Detecting Changes in Offline and Online Classification Tasks

By

Shuyi Zhang

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
January 2023

ABSTRACT

In machine learning, an essential assumption to build a well-performing classification model is that it should be trained and tested against data that come from the same distribution. However, in the real-world, once a model is in the deployment stage, the control over incoming data is limited. Accurately and efficiently detecting changes violating the fundamental assumption for classification tasks is crucial to ensure the reliability and performance of the artificial intelligence systems.

Different types of changes can arise in offline and online classification tasks. The goals and methods for change detection in the two scenarios are also different. As a starting point, this thesis first focuses on the detection of out-of-distribution examples in the testing data set in offline classification tasks. A purely unsupervised detector **L**abel-**A**ssisted **M**emory **A**uto-**E**ncoder (LAMAE), and its refined version LAMAE+, are proposed to improve the detection of a wider range of out-of-distribution examples. Afterwards, this thesis progresses to the online classification scenario. In a streaming data environment, concept drift, which is a change in the underlying data distribution may occur. Instead of detecting single examples as in the offline scenario, online scenario requires sophisticated algorithms to identify if and when a change occurs in the underlying data distribution. This thesis proposes a novel concept drift detection framework named **H**ierarchical **R**educed-space **D**rift **D**etection framework (HRDD) to meet this goal. HRDD not only recognizes a wider range of drifts regardless of their effects on classification performance, but also does so with an improved efficiency than existing methods. Another challenge faced by existing concept drift detectors is the assumption of data independence on data streams. To further approximate the reality, this thesis also

attempts to investigate the new challenges brought by the relaxation of the independence assumption. A novel problem formulation is constructed taking into account temporal dependency, under which a greater variety of drift forms can possibly emerge. Afterwards, a simple and effective solution named **C**oncept **D**rift detection for **T**emporally **D**ependent data streams (CDTD) to detect drifts, especially the ones that are being neglected by existing detectors, is presented.

In summary, this thesis tackles the detection of change in offline and online classification tasks. The approaches taken in the thesis are both efficient and effective, and have important significance in minimizing the disparity between the simulated environment and the physical reality.

DEDICATION

This thesis is dedicated to my parents

ZHANG Baocheng

and

ZHANG Haizhi,

with love and gratitude.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest and sincere gratitude to my supervisors Prof. Xin Yao and Prof. Peter Tino for their guidance, patience, and support. It has been a privilege to work with them.

I extend my sincere thanks to Prof. Ata Kaban and Dr. Eike Ritter for being my thesis group members. I would also like to thank my colleagues in the lab: Liyan Song, Changwu Huang, Yinghua Yao, Shuxian Li, Zhi Cao and Xiaofen Lu for many fruitful discussions.

I am immensely grateful for the support from my family. My parents, Baocheng Zhang and Haizhi Zhang, have always been there for me, through the good times and the bad. My husband, Zhongqi Zhang, has also been so supportive and caring. Having my family by my side, I can always find my courage and my peace. I would also like to thank my confidants Fan Bu, Aidi Zheng and Xinwei Dong for the emotional support throughout the years.

Finally, I wish to acknowledge Southern University of Science and Technology for providing me with a doctoral scholarship to complete this journey.

Contents

List of Figures

List of Tables

Notations List

$(\mathbf{x}, y)$	A data example
$\mathbf{X}$	The random variable for data feature
Y	The random variable for label
$\mathbb{X} = \mathbf{R}^d$	The d-dimensional input feature space
$\mathbb{Y} = \{0, 1, ..., Q\}$	A finite label space with $Q + 1$ labels
$P(\mathbf{X}, Y)$	The joint probability of random variables $\mathbf{X}$ and Y
$p(\mathbf{x}, y)$	The actual probability for a specific data example
$P(\mathbf{X})$	The probability distribution of $\mathbf{X}$
t	A time stamp
T^*	The actual drift starting time
$\widehat{T}$	The drift detection time
$\mathcal{X} = \{\mathbf{X}_t : t \in \mathbb{Z}^+\}$	The stochastic process for data feature stream
$\mathbb{P}\{\mathcal{X}\}$	The law of stochastic process $\mathcal{X}$

Chapter One

Introduction

Machine learning is changing the world by transforming all segments of our daily lives. By building systems that can learn from historical data, identify patterns, and make logical decisions with little to no human intervention, machine learning tools are acting as a key to guide better decisions and smart actions. Classification is a task of supervised machine learning that refers to the process of anticipating a categorical class label for a specific example of input (Hart et al., 2000). This is also one of the most common machine learning tasks in various application scenarios.

Classification predictive modelling is the task of approximating the mapping function from input features to discrete classes. An input is a vector sampled from a feature space. This can be of various forms, for example, univariate or multivariate, numerical or categorical. In order to construct a classification model, an initial training data set has to be collected. Each training example consists of a set of input features and the respective class label. Then, a function is learnt based on the training data set, which can be used to output the predicted class label for a given example of testing data.

Classification tasks can either be offline or online. Offline learning deals with static training and testing data sets. Online learning refers to the case where data arrives one by one in a sequential order (Wang et al., 2018). Traditional classification algorithms are mainly designed for offline scenarios (Sen et al., 2020). In offline learning tasks, the data

get accumulated and are presented to the model at once, hence there is a requirement for data storage. Besides, due to the lack of adaptability, offline learning methods become more and more restricted especially when live data grow and evolve rapidly.

In contrast to offline learning, online learning is a fundamentally different way of approaching machine learning problems (Langley, 1996; Wang et al., 2018). It refers to the situation where models receive and process data arriving one by one in a sequential order. Online learning is associated with very restricted storage requirement. In most general cases, each data point can be discarded straight away once it has been processed. Streaming data are prevalent in domains such as health monitoring, traffic management, financial transactions and social networks. Online learning mode is ideal for machine learning systems that receive data as a continuous flow. An example of online classification tasks is image recognition tasks in autonomous driving applications. Real-time prediction has to be made for each incoming image upon its arrival. Online learning methods are more adequate for large data. However, there is an extra computational need for in-time model adaptation.

Nowadays, various classification algorithms have been proposed for both online and offline scenarios (Oza and Russell, 2001; Orabona and Crammer, 2010; Kotsiantis, 2013). Nonetheless, an inescapable pre-requisite for almost all conventional machine learning models is the data *independent and identically distributed* (*i.i.d*) assumption. That is, the input data to be predicted and the data used to train the model shall come from the same distribution, and all the input data shall be independent to each other (Kubat and Kubat, 2017).

Although the *i.i.d* assumption is the key to guarantee the reliability and stability of classification models in both offline and online mode, it is very difficult, if not impossible, to be satisfied in reality. In the first place, once the model trained in the offline mode is deployed in an open world scenario, the control over the collected testing data set is limited (Drummond and Shearer, 2006). For instance, in applications such as medical diagnostics and autonomous driving, models are trained in a laboratory environment with

a finite training dataset. The definite label set may not cover all possible rare diseases or objects that may exist in a separately collected testing data set. When the testing data set consists of examples coming from a different distribution, the *identically-distributed* assumption is violated. Such testing data are also known as *out-of-distribution* (OoD) examples, contrasting with the *in-distribution* (ID) training data. In offline classification tasks where the models are trained at one go without any further adaptations, the OoD examples that cannot be handled by the current model should be flagged out and discarded.

The *identically-distributed* assumption is more likely to fail in online scenarios when streaming data is dealt with. As time passes, the underlying distribution of the data and its relationship with the target variable are likely to evolve, also known as *concept drift*. Different from offline scenarios where the learning goal is fixed and OoD examples only need to be neglected, models in online scenario also face the challenge of suitable and timely adaptations when a change takes place. Both the classifier and the detector model have to accommodate the new pattern. Accurately detecting when and where the *identically-distributed* assumption fails is a key to guarantee stable and satisfactory performance throughout the deployment stage.

Besides, the *independently-distributed* assumption is also hardly achievable in classification tasks, especially in an online scenario. The sequential data often arrive following certain orders. For instance, in stock market prediction tasks, whether the price is going up or down today not only depends on the current market demand and supply, but also the market performance of the past few days. Most existing concept drift detection methods assume data independence in the first place to simplify the situation. When dependency exists, the effectiveness and trustworthiness of these methods can no longer be guaranteed. Therefore, a comprehensive study of the effect of dependency on the functionality of existing concept drift detection methods is necessary. It can guide researchers to develop detectors that can better meet practical needs.

In summary, this thesis attempts to make some contributions on the detection

of various types of changes that violate the *identically-distributed* assumption for both offline and online classification tasks. This thesis not only designs sophisticated detection tools that work better in identifying various changes, but also provides an in-depth study for the more complex online scenario with a further relaxation of the *independently-distributed* assumption. Following this, a possible mitigation to accommodate the issue of concept drift detection in a temporally-dependent environment is also proposed with the aim to further reduce the disparity between the simulated laboratory environment and the physical reality.

