameswaran, "Towards observability for production machine learning pipelines," *Proc. VLDB Endow*, vol. 15, no. 13, pp. 4015–4022, Sep. 2022. [Online]. Available: https://doi.org/10.14778/3565838.3565853

[4] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—A survey on monitoring in evolving environments. part a: Detecting concept drift," *Front. Artif. Intell.*, vol. 7, 2024, Art. no. 1330257. [Online]. Available: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1330257

[5] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—A survey on monitoring in evolving environments. part b: Locating and explaining concept drift," *Front. Artif. Intell.*, vol. 7, 2024, Art. no. 1330258. [Online]. Available: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1330258

[6] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. dos Santos, "An overview of unsupervised drift detection methods," *WIREs Data Mining Knowl. Discov.*, vol. 10, no. 6, 2020, Art. no. e1381.

[7] P. Shen, Y. Ming, H. Li, J. Gao, and W. Zhang, "Unsupervised concept drift detectors: A survey," in *Proc. Adv. Natural Comput. Fuzzy Syst. Knowl. Discov.*, 2023, pp. 1117–1124.

[8] E. Werner, N. Kumar, M. Lieber, S. Torge, S. Gumhold, and W. E. Nagel, "Towards computational performance engineering for unsupervised concept drift detection: Complexities, benchmarking, performance analysis," in *Proc. 13th Int. Conf. Data Sci., Technol. Appl.*, 2024, vol. 1, pp. 318–329.

[9] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Know.-Based Syst.*, vol. 245, 2022, Art. no. 108632. [Online]. Available: https://doi.org/10.1016/j.knosys.2022.108632

[10] H. Hu, M. Kantardzic, and T. S. Sethi, "No free lunch theorem for concept drift detection in streaming data classification: A review," *WIREs Data Mining Knowl. Discov.*, vol. 10, no. 2, 2020, Art. no. e1327.

[11] P. Wang, H. Yu, N. Jin, D. Davies, and W. L. Woo, "QuadCDD: A quadruple-based approach for understanding concept drift in data streams," *Expert Syst. Appl.*, vol. 238, 2024, Art. no. 122114. [Online]. Available: https://doi.org/10.1016/j.eswa.2023.122114

[12] J. N. Adams, C. Pitsch, T. Brockhoff, and W. M. P. van der Aalst, "An experimental evaluation of process concept drift detection," *Proc. VLDB Endow*, vol. 16, no. 8, pp. 1856–1869, Apr. 2023. [Online]. Available: https://doi.org/10.14778/3594512.3594517

[13] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Adv. Artif. Intell.*, 2004, pp. 286–295.

[14] M. Baena-Garcia, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowl. Discov. Data Streams*, 2006, pp. 77–86.

[15] J. Gama and G. Castillo, "Learning with local drift detection," in *Proc. Adv. Data Mining Appl.*, 2006, pp. 42–55.

[16] I. Frías-Blanco, J. D. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, Mar. 2015.

[17] A. Liu, G. Zhang, and J. Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *Proc. 2017 IEEE Int. Conf. Fuzzy Syst.*, 2017, pp. 1–6.

[18] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, pp. 433–449, 2017.

[19] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. 2007 SIAM Int. Conf. Data Mining*, 2007, pp. 443–448.

[20] S. Arora, R. Rani, and N. Saxena, "SETL: A transfer learning based dynamic ensemble classifier for concept drift detection in streaming data," *Cluster Comput.*, vol. 27, pp. 3417–3432, 2023. [Online]. Available: https://doi.org/10.1007/s10586-023-04149-w

[21] P. Vorburger and A. Bernstein, "Entropy-based concept shift detection," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 1113–1118.

[22] P. Grulich, R. Saitenmacher, J. Traub, S. Breß, T. Rabl, and V. Markl, "Scalable detection of concept drifts on data streams with parallel adaptive windowing," in *Proc. 21st Int. Conf. Extending Database Technol.*, 2018, pp. 477–480.

[23] N. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4822–4832, Oct. 2018.

[24] M. Z. A. Mayaki and M. Riveill, "Autoregressive based drift detection method," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2022, pp. 1–8.

[25] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. Thirtieth AAAI Conf. Artif. Intell.*, 2016, pp. 1652–1658.

[26] F. Pinagé, E. M. dos Santos, and J. A. Gama, "A drift detection method based on dynamic classifier selection," *Data Min. Knowl. Discov.*, vol. 34, no. 1, pp. 50–74, Jan. 2020. [Online]. Available: https://doi.org/10.1007/s10618-019-00656-w

[27] A. Liu, J. Lu, Y. Song, J. Xuan, and G. Zhang, "Concept drift detection delay index," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4585–4597, May 2023.

[28] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, pp. 723–773, 2012. [Online]. Available: http://jmlr.org/papers/v13/gretton12a.html

[29] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1545–1554.

[30] L. Bu, C. Alippi, and D. Zhao, "A pdf-Free change detection test based on density difference estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 324–334, Feb. 2018.