In this chapter, Section 1.1 describes the two general problems addressed in this thesis and their backgrounds. Section 1.2 provides a clear outline of the specific research questions this thesis aims to answer. Section 1.3 summarizes the main contributions. Section 1.4 presents the thesis organization.

1.1 Problem Areas

1.1.1 Detecting Changes in Offline Classification Tasks

With the advent of rich machine learning models and high computational power, intelligent classification systems are finding more operational applications. Meanwhile, multiple challenges that are not apparent in controlled lab environments can emerge. In an open world setting, unknown inputs may be collected to the testing data set, leading to a discrepancy in the testing and training distributions violating the crucial *identically-distributed* assumption.

Such unknown inputs are named as out-of-distribution (OoD) examples. The threat of OoD examples was first spotted by Hendrycks and Gimpel (2017). Since then it has received increasing attention from the research community. The authors notice that neural network models can produce overconfident predictions even for the OoD examples

with unseen semantic labels in image classification and text categorization tasks. The existence of OoD examples in the testing data set can seriously harm the trustworthiness of machine learning models (Hendrycks et al., 2018). Therefore, OoD detection to identify and reject the unknown examples is critical to prevent irrational predictions, which helps to ensuring the reliability and safety of machine learning systems.

The application of OoD detection usually falls into safety-critical situations including but not limited to autonomous driving (Yuan et al., 2016), surveillance tracking (Zhao et al., 2017) and rare disease identification (Heer et al., 2021). For instance, when an autonomous car trained in Europe is transferred to Asia, a disparate set of traffic rules and signs will be encountered. When a system meets scenes or objects that have never been seen during the training phase, OoD detection tools are expected to notify the users about the abnormalities and allow the system to trigger a safe fallback mode, so that the unknown inputs can be handed over to human to make safe travel instructions.

Hsu et al. (2020) provide a conceptual categorization of OoD examples depending on their semantics. OoD examples with a semantic shift from the training data are the examples that belong to non-overlapping classes as the ID training label set, but with no stylistic change. For instance, both ID and OoD examples are photos but of different object classes. On the other hand, non-semantic shift refers to a change in the style of the inputs from the same label set. e.g., a sketch and a photo of the same object. Non-semantic and semantic shifts can also coexist, resulting in OoD examples with both semantic and stylistic changes compared to the ID data set.

According to Hsu et al. (2020), semantic shifts are the hardest ones to detect, followed by non-semantic shifts. The non-semantic and semantic shifts are the easiest to detect. Most current work in OoD detection regard one benchmark image dataset as the ID set and a completely different data set as the OoD set, which is in essence dealing with the non-semantic and semantic shifts simultaneously. The performance on semantic shifts is less satisfying. The authors pointed out the challenge of detecting semantic shift, which is the hardest type to detect, for future works to address.

Two closely related problem areas are anomaly (or outlier) detection and novelty detection. Anomaly detection aims to detect infrequent patterns that seem to be outside the "normal distribution" of a set of homogeneous examples. The training data usually contains one semantic class only. Based on the above categorization, anomaly detection mainly targets examples with non-semantic shifts (Yang et al., 2021).

Novelty detection, on the other hand, targets at identifying testing examples from a novel class unseen to the classifier during the training stage (Yang et al., 2021). There are two modes of novelty detection: one-class novelty detection and multi-class novelty detection. One-class approaches consider novelty detection as a binary classification problem, hence are limited in scalability when there are multiple classes in the training data set. On the contrary, multi-class novelty detection considers multiple semantic classes in the training data set, hence can be regarded as a similar issue as OoD detection. Yang et al. (2021) conclude in their survey that the key difference between multi-class novelty detection and OoD detection is that the latter also serves the goal of ID classification, so the detection of OoD should not harm the ID classification capability. There is no such requirement for novelty detection tasks.

It is also worth pointing out that some of the anomaly detection and novelty detection methods can be transformed to tools for OoD detection by treating the ID data set as a "normal/known" class and the OoD data set as "abnormal/unknown" (Diers and Pigorsch, 2022). However, these methods may miss some intrinsic characteristics of the ID training data set, hence can only capture some particular types of OoD examples.

Although the number of OoD detection algorithms has been increasing significantly in the last few years, many of the early-proposed OoD detection methods are supervised (Zhou and Paffenroth, 2017; Liang et al., 2017; Lee et al., 2017; Hendrycks et al., 2018; Lee et al., 2018; Shafaei et al., 2019; Yu and Aizawa, 2019; Masana et al., 2018; Ren et al., 2019; Abdelzad et al., 2019). That is, they require the aid of some kinds of genuine or synthetic OoD examples in the training stage. This is an unrealistic requirement since it is often hard, if not impossible, to gain any information regarding OoD a priori. Besides,

the choice of synthetic OoD data set can also heavily impacts the generalization ability of the detectors (Katz-Samuels et al., 2022).

This thesis aims to resolve the issues in existing OoD detection algorithms by developing effective solutions which do not rely on any external information other than what is self-contained within the ID training data, and at the same time, well-detect a wider range of OoD examples.

1.1.2 Detecting Changes in Online Classification Tasks

In conventional offline machine learning framework, it is assumed that the data distribution of interest does not change. Therefore, OoD examples have to be detected and removed from the testing data set. However, in an online scenario where data becomes available in a sequential order, the data generation process may change over time and both the classification model and the change detection algorithm have to adapt to the latest data.

Concept drift refers to a change in the relationships between a change in the underlying data distribution and/or its relationship with the target label over time (Gama et al., 2014). In such cases, the classification model becomes obsolete as the data distribution changes and the predictive function no longer correctly maps features to labels. Sometimes, concept drift could also cause the model to undergo severe performance degradation. Methods and tools to detect such changes as early as possible and pinpoint the time at which the changes occurred are needed. With a comprehensive characterization of the detected drifts, practitioners can also conduct more in-depth analysis of the overall process generating the data. Then, various strategies may be applied to adapt or retrain the classification model to maintain the quality of the operating learning system.

Depending on whether the classification boundary is impacted, drifts are commonly categorized into two different types: *real* and *virtual* (Gama et al., 2004; Hoens

et al., 2012). Real drift refers to a change affecting the classification boundary (Gama et al., 2004). It has been referred to as concept shift in Salganicoff (1997) and conditional change in Gao et al. (2007). In contract, virtual concept drift is defined by a change in data distribution not affecting classification. Virtual drift has also been referred to as temporary drift (Lazarescu et al., 2004) and feature change (Gao et al., 2007).

A plethora of concept drift detection methods has been developed in the literature. Classifier-based detectors, which are also the mainstream methods, focus on detecting real concept drifts which cause direct classification performance deterioration (Lu et al., 2018; Wang et al., 2017). On the contrary, data-based detectors focus more on the detection of virtual drifts (Lu et al., 2020). Virtual drifts are often considered to be less harmful than real drifts (Wang et al., 2018). Nonetheless, although they may not have immediate impact on classification performance, they present a hidden risk that the model may treat certain outlier examples from the new distribution incorrectly. In a multi-drift scenario, detecting underlying drifts before they become dominant on classification performance is also desirable. Hence, both types of drifts are equally important (Wang et al., 2017).

Both classifier-based methods and data-based methods have their inherent limitations. This thesis seeks to provide a more general detection framework for supervised data streams that detects both real and virtual drifts simultaneously, and at the same time, avoid the drawbacks of both types of methods. Meanwhile, detection efficiency and computational cost when handling high-dimensional data streams is also carefully examined and controlled, which is particularly important for online learning tasks.

It is also worth pointing out that almost all existing concept drift detection methods, no matter classifier-based or data-based, rely on the assumption of data independence. The *independently-distributed* assumption is too strict for real-world scenarios. How well existing detectors can perform in an environment with data dependency exists is still unknown. This thesis also includes a rigorous investigation of this commonly-existing situation, systematically points out the new challenges and opportunities of the combined issue, and provides a baseline solution with an aim to inspire future research

in the field.

1.2 Research Problems

Based on the background of change detection in classification problems, this thesis first aims to develop tools to understand and tackle OoD detection problem in offline scenarios, and then escalates to online scenarios where concept drift detection problem is closely looked at. The specific research questions are presented in this section.

1.2.1 Unsupervised Out-of-distribution Detection

Among the wide range of OoD detection methods proposed so far, reconstruction-based methods are gaining increasing attention since they are the only subset of OoD detection methods that is naturally unsupervised and are intuitively reasonable tools with relatively low computational costs (An and Cho, 2015; Berkhahn et al., 2019; Nalisnick et al., 2018). The core assumption behind these methods is that reconstruction models trained with ID examples only cannot well-recover the OoD examples. Hence, the models would produce higher reconstruction error for unseen OoD examples than ID examples. However, recent work has questioned their stability when dealing with various types of OoDs (Denouden et al., 2018; Nalisnick et al., 2018). The validity of this assumption has been shown to depend on the specific characteristics of OoD examples. Sometimes reconstruction-based detectors can "generalize" so well that it can also reconstruct OoD data with low reconstruction error, causing unsatisfactory detection performance (Denouden et al., 2018; Gong et al., 2019). This is especially the case for semantic OoDs. Therefore, we first attempt to answer the following research question: *How to detect various types of OoD examples more accurately in a purely unsupervised manner?*

To answer this questions, this thesis first examines the reason behind the occasional failures of existing reconstruction-based methods, which helps the development of

more advanced detection tools. Our preliminary investigation findings on the detection performance of various types of OoD examples have shown that the current categorization of OoD examples according to their semantics may not be sufficient since not all types of semantic OoD examples are equally difficult to detect. The performance of semantic OoD examples can also vary by a large extent. Hence, in order to improve the trustworthiness of existing detection methods, we study the questions of : *How to characterize different types of OoDs more comprehensively and improve the detection performance for the more difficult OoDs?* To answer these questions, we provide a finer categorization for OoD examples and investigates the possibility of utilizing their intrinsic characteristics to boost detection performance.

1.2.2 Concept Drift Detection for Multivariate Data Streams

After looking at the detection of possible types of change in an offline classification scenario, this thesis extends the time scale and moves to the online scenario where data arrives in the form of a stream. Data streams are likely to be time-varying. Concept drift detection has received growing attention not only because drifts may greatly harm the reliability of real time machine learning systems, but also because it is of practical importance to understand the nature of the data generation process which may reveal valuable information of the application systems (Lu et al., 2018).

Most existing drift detection methods for supervised data streams are classifier-based. They detect real drifts directly by monitoring the classification performance (Lu et al., 2018; Wang et al., 2017). Detecting only real drifts may not be sufficient in many application areas where the reason behind a drift is also essential. Besides, in many applications such as fault detection, it is sometimes possible and crucial to detect a drift before it becomes noticeable on the performance. Besides, the performance of classifier-based methods can heavily depends on the underlying classifier adopted. Various classifiers can lead to very different detection outcomes, harming the applicability and

reliability of the detectors in real-world applications.

While data-based detectors are more focused on the detection of virtual drifts, the earliest detectors are mainly designed for one-dimensional data streams. Detection algorithms designed for multivariate data streams either try to apply statistical tests over the estimated distribution density (Gu et al., 2016; Dong et al., 2017) or examine each dimension individually (Alippi et al., 2010b; Faithfull et al., 2019). Both strategies are not suitable for higher dimensional data streams. For the former, the computational burden of sequential density estimation can be very high, especially when data dimensionality rises (Hwang et al., 1994). For the latter, usually a drift is reported whenever one of the dimensions raises an alarm, hence a high-dimensional data stream increases the possibility of false alarms. To our best knowledge, detectors proposed in existing work have only been tested on data streams with less than 10 dimensions (Schlimmer and Granger, 1986; Losing et al., 2016; Minku and Yao, 2012). In addition, although data-based detectors that monitor the feature input space are also capable of detecting some real drifts that influence the data distribution. Nonetheless, changes affecting the labelling mechanism only cannot be identified.

For the above reasons, the thesis then studies the following set of research questions: *How to detect both real and virtual drifts in supervised data streams regardless of their effect on classification performance? How to improve the efficiency of data distribution-based detector for high-dimensional data streams? How to improve detection performance to achieve high true detections and low false alarms within a specified delay range for all types of drifts even when the magnitude of drift is small?*

To achieve these goals, a novel detection framework which is flexible and efficient for multivariate data streams is proposed. The framework not only pinpoints the time of drift occurrence in a more efficient manner, but also provides relevant information regarding sub-regions of the data distribution. Thus, potentially useful knowledge that could improve model adaptation can be preserved and better utilized to enhance overall performance.

1.2.3 Concept Drift Detection in Data Streams with Temporal Dependency

While concept drift detection methods provide a remedy to raise alarms when the *identically-distributed* assumption of machine learning is violated, the validity of *independently-distributed* assumption and its effect on existing concept drift detection methods still remain relatively unexplored.

This temporal property of data stream is an important characteristic that directly distinguishes a streaming data environment from a non-streaming data environment. Temporal dependency is about the impact of previous events on current event. For instance, in the classic electricity price prediction task (Harries and Wales, 1999), whether the electricity price today is going up or down not only depends on the supply and demand today, but also that of the past few days. Given the fact that temporal dependency is a very common issue in the real world, examining the combined issue of concept drift and temporal dependency is crucial for removing the "simulation-to-reality" gap.

The existence of temporal dependency complicates the data generation process, hence new challenges may arise. Nonetheless, most existing detectors, either explicitly or implicitly, assumes data independence. Thus, the next set of research questions we wish to investigate is: *If and how temporal dependency affects existing concept drift detection methods? What forms of drifts can possibly occur under such scenario? How to detect the wide range of possible drifts in temporal dependent data streams?*

This thesis aims to fill in the research gap by providing a systematic formulation of the joint issue and discussing the new challenges that their interaction could bring to the current state of research. Following this analysis, this thesis also seeks to utilize the temporal dependency itself as a novel concept expression for more efficient and accurate drift detection.

1.3 Research Contributions and Deliverables

The main contributions of this research are summarised as follows:

1. We propose a new OoD detector LAMAE (**L**abel-**A**ssisted **M**emory **A**uto-**E**ncoder) that leverages the information of the class-labels of ID examples to constrain the the reconstruction of OoD samples while preserving the reconstruction capability of ID examples. Hence, the differentiation between ID and OoD can be promoted and better detection can be achieved. We also propose a new criterion to characterize OoD data sets and a refined version of LAMAE named LAMAE+. LAMAE+ mitigates the bias induced by inherent image complexity, it adopts a Complexity Normalizer (CN) to adjust the reconstruction error. Hence, the detection of a wider range of OoDs, including semantic OoDs, can be further improved (Chapter 3).

2. We propose a novel framework for concept drift detection named HRDD (**H**ierarchical **R**educed-space **D**rift **D**etection framework). HRDD detects both real and virtual drift accurately and efficiently for multi-dimensional data streams, and is also capable of detecting subtle drifts. HRDD can be used in conjunction with any base detection test and classifier, and its performance is independent of the choice of the classifier (Chapter 4).

3. We formally demonstrate that temporal dependency is indeed a generic problem that cannot be separated from concept drift detection. We provide a new problem formulation taking into account temporal dependency, and a novel taxonomy for various forms of concept drifts that may exist in the new environment. With this taxonomy and a new data stream generator to simulate different drifts in temporal-dependent data streams, we experimentally demonstrate that when taking temporal dependency into consideration, existing detectors fail in many cases. We summarize the types of drifts requiring urgent attention and conclude that concept drift detection for temporal dependent data streams is an issue worth further investigation (Chapter 5).

4. Based on the new problem formulation, we propose a baseline solution CDTD (**C**oncept **D**rift detection for **T**emporally **D**ependent data streams) detecting concept drifts for temporally dependent data streams (Chapter 5).

The following papers also summarize the findings reported in this thesis:

Published papers:

1. Zhang, S., Tino, P., & Yao, X. (2021). Hierarchical Reduced-space Drift Detection Framework for Multivariate Supervised Data Streams. IEEE Transactions on Knowledge and Data Engineering (Vol 25, no.1, pp. 95-110.).

2. Zhang, S., Pan, C., Song, L., Wu, X., Hu, Z., Pei, K., Tino, P., & Yao, X. (2021). Label-Assisted Memory Autoencoder for Unsupervised Out-of-Distribution Detection. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 795-810).

Working papers ready for submission (The survey paper is based on the proposed future work following the research carried out in this program, hence is not included in this current thesis.):

1. Zhang, S., Tino, P., & Yao, X. (2023). How Important is Temporal Dependency in Concept Drift Detection for Supervised Data Streams? (To be submitted).

2. Zhang, S., Wu, X., Hu, Z., Tino, P., & Yao, X. (2023). A Survey for Online Learning with Recurrent Concept Drift. (To be submitted).

1.4 Thesis Organization

The logical structure of this thesis and the relationship between the chapters are shown in Figure 1.1.