[31] T. Cerquitelli, S. Proto, F. Ventura, D. Apiletti, and E. Baralis, "Towards a real-time unsupervised estimation of predictive model degradation," in *Proc. Real-Time Bus. Intell. Anal.*, 2019, pp. 1–6.

[32] F. Ventura et al., "A new unsupervised predictive-model self-assessment approach that scales," in *Proc. IEEE Int. Congr. Big Data*, 2019, pp. 144–148.

[33] S. A. Bashir, A. Petrovski, and D. Doolan, "Udetect: Unsupervised concept change detection for mobile activity recognition," in *Proc. 14th Int. Conf. Adv. Mobile Comput. Multi Media*, 2016, pp. 20–27, doi: 10.1145/3007120.3007144.

[34] R. F. de Mello, Y. Vaz, C. H. Grossi, and A. Bifet, "On learning guarantees to unsupervised concept drift detection on data streams," *Expert Syst. Appl.*, vol. 117, pp. 90–102, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418305682

[35] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proc. 13th Int. Conf. Very Large Data Bases*, 2004, pp. 180–191.

[36] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Discovery Science*. Berlin, Germany: Springer, 2007, pp. 264–269.

[37] S. Rabanser, S. Günnemann, and Z. C. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 1396–1408, [Online]. Available: https://api.semanticscholar.org/CorpusID:53096511

[38] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *J. Amer. Stat. Assoc.*, vol. 46, 1951, Art. no. 253.

[39] H. Cramér, "On the composition of elementary errors: First paper: Mathematical deductions," *Scand. Actuarial J.*, vol. 1928, pp. 13–74, 1928.

[40] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Drift detection in data stream classification without fully labelled instances," in *Proc. 2015 IEEE Int. Conf. Evolving Adaptive Intell. Syst.*, 2015, pp. 1–8.

[41] A. Suprem, J. Arulraj, C. Pu, and J. Ferreira, "ODIN: Automated drift detection and recovery in video analytics," *Proc. VLDB Endow.*, vol. 13, pp. 2453–2465, 2020. [Online]. Available: https://doi.org/10.14778/3407790.3407837

[42] M. Hushchyn and A. Ustyuzhanin, "Generalization of change-point detection in time series data based on direct density ratio estimation," 2020. [Online]. Available: https://arxiv.org/abs/2001.06386

[43] K. Yamanishi and J.-I. Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2002, pp. 676–681, doi: 10.1145/775047.775148.

[44] F. Hinder, V. Vaquet, J. Brinkrolf, and B. Hammer, "On the hardness and necessity of supervised concept drift detection," in *Proc. 12th Int. Conf. Pattern Recognit. Appl. Methods*, 2023, pp. 164–175.

[45] O. Gözüaçik, A. Büyükçakir, H. Bonab, and F. Can, "Unsupervised concept drift detection with a discriminative classifier," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, 2019, pp. 2365–2368, doi: 10.1145/3357384.3358144.

[46] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 2280–2286.

[47] S. Hido, T. Idé, H. Kashima, H. Kubo, and H. Matsuzawa, "Unsupervised change analysis using supervised learning," in *Proc. Adv. Knowl. Discov. Data Mining*, Berlin, Heidelberg, 2008, pp. 148–159.

[48] F. Hinder, V. Vaquet, J. Brinkrolf, and B. Hammer, "Model-based explanations of concept drift," *Neurocomputing*, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231223007634

[49] F. Hinder, A. Artelt, V. Vaquet, and B. Hammer, "Contrasting explanation of concept drift," in *Proc. Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2022, pp. 293–298.

[50] J. N. Adams, S. J. van Zelst, T. Rose, and W. M. van der Aalst, "Explainable concept drift in process mining," *Inf. Syst.*, vol. 114, 2023, Art. no. 102177. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437923000133

[51] L. Yang et al., "CADE: Detecting and explaining concept drift samples for security applications," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2327–2344.

[52] F. Hinder, V. Vaquet, J. Brinkrolf, A. Artelt, and B. Hammer, "Localization of concept drift: Identifying the drifting datapoints," in *Proc. 2022 Int. Joint Conf. Neural Netw.*, 2022, pp. 1–9.

[53] F. Hinder and B. Hammer, "Concept drift segmentation via kolmogorov-trees," in *Proc. ESANN, 29th Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2021.

[54] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–42, Aug. 2018. [Online]. Available: https://doi.org/10.1145/3236009

[55] C. Molnar, *Interpretable Machine Learning*. UAE: Lulu.com, 2020.

[56] X. Wang et al., "Conceptexplorer: Visual analysis of concept drifts in multi-source time-series data," in *Proc. 2020 IEEE Conf. Vis. Analytics Sci. Technol.*, 2020, pp. 1–11.

[57] K. B. Pratt and G. Tschapek, "Visualizing concept drift," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2003, pp. 735–740, doi: 10.1145/956750.956849.

[58] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Min. Knowl. Discov.*, vol. 32, no. 5, pp. 1179–1199, Sep. 2018, doi: 10.1007/s10618-018-0554-1.