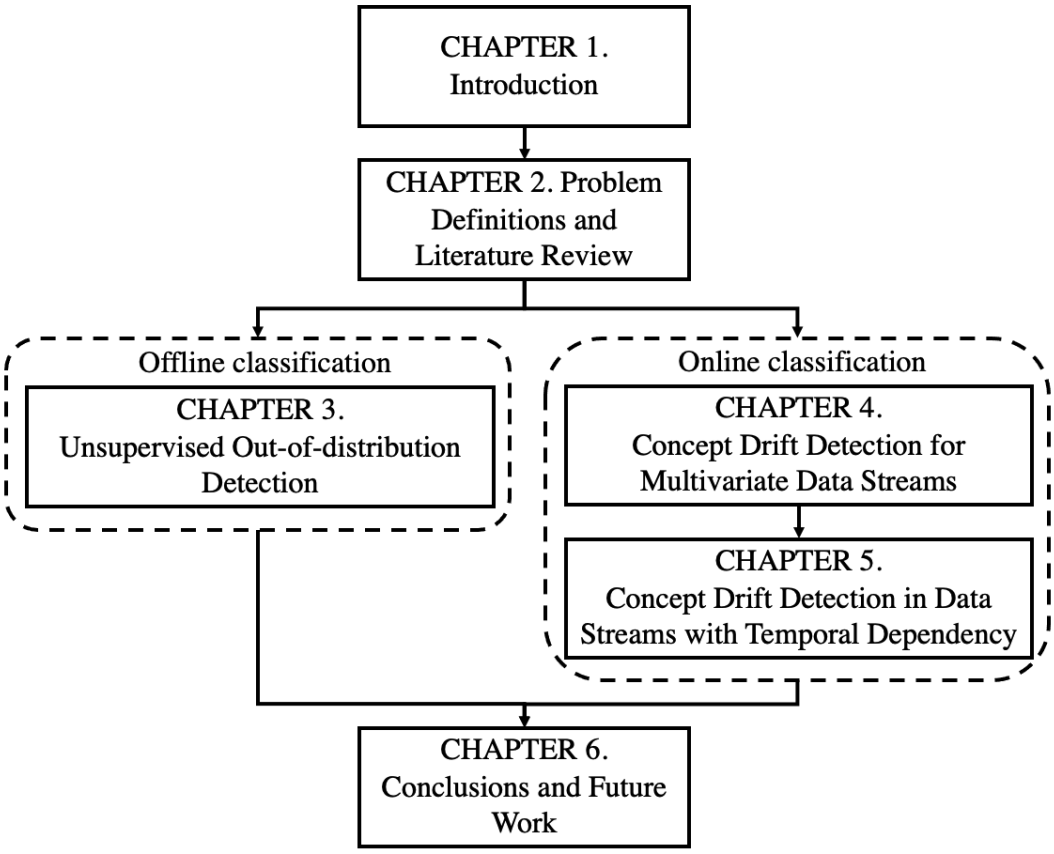


Figure 1.1: Overall thesis structure and relationship between chapters.

In Chapter 2, the formal problem formulations for OoD detection and concept drift detection are introduced in Section 2.1. In particular, Section 2.1.3 summarizes the key similarities and differences between the two problems, showing that they are closely related with the latter being an escalated issue of the former. Then, we provide an inclusive review of existing methods tackling the two problems in Section 2.2.

In Chapter 3, this thesis aims to overcome the limitation of existing OoD detection methods to answer the research questions in Section 1.2.1. A novel reconstruction-based detector, LAMAE, and its variation LAMAE+, are put forward. LAMAE exploits as much self-contained information as possible to aid the training of the underlying reconstruction model. On top of this, we also examine the relationship between the intrinsic characteristics of various OoD examples and their detection difficulties, and then utilized this information to further improve detection performance with LAMAE+. Comprehensive experiments are conducted to evaluate their performance against their competitors.

Chapter 4 moves from the offline scenario to the online scenario. To address the issues listed in Section 1.2.2, HRDD, a novel framework for concept drift detection for multivariate supervised data streams is proposed. This framework has three main novel components and is of high flexibility. HRDD has been demonstrated to remain competitive in terms of both accuracy and efficiency when dealing with various types of drift, no matter the drift is real or virtual. Its superiority also rises when the dimensionality of data streams increases. Therefore, HRDD is shown to be a very promising procedure to follow when designing more advanced concept drift detectors.

In Chapter 5, the $i.i.d$ assumption of existing concept drift detectors is further relaxed. An investigation to gain a better understanding of the role of temporal dependency in concept drift detection tasks is provided. To tackle the research questions in Section 1.2.3, this thesis first provides a new problem formulation that better describes the background of supervised data streams, based on which a new taxonomy of possible forms of drifts is presented. This thesis not only theoretically illustrates the close connection between temporal dependency and concept drift, but also creates a drift simulator so that the performance of existing drift detectors can be illustrated experimentally. Then, a novel detector CDTD that well accommodates this new situation is developed to fill in the gap of existing research.

Finally, Chapter 6 concludes the paper by summarizing the findings of this thesis and pointing to directions for future work.

Chapter Two

Problem Definitions and Literature Review

This chapter consists of two sections. Section 2.1 provides the formal problem definitions for the detection of changes in both offline and online classification tasks: OoD detection and concept drift detection. Then a summary of the key differences and similarities between the two problems is also presented. Section 2.2 consists of a comprehensive literature review for each of the two problems.

2.1 Problem Definitions

This section introduces the formal problem definitions for OoD detection and concept drift detection. Since both problems are dealing with changes in machine learning classification tasks, this section starts by describing the background of classification. The following set of notations will be used throughout the thesis.

In supervised learning, each example is a pair $(\mathbf{x}, y)$ consisting of a d-dimensional feature vector $\mathbf{x} \in \mathbb{X} = \mathcal{R}^d$ and a respective class label $y \in \mathbb{Y} = \{0, 1, ..., Q\}$. For a binary classification task, $\mathbb{Y} = \{0, 1\}$. The goal of classification task is to train a model $f_{clf}(\cdot)$ in order to predict the label $\hat{y}$ for new input data. i.e., $\hat{y} = f_{clf}(\mathbf{x})$. The predictions are only

reliable when the new input data and the data used to train the model are identically distributed. That is, they come from the same joint probability distribution $P(\mathbf{X}, Y)$. Various types of change could exist in different scenarios. In sections 2.1.1 and 2.1.2 below, we explain two specific types of changes this thesis aims to tackle in offline and online supervised learning tasks: OoD detection and concept drift detection.

For online learning, there are actually two essential stages involved, concept drift detection and classifier adaption. This thesis focuses on the detection of change only. Nonetheless, for the sake of completeness, a brief review of the adaptation stage is also included in Section 2.1.3.

2.1.1 OoD Detection

In offline supervised learning scenario, a training dataset can be used to learn an optimal mapping to correctly determine the class labels for examples in a separately collected testing dataset. Formally, a training data set D_{train} consisted of L examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$ is used to train a predictive classifier $f_{clf}(\cdot)$. All examples in D_{train} are generated by a so-called in-distribution (ID) probability function P_{in}. In order to guarantee the performance of model $f_{clf}(\cdot)$, it is crucial to ensure that the testing data set D_{test} contains only examples drawn from the ID distribution P_{in}. However, it is often impossible to control the collected testing dataset, especially in an open-world scenario. D_{test} may contain completely different inputs that ruin the *identically-distributed* assumption and therefore render classic learning theory inapplicable. An example $(\mathbf{x}, y) \sim P_{out}$ where $P_{out} \neq P_{in}$ is defined as an out-of-distribution (OoD) example.

OoD detection aims to identify the OoD examples and discard them in order to help the classifier avoid making wrong predictions. Ideally, the detection model f_{ood} shall be purely unsupervised, meaning that it is also to be built with the ID training data set D_{train} only. For each example in the testing data set, a test statistic is evaluated based on the trained detection model f_{ood}. Then the test statistic is compared to certain threshold

Figure 2.1: Illustration of datasets MNIST (left) (LeCun, 1998), SVHN (middle) (Netzer et al., 2011) and CIFAR10 (right) (Krizhevsky and Hinton, 2009).

to determine if the example is OoD. In OoD detection tasks, it is also important not to harm the ID classification performance (Yang et al., 2021).

The existence of OoD examples in the testing data set is essentially a change in the joint probability P_{in}. Note that a joint probability $P(\mathbf{X}, Y)$ can be further decomposed into $P(Y) \cdot P(\mathbf{X}|Y)$. This decomposition also corresponds to the categorization of OoD examples provided by Hsu et al. (2020). The authors conceptually categorize possible changes brought by OoD examples into three categories, semantic (S) shift, non-semantic (NS) shift, and the combination of the two (NS+S), by whether a shift is related to the inclusion of new semantic categories. A semantic shift refers to the occurrence of new semantic classes. Formally, semantic shift refers to the situation where data are drawn from a distribution $P_{out}(\mathbf{X}, \bar{Y})$ with $\{\bar{Y}\} \cap \{Y\} = \emptyset$. In other words, semantic OoDs belong to a label set $\bar{Y}$ that is completely different from that of the ID data Y. If the label set of OoD examples remains the same, but only the forms of presentation change, the shift is said to be non-semantic. Non-semantic shift is also widely discussed in the problem of model generalization and robustness improvements. Both types of shift can also exist simultaneously in some OoD examples, leading to a non-semantic and semantic (NS+S) shift.

To better understand the differences between each type of shift, we provide an illustrative example of the testing protocol in Table 2.1. As shown in Figure 2.1, MNIST is a dataset for handwritten digits with labels "0" to "9" (LeCun, 1998), SVHN is a

Table 2.1: Examples of the testing protocol setting with different types of OoD following the existing categorization based on semantics (Hsu et al., 2020).

ID data set	ID label set	OoD data set	OoD label set	Type of OoD
“1” to “4” from MNIST	{“1”, ... ,“4”}	“5” to “9” from MNIST	{“5”, ... ,“9”}	Semantic
		“1” to “4” from SVHN	{“1”, ... ,“4”}	Non-semantic
		all classes from CIFAR10	{“airplane”, “bird”, ...}	Non-semantic and semantic

dataset for door number digits with the same set of labels “0” to “9” (Netzer et al., 2011), and CIFAR10 is a dataset of natural images of different objects such as airplane, bird and cat (Krizhevsky and Hinton, 2009). Assuming digits “1” to “4” from the MNIST dataset is taken as the ID dataset. An example of S shift could be OoD examples from “5” to “9” from the MNIST dataset, with a change in the label space only. An example of NS shift could be OoD examples with labels “1” to “4” from the SVHN dataset. The form of presentation switches from handwritten pattern to photographic formats, without changing the underlying class labels. An example for the presence of NS+S shift could be OoD examples drawn from CIFAR10 dataset, where both the semantic and presentation of the data are changed. Detection of NS+S shifts is the current mainstream of OOD detection algorithms. However, according to Hsu et al. (2020), NS+S shift is the easiest type to detect, followed by NS shift. S shift turns out to be the hardest one to detect.

Depending on whether the classifier for ID data classification is involved for detection, OoD detection methods can be divided into two categories: classifier-based detectors and data-based detectors. A detailed summary of existing methods is presented in Section 2.2.1.

2.1.2 Concept Drift Detection

In online scenario, data arrives in a sequential, continuous fashion. Data streams are likely to be time-varying. The dynamicity of real-world systems poses a significant challenge to deployed predictive machine learning models. Formally, a supervised data stream to be inspected for change is formed by observations $\{(\mathbf{x}_t, y_t), t \in \mathbb{Z}^+\}$. $\mathbf{x}_t$ represents the observation at time stamp t and y_t is its class label. Concept drift refers to a change in the joint probability $P_t(\mathbf{X}, Y)$ generating the data stream (Gama et al., 2014). Concept drift detection aims to accurately detect such drifts as time goes on. Formally, a concept drift is said to occur at time T^* if $\{(\mathbf{x}_t, y_t)\} \sim P_{t_0}(\mathbf{X}, Y)$ for $t < T^*$, $\{(\mathbf{x}_t, y_t)\} \sim P_{t_1}(\mathbf{X}, Y)$ for $t \geq T^*$, and $P_{t_0}(\mathbf{X}, Y) \neq P_{t_1}(\mathbf{X}, Y)$ in a single-drift scenario. Multi-drift scenarios can be defined analogously.

To simplify the situation in the first place, most existing work assume that observations are generated independently (Harel et al., 2014; Bu et al., 2016; Gama et al., 2004; Baena-García et al., 2006; Ross et al., 2012). Formally, for observations on a stationary concept starting and ending at time stamps 1 and h respectively under the assumption of independence, $P(\mathbf{X}, Y) = p((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_t, y_t)) = p(\mathbf{x}_1, y_1) \cdot p(\mathbf{x}_2, y_2) \cdot ... \cdot p(\mathbf{x}_h, y_h)$. Thus, drifts can be detected by monitoring if there is a significant change in $p(\mathbf{x}_t, y_t)$ at each time stamp t.

Note that the joint probability $P(\mathbf{X}, Y)$ can be written as

$$P(\mathbf{X}, Y) = P(Y|\mathbf{X}) \cdot P(\mathbf{X}), \tag{2.1}$$

where $P(\mathbf{X})$ can be further decomposed through marginalization

$$P(\mathbf{X}) = \sum_{q=0}^{Q} P(Y = q) \cdot P(\mathbf{X}|Y = q). \tag{2.2}$$

Based on the probabilistic definition of a concept drift and the above decomposition, it is not difficult to tell that the change can manifest itself in different forms corresponding to the different components of the joint probability (Webb et al., 2016; Gao et al., 2007). Drift can occur in: 1) the marginal distribution over covariates $P(\mathbf{X})$; 2) the posterior

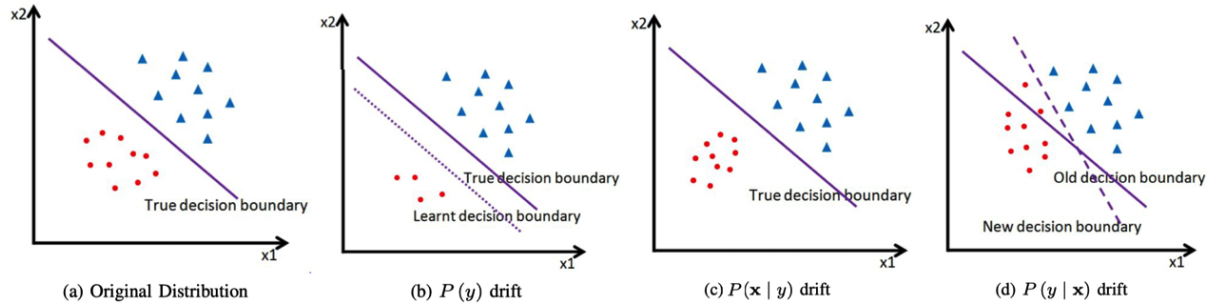


Figure 2.2: Illustration of three concept drift types (Wang et al., 2017).

class probability or classification boundary $P(Y|\mathbf{X})$; 3) one or more class-conditional distributions $P(\mathbf{X}|Y)$ and 4) the prior distribution $P(Y)$.

The most widely used categorization for concept drifts is to classify them by examining their impacts on the classification boundary (Gama et al., 2004; Ramírez-Gallego et al., 2017). Real drifts refer to drifts affecting $P(Y|\mathbf{X})$ only. Virtual drifts are changes in $P(\mathbf{X})$ not affecting $P(Y|\mathbf{X})$. Figure 2.2 provides an example of three types of drift. Among them, type (d) is considered to be a real drift, type (b-c) are virtual drifts.

Different from OoD detection where most work aim to detect the existence of new semantic classes, it is often assumed in concept drift detection problems that the label space is balanced and remains constant. The emergence of new classes (as well as disappearance of existing classes) is a special sub-category of concept drift, also known as concept evolution (Haque et al., 2016). Concept evolution relies on a slightly different problem setting and testing protocol. It is beyond the scope of this thesis. This thesis focuses on the more general case of concept drift.

Categorizing drift by its influence on $P(Y|\mathbf{X})$ is one possible way to describe the drift. Drifts also posses quantitative properties. There are also some research to measure the magnitude, speed and frequency of drifts, for instance, drift severity (Minku and Yao, 2009), degree of dominance (Minku and Yao, 2009), drift magnitude (Webb et al., 2016), drift path length (Webb et al., 2016), drift rate (Bartlett et al., 2000; Webb et al., 2016), drift duration (Webb et al., 2016), drift interval and drift volatility (Huang et al., 2014). Drifts can also be categorized into different classes based on these measures, for instance,

a drift with duration 1 is an abrupt drift. However, many of the quantitative measures can be very difficult to calculate in reality.

A concept drift detector usually consists of three stages: detector initialization, operation and reconfiguration. Usually, it is assumed that at least an initial segment of data stream is stationary so that a set of initial supervised observations can be used to constitute a training sequence (Alippi et al., 2010b). The initialization of the detector is built with this initial training sequence. Then as new data is being received, the carefully designed concept features and test statistics are extracted and monitored. A detection is raised whenever the pre-defined criterion is met. Once an alarm is confirmed, the detector should be able to reconfigure itself on the new concept as soon as possible. Ideally, no human intervention should be required.

Similar to OoD detection methods, concept drift detection methods can also be categorized into classifier-based methods and data-based methods. A detailed review of existing methods is presented in Section 2.2.3.