[59] J. A. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, pp. 1–37, 2014, doi: 10.1145/2523813.

[60] S. Greco and T. Cerquitelli, "Drift lens: Real-time unsupervised concept drift detection by evaluating per-label embedding distributions," in *Proc. 2021 Int. Conf. Data Mining Workshops*, 2021, pp. 341–349, doi: 10.1109/ICDMW53433.2021.00049.

[61] S. Greco, B. Vacchetti, D. Apiletti, and T. Cerquitelli, "Driftlens: A concept drift detection tool," in *Proc. 27th Int. Conf. Extending Database Technol.*, Paestum, Italy, 2024, pp. 806–809, doi: 10.48786/edbt.2024.75.

[62] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[63] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *J. Multivariate Anal.*, vol. 12, pp. 450–455, 1982.

[64] L. N. Wasserstein, "Markov processes over denumerable products of spaces describing large systems of automata," *Problemy Peredachi Informatsii*, vol. 5, pp. 64–72, 1969.

[65] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local NASH equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6629–6640.

[66] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[67] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 145–151, Jan. 1991.

[68] "Reprint of: Mahalanobis, P.C. 1936, "On the generalised distance in statistics," *Sankhya A*, vol. 80, no. 1, pp. 1–7, Dec. 2018.

[69] A. Bhattacharyya, "On a measure of divergence between two multinomial populations," *Sankhyā: Indian J. Statist.*, vol. 1946, pp. 401–406, 1946. [Online]. Available: http://www.jstor.org/stable/25047882

[70] R. M. J. Byrne, "Counterfactuals in explainable artificial intelligence (XAI): Evidence from human reasoning," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 6276–6282, doi: 10.24963/ijcai.2019/876.

[71] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in AI," in *Proc. Conf. Fairness, Accountability, Transparency*, 2019, pp. 279–288, doi: 10.1145/3287560.3287574.

[72] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.

[73] T. Mitchell, "Twenty newsgroups," *UCI Mach. Learn. Reposit.*, 1997. [Online]. Available: https://doi.org/10.24432/C5C323

[74] M. De-Arteaga et al., "Bias in bios: A case study of semantic representation bias in a high-stakes setting," in *Proc. Conf. Fairness, Accountability, Transparency*, 2019, pp. 120–128.

[75] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[76] I. Corporation, "Intel image classification dataset," [Online]. Available: https://www.kaggle.com/datasets/puneet6060/intel-image-classification

[77] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," *J. Mach. Learn. Res. - Proc. Track*, vol. 15, pp. 215–223, 2011.

[78] K. Karkkainen and J. Joo, "Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2021, pp. 1548–1558.

[79] R. Ardila et al., "Common voice: A massively-multilingual speech corpus," in *Proc. 12th Lang. Resour. Eval. Conf.*, 2020, pp. 4218–4222. [Online]. Available: https://aclanthology.org/2020.lrec-1.520

[80] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[81] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of bert: Smaller, faster, cheaper and lighter," 2020, *arXiv: 1910.01108*.

[82] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv: 1907.11692*.

[83] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[84] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv: 2010.11929*.

[85] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "Wav2Vec: Unsupervised pre-training for speech recognition," 2019, *arXiv: 1904.05862*.

[86] S. Ravfogel, Y. Elazar, H. Gonen, M. Twiton, and Y. Goldberg, "Null it out: Guarding protected attributes by iterative nullspace projection," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7237–7256. [Online]. Available: https://aclanthology.org/2020.acl-main.647/

[87] S. Greco, K. Zhou, L. Capra, T. Cerquitelli, and D. Quercia, "NLPguard: A framework for mitigating the use of protected attributes by nlp classifiers," in *Proc. ACM Hum.-Comput. Interact.*, vol. 8, pp. 1–25, 2024, doi: 10.1145/3686924.

[88] A. Van Looveren et al., "Alibi detect: Algorithms for outlier, adversarial and drift detection," 2019. [Online]. Available: https://github.com/SeldonIO/alibi-detect

[89] *Spearman Rank Correlation Coefficient*. Berlin, Germany: Springer, 2008, pp. 502–505, doi: 10.1007/978-0-387-32833-1_379.

[90] F. Ventura, S. Greco, D. Apiletti, and T. Cerquitelli, "Trusting deep learning natural-language models via local and global explanations," *Knowl. Inf. Syst.*, vol. 64, pp. 1863–1907, 2022, doi: 10.1007/s10115-022-01690-9.

[91] F. Ventura, S. Greco, D. Apiletti, and T. Cerquitelli, "Explaining deep convolutional models by measuring the influence of interpretable features in image classification," *Data Mining Knowl. Discov., vol.* 38, pp. 3169–3226, 2024, doi: 10.1007/s10618-023-00915-x

[92] E. Poeta, G. Ciravegna, E. Pastor, T. Cerquitelli, and E. Baralis, "Concept-based explainable artificial intelligence: A survey," 2023, *arXiv:2312.12936*.