2.1.3 From Offline to Online: Static to Dynamic

From the above problem formulations, it can be seen that both OoD and concept drift detection are two closely related topic areas under different learning scenarios. For offline learning scenario, the learning task of interest is static. In other words, the underlying distribution generating the data of interest (the *in-distribution*) remains unchanged. On the other hand, for online learning scenario, the learning task is non-stationary. The underlying data distribution of interest as well as the task goals can change as time passes. Therefore, concept drift detection can be regarded as an escalated problem of OoD detection since it deals with a more complicated dynamic environment.

An additional module of online learning tasks is the adaptation stage to accommodate the latest concept. Two general types of adaptation approach commonly exist:

Table 2.2: A comprehensive comparison between OoD detection and concept drift detection.

	OoD Detection (Hendrycks and Gimpel, 2017)	Concept Drift Detection (Baena-García et al., 2006)
Dissimilarity	Offline learning scenario	Online learning scenario
	Detection of OoD examples	Detection of the drift times
	No assumption on the label space Y	Assumes a fixed label space Y except for work in concept evolution
	Detection made for each example independently	Detection made based on a sequence of examples
	Discards detected examples and maintains previous information	Discards past information after detections and adapt the new information
	No reconfiguration required	Requires self-reconfiguration after each detection
Similarity	Detection of changes affecting $P(\mathbf{X}, Y)$ No prior knowledge of the changes should be available Detection methods can be classifier-based or data-based	

active approaches and passive approaches. Active approaches rely on the concept drift detection results to decide when and how to update the classification model. The most standard adaptation strategy is to retrain a new model with the latest data representing the new concept. Many existing drift detection methods, especially the classifier-based ones, automatically save the suspicious data belonging to the new concept whenever a warning signal is triggered (Baena-García et al., 2006; Frias-Blanco et al., 2014; Barros et al., 2017). These data can then be used for classification model retraining. For other methods, a small window of the most recent data may be stored as time passes by. After a drift is detected, the detection methods trace back to estimate the potential starting point of the new concept and use the appropriate data for retraining (Alippi et al., 2011b; Yu et al., 2018). In addition to simple model retraining, model modification is another branch of adaptation strategy. Ensemble approaches and neural network-based approaches are

common in this category (Xu and Wang, 2017; Museba et al., 2021). For instance, when the classifier-based drift detector raises an alarm, Dynamic Extreme Learning Machine (DELM) (Xu and Wang, 2017) adds more nodes to the network layers to improve its approximation capability. Without training new classification models completely from scratch, the computational cost can be reduced. Besides, previous information can also be stored, which may benefit classification performance if similar concept reappears.

For active approaches, the accurate detection of concept drift is crucial in guaranteeing the overall performance of the classification system. Passive approaches, on the contrary, are not incorporated with an explicit drift detection mechanism. Instead, they constantly adapt themselves on the latest data (Hulten et al., 2001; Kolter and Maloof, 2007; Brzezinski and Stefanowski, 2013). Consequently, they can also slowly adapt to new concept when a drift takes place with a relatively low speed. Although the cost of incremental maintenance may be low, these methods can only adapt to certain types of drifts. Furthermore, they hardly provide any information about the underlying data generation process, making it difficult for practitioners to understand the reason behind the drifts.

It is worth pointing out that detecting concept drifts and adapting the classification model to the data are two different mechanisms. From the practical point of view, an accurate detector is crucial for maintaining good classification performance in the long run. How to choose the most appropriate underlying classification model and how to update the model after each detection are a set of separate question beyond the scope of this thesis.

This thesis focuses on the detection stage only. Table 2.2 summarizes the key differences and similarities between OoD detection and concept drift detection. Although OoD detection and concept drift detection deal with different types of change in various scenarios, they still have many characteristics in common. The standard procedures for detection are also very similar and can be summarized as two stages: test statistic extraction and similarity comparison. It can also be noted that detection methods for both

problems can be summarized as classifier-based and data-based approaches. The following section will summarize state-of-the-art methods and point out the current research gaps in each problem.

2.2 Literature Review

This section systematically categorizes and reviews the state-of-the-art methods for OoD detection and concept drift detection in sections 2.2.1 and 2.2.3, respectively.

2.2.1 OoD Detection Methods

In this section, OoD detection methods are first categorized into classifier-based methods and data-based methods according to whether they rely on the ID classification model to conduct inference. Sub-categories also exist. Figure 2.3 shows the overall taxonomy of OoD detection methods.

Classifier-based methods rely on the classifier for ID data to detect OoD. Data-based methods do not require the involvement of an explicit classifier. They utilize tools to extract data-related information for OoD detection. For classifier-based methods, it is also important that the classification performance of ID data is maintained. For data-based methods, this requirement is automatically satisfied because an explicit mechanism is adopted for detection. The classification model for ID examples is not altered. It has

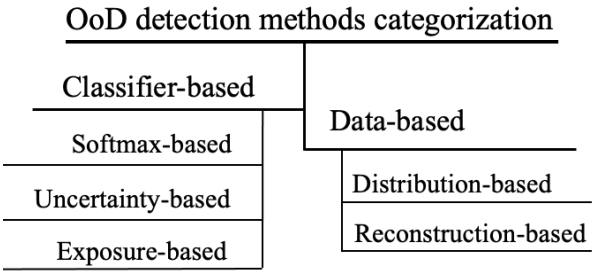


Figure 2.3: Categorization of existing OoD detection methods.

been stated earlier in section 1.1 that some anomaly detection and novelty detection algorithms can also be easily adapted for OoD detection, if all ID classes are regarded as a single normal class, and all other unseen classes as anomalous. Therefore, the review of data-based methods will also contain some relevant anomaly detection and novelty detection methods.

OoD detection methods can also be either supervised or unsupervised. Supervised methods are the ones that rely on some external genuine or artificial OoD examples for training or parameter-tuning. Although this can lead to better results in some test cases, such information is hardly available in the real world. The characteristics of the given or synthetically generated examples also limit the generalization ability of the detectors. The majority of classifier-based methods are supervised methods. The detection performance of such methods also depends on the performance of the associated ID classifier. Details of these methods will be explained later in this section.

Unsupervised detection methods, on the other hand, do not require any OoD data for training. In other words, they make no assumptions on the OoD data. They work in a similar manner as one-class classifiers. This is a desired property of OoD detection methods. While data-based methods are mostly unsupervised, they still have their own limitations such as the difficulty in finding the appropriate feature extraction tool and high computational cost.

Classifier-based OoD Detection Methods

Classifier-based approaches can be further divided into three categories, depending on what test statistics are used and how they are obtained. Softmax-based methods utilize direct information from the ID classifier for detection. Uncertainty-based methods employ uncertainty modelling techniques for detection. Exposure-based methods aim to mine or synthesize useful OoD examples and utilize them together with the ID data to train a classification model for OoD detection.

softmax-based methods

In one of the first works on OoD detection, Hendrycks and Gimpel (2017) present a baseline method based on maximum softmax probability (MSP). The underlying assumption is that for a neural network well-trained with ID training data only, higher softmax scores will be assigned to ID testing examples than OoD examples, allowing for successful differentiation. A new input is considered to be ID if the softmax score is above a certain threshold and OoD otherwise. Whilst intuitive, it is found that deep neural networks can easily produce abnormally high softmax scores even for inputs far away from the ID data (Hendrycks and Gimpel, 2017).

Later on, many methods have been proposed to improve the detection performance of using softmax scores. For instance, Out-of-DistrIbution detector for Neural networks (ODIN) (Liang et al., 2017) proposes to increase the difference between the softmax scores of ID and OoD samples by adopting two techniques to calibrate pre-trained models: temperature scaling and input perturbation. Temperature scaling calibrates the softmax scores by scaling the logits by a large constant parameter, which is referred to as the temperature (Lin et al., 2021). A sufficiently large temperature has a strong smoothing effect on the softmax scores which can help to distinguish ID and OoD effectively. Input perturbation is a data pre-processing step before feeding the example to the neural network. The amount of perturbation also relies on an additional scale parameter. Results from the experiments in the original paper have shown that the perturbation can have stronger effect on the ID examples than that on OoD examples, making them more separable. The key challenge for ODIN is to select the appropriate hyper parameters. A good manipulation of the parameters can significantly improve detection performance.

The loss function of neural network models can also be modified. Wei et al. (2022) observe that the common softmax cross entropy loss is a reason for the abnormally high softmax scores assigned to the OoD examples. They propose a simple fix by enforcing a constant norm on the logit vector so as to limit the magnitude of softmax output vectors.

This logit normalization can be viewed as an input-dependent, rather than a global constant, temperature applied on the logits. For hyper-parameter tuning, a Gaussian noises data set is used as the validation set.

The above methods are built with state-of-the-art model architectures. However, with limited modifications, detection performance can also be restricted. The structure of networks can also be altered to achieve better detection performance. Yu and Aizawa (2019) propose to use a two-head CNN inspired by Saito et al. (2018). The model consists of a common feature extractor network and two classifiers. The network is first trained to classify ID examples correctly and then trained to maximize the discrepancy between the classifiers outcomes on some unlabelled data set with mixed ID and OoD data. At inference time, the $L1$ distance between the two classifiers softmax outputs is calculated and used for detection. Longer distance symbolizes higher probability of being OoD. However, the fine-tune of the classifier to detect OoD examples leads to a drop in classification accuracy compared to the original classifier, which is an undesirable outcome violating the premise of OoD detection.

Ensemble methods that combine softmax scores from several neural network models are also proposed. For instance, Vyas et al. (2018) propose to detect OoD examples by examining the softmax probabilities of all base learners within an ensemble classifier. An ID dataset and a small number of OoD examples are utilized together for training. The ID training dataset is partitioned into subsets mutually exclusive to each other to form leave-out training data sets. The authors also propose a novel loss term which encourages the probabilities of all the classes to be equal for OoD examples, while maximizing the probabilities of the correct classes for ID examples. Temperature scaling and input perturbation techniques are also applied. Although effective, the proposed method of the ensemble of neural networks requires large memory and computational resources.

As can be seen from above, softmax-based methods are mainly supervised. For instance, ODIN (Liang et al., 2017) selects an independent data set that is neither included in the ID data sets nor the OoD sets for hyperparameter tuning. In the work

by Yu and Aizawa (2019), it is assumed that in addition to the ID data sets, a set of unlabelled images containing at least partial OoD images are accessible during training.

Not until very recently have researchers started to develop OoD detection methods that do not rely on any external information. Generalized ODIN (GODIN) (Hsu et al., 2020) is a recently proposed OoD detector that do not require any OoD data for training. It is an advancement based on one of the benchmark detectors, ODIN (Liang et al., 2017). GODIN frees the algorithm from explicit parameter-tuning with respect to specific OoD datasets for both temperature scaling and input perturbation techniques. It decomposes the class posterior probability using the rule of conditional probability during training and uses only the numerator, i.e., the joint class-domain probability, for detection.

A recent work conclude that OoD performance is highly correlated with its accuracy on the ID classes. Taking MSP as an example, the authors Vaze et al. (2021) demonstrate that a good closed-set classifier is sufficient for good OoD detection. This argument also highlights the fact that the strength of softmax-based OoD detectors is dependent on the base classifier involved. It is also hard to achieve outstanding OoD detection performance for the more difficult classification tasks.

uncertainty-based methods

Many existing work have demonstrated that prediction probability from a softmax distribution has a poor direct correspondence to confidence (Nguyen and O'Connor, 2015; Nguyen et al., 2015). Even with a high softmax output, a model can still be uncertain in its predictions (Gal and Ghahramani, 2016). Uncertainty- (or confidence-) based approaches attempt to exploit and utilize model uncertainty (or confidence) for detection. Uncertainty-based methods are based on the premise that a classifier trained with ID examples should be less uncertain (more confident) about its prediction when the input is ID. Therefore, examples having lower confidence scores are classified as OoD.

The focus of various uncertainty-based methods is on the accurate measure of

model uncertainty. For instance, based on the baseline detection method MSP proposed by Hendrycks and Gimpel (2017), DeVries and Taylor (2018) attach an auxiliary confidence estimation branch onto a pre-trained classifier to output a single confidence score. The authors amend the training loss function to a multi-task loss so that the network can minimize its overall loss if it can successfully predict which inputs are more likely to be classified incorrectly.

Bayesian models, which learn a distribution over weights, are popular tools for uncertainty estimation (Rasmussen and Quinonero-Candela, 2005; Gal and Ghahramani, 2016). For instance, Malinin and Gales (2018) and Chen et al. (2018) first attempt to parametrize a Dirichlet distribution over the output class distributions using deep neural networks, and then use various measures of entropy of the learned posterior distribution as the uncertainty measure for OoD detection. Examples with high entropy are more likely to be OoDs.

ENsemble Distribution Distillation (EnD2) (Malinin et al., 2019) is an ensemble-based approach from the Bayesian viewpoint. In EnD2, a criterion for mutual information that decomposes the total uncertainty into knowledge uncertainty and expected data uncertainty is calculated. For OoD inputs, the ensemble yields diverse distributions over classes such that the predictive posterior is near uniform, while the expected entropy of each sub-model is much lower. EnD2 also distils the distribution of the predictions from an ensemble into a single model. Distillation is the procedure of teaching a single network to represent the knowledge of a group of neural networks (Bucilua et al., 2006). With distillation, the computational cost of the ensemble is reduced, while the ensemble model diversity is maintained.

However, Bayesian-methods are difficult to train and are associated with high computational costs (Lakshminarayanan et al., 2016). Lakshminarayanan et al. (2016) propose an alternative to Bayesian neural networks named Deep Ensemble. It is a simple and scalable non-Bayesian solution for uncertainty quantification. It consists of three components: network training with a proper scoring rule that fairly judges the quality

of predictive uncertainty, adversarial training to smooth the predictive distributions, and an ensemble structure which helps generate well-calibrated uncertainty estimates.

Maximizing Overall Diversity (MOD) (Jain et al., 2020) notices the excessive sampling variability in the uncertainty estimates of ensemble neural network models, especially for low-density regions of the ID distribution. The authors aim to improve and stabilize model uncertainty estimation by encouraging higher ensemble diversity. It introduces an auxiliary diversity-encouraging loss that is computed over some synthetic OoD data.

Diversity inducing Information Bottleneck enSemble model (Dibs) (Sinha et al., 2021) is another ensemble learning method that promotes diversity among pairwise latent ensemble variables. It adopts a measure that effectively captures the combined predictive uncertainty (epistemic uncertainty and aleatoric uncertainty) for reliable OOD detection. MOD and Dibs form a promising direction of increasing diversity to aid detection, but require computationally expensive adversarial setups (Mehrtens et al., 2022).

Uncertainty-based methods are in general less competitive for OoD detection for the following reason. Even if the uncertainty can be accurately estimated, rejecting uncertain examples is not enough since uncertain is not the same as unknown, and the unknown OoD examples need not appear to be uncertain to a learning system (Boult et al., 2019).

exposure-based methods

Exposure-based methods assume that an OoD data set is available for training. They deliberately synthesize and/or select OoD examples and utilize them for training the detector. As a results, these methods are all supervised.

A straightforward approach with outlier exposure spares an extra rejection class to the ID classification model, and utilize the given OoD examples along with the ID data

set to train the classifier. Following this approach, Adversarial Training with informative Outlier Mining (ATOM) (Chen et al., 2021) mines informative OoD data in the given OoD data set. During the training stage, OoD training examples where the classifier is uncertain about are adaptively chosen and used for the next epoch of training. In this way, a tight decision boundary between ID and OoD can be estimated.

More advanced approaches include various methods to synthesize useful OoD examples. For instance, after proposing the baseline detector MSP, Hendrycks and Gimpel (2017) also attempt to improve detection ability by adding an auxiliary decoder and an abnormality module to the normal classifier. Some artificial examples are generated by adding various noises to the original input (e.g., white noise, brown noise, or pink noise), or blurring and rotating the inputs. Then the abnormality module sees both original ID examples and noised examples. Noised examples are labelled with the abnormal class, clean examples are labelled with the normal class, and the sigmoid is trained to output to which class an input belongs.

Lee et al. (2017) rely on the Generative Adversarial Network (GAN) (Goodfellow et al., 2014) to generate synthetic OoD examples that are close to training distribution and also simultaneously have high entropy in terms of classifier output. They show that for effective OoD detection, the generated OoD examples should cover the low-density boundary of ID examples. Then authors explicitly train a classifier with both ID and the generated OoD examples. Nonetheless, Vernekar et al. (2019) argue that generated OoD samples close to the ID boundary fail to cover the entire boundary effectively. They aim to generate more effective OoD examples using a manifold learning network (e.g., variational autoencoder) and then create an additional class for OoD in the multi-class classifier.

Exposure-based methods are not the best choice for OoD detection. Research have shown that the performance can be largely affected by the correlations between given/synthesized and real OoD examples (Shafaei et al., 2018). The training time and computational cost are also higher for exposure-based methods.

In summary, although the performance of exposure-based methods can be enhanced by the availability of OoD examples in a laboratory environment, it can also be largely affected by the correlations between given and real OoD samples.

Data-based OoD Detection Methods

Data-based OoD detection methods do not require the involvement of an explicit classifier. Unlike classifier-based methods where OoD detection performance and ID classification performance of ID data are interdependent, data-based detection methods treat OoD detection problem as a separate and independent issue. Data-based OoD detectors can be further categorized into two branches: distribution-based methods and reconstruction-based methods.

distribution-based methods

The assumption of distribution-based detectors is that OoD examples tend to be farther away from ID data, or fall into the low-density region. Hence, OoD examples can be detected by either calculating the estimated density or distance between the new input and the ID data. The first stage is to extract a compact representation of the data features. Various tools can be used, for instance, different layers of the latent space of neural network models. Although methods in this category are often used in conjunction with a neural network model, the underlying models can be either non-classification models or classification ones. They only serve the role of a feature extractor.

Lee et al. (2018) propose a simple framework which models the penultimate layer output of deep neural networks with class-conditional Gaussian distributions. A score for OoD detection is obtained by calculating the Mahalanobis distance between the testing example and the closest class-conditional Gaussian distribution. Input perturbation technique is also adopted here to enlarge the difference between OoD and ID examples.

Instead of using the penultimate layer output directly, Abdelzad et al. (2019) propose to find the optimal output layer of a pre-trained deep neural network based on a holdout validation OoD dataset. Then they apply one-class classification algorithm to separate ID sample and OoD examples. However, the results are heavily dependent on what synthetic or real OoD examples are used for layer selection.

Later, instead of training a unified OoD detector at a fixed ending layer, Wang et al. (2022) train multiple one-class classification-based OoD detectors simultaneously at different intermediate layers to exploit the full-spectrum characteristics encoded at varying depths of deep neural networks. Consequently, the computational cost for this method would be relatively high.

Erdil et al. (2020) propose an unsupervised OOD detection method using kernel density estimation (KDE), which is a non-parametric method for estimating probability density functions of features obtained at different layers of deep neural networks. Specifically, they estimate the probability density functions of ID features from each channel of the network by performing KDE. At testing time, the probability functions are evaluated on the testing data to obtain a score, which is expected to be higher for ID and lower for OOD samples.

The Density of States Estimator (DoSE) (Morningstar et al., 2021) also employs a non-parametric density estimator to measure the so-called density of states on several summary statistics of the ID data such as the negative log likelihood loss and classification loss. Test examples with low support under the observed densities of the measurements are regarded to be OoD.

Deep one-class classification algorithms can also be directly used for OoD detection. They simply treat various ID semantic classes as a whole (Ruff et al., 2018; Perera and Patel, 2019). Deep features are first extracted from a pre-trained deep model based on the given training examples, then various fine-tuning and model building strategies can be carried out to improve detection performance. Some extra OoD data sets may be

required to improved detection performance.

Autoencoders (AE) that seek to learn a compressed representation of the inputs can also be used as feature extraction tools. Then similar tests can be applied to assess the discrepancy between the projections of newly arrived test examples and the ID dataset within the latent space (Sarafijanovic-Djukic and Davis, 2019; Andrews et al., 2016; Xu et al., 2015; Zong et al., 2018; Guo et al., 2018). For instance, Sarafijanovic-Djukic and Davis (2019) build a convolutional autoencoder (CAE) to extract a low-dimensional representation of the images, then distances within the latent space is used as an anomaly score. Guo et al. (2018) first extract the compressed hidden layer vectors of the data, and then explore the usage of K nearest neighbours distance to detect abnormal data.

In summary, feature extraction with pre-trained neural network classification models can be simple and quick. However, the selection of the appropriate extraction tool may still benefit from external information related to OoD data. In addition, when the number of dimensions of extracted features is high, which is a typical issue of data in the open world setting, these approaches can suffer from the curse of dimensionality.

reconstruction-based methods

Different from distribution-based methods, reconstruction-based methods rely on the reconstruction error or probability provided by generative models for detection directly. Reconstruction-based methods are based on the assumption that when trained with ID examples only, the generative models are only expected to capture representative features of the ID examples well. Hence, the restoration of testing examples is usually used as a criterion to make a detection. Some of the methods in this category are also single-class anomaly detection and outlier detection methods.

Autoencoder (AE) models and their variations have been used extensively in reconstruction-based methods. Unlike distribution-based methods built with AEs that utilize the deep feature representation ability of the models, methods in this category

focus on the reconstruction aspect of the models. An and Cho (2015) adopt a variational autoencoder (VAE) (Kingma and Welling, 2013) as the base model and utilize reconstruction probability to detect OoD. OoD examples are expected to have low probability density. Semi-supervised VAE (SSVAE) is a more advanced VAE for semi-supervised learning scenario (Berkhahn et al., 2019). The authors supplement the classification loss with the VAE loss so that the performance for both ID classification and OoD detection can be improved. However, VAEs have their own limitations such as some parametric prior assumptions on the latent space. Moreover, there are various ways of how the reconstruction likelihood can be formulated, which may also affect detection performance. Furthermore, many recent work are challenging the use of reconstruction likelihood of flow-based generative models such as VAE for OoD detection because extensive experiments have shown that it is not a reliable metric as expected (Nalisnick et al., 2018; Huang et al., 2019).

On the other hand, the reconstruction error of AEs is a more straight-forward metric to use. However, there are also other challenges experienced by them. Denouden et al. (2018) notice that AEs can sometimes reconstruct the semantic OoD examples with less error than ID examples. To solve this issue, they adjust the reconstruction-based detection criterion by adding the Mahalanobis distance between the test example and the training set mean within the AE latent space. Memory AE (MemAE) (Gong et al., 2019) is another recently proposed method aiming to improve detection performance of reconstruction error-based methods. It incorporates within the training stage a memory module to store prototypical elements of the ID data. Hence, the reconstruction of any test examples will be forced to be more similar to the most representative ID examples. Thus, the reconstruction error will be enlarged for OoD examples.

Although AE-based methods have been shown to be a promising tool for OoD detection, they are mostly tested in very restricted scenarios only. For instance, the VAE-based detector (An and Cho, 2015), Mahalanobis-based AE detector (Denouden et al., 2018), and MemAE (Gong et al., 2019) have all been tested in scenarios where

the ID data set contains images from one semantic class only. The performance with multi-class ID data sets remains unknown.

In addition to AE and its variations, deep generative models such as PixelCNN (Oord et al., 2016) and Glow (Kingma and Dhariwal, 2018) are also popular reconstruction tools. An empirical study has also shown that PixellCNN and Glow are better candidates for OoD detection tasks among various deep generative models (Xiao et al., 2020). More recent detectors are based on these models. While directly estimating the likelihood of such models is an intuitively natural approach, some work have also shown that higher likelihoods are falsely assigned to certain types of OoD examples. One possible reason is that deep generative models overly rely on the background information to estimate the likelihood. Ren et al. (2019) propose to concurrently train a background model with perturbed inputs, and use the likelihood ratio from the usual model along with that from the background model for OoD detection. In this way, the bias from the background pixels can be removed. Cai and Li (2023) propose a novel Frequency-Regularized Learning (FRL) framework for OoD detection, which incorporates high-frequency information into training and guides the model to focus on semantically relevant features. Nonetheless, these methods have only been tested on non-semantic data sets, and the computational cost for training deep generative models is much higher than that for AE models.

Reconstruction-based approaches are receiving increasing attention since they are naturally unsupervised, i.e., they do not require any assumptions on the OoD data or input preprocessing techniques, and models like AEs can be built with low computational costs.

2.2.2 Performance Evaluation Metrics for OoD detection

OoD detection assigns a score to each testing example. Then the decision can be made according to a specific threshold. It can also be regarded as a two-class problem. Treating OoD examples as the negative class, the performance can be summarized by a confusion

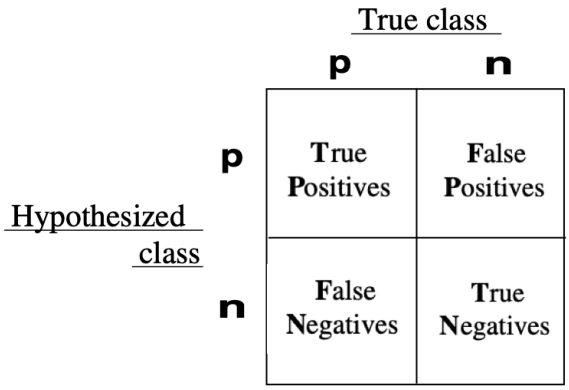


Figure 2.4: Confusion matrix for bi-class classification problem (Fawcett, 2006).

matrix as in Figure 2.4. Each entry in the confusion matrix can be defined as follow:
TP: the number of correctly classified examples belonging to the ID class;
TN: the number of correctly classified examples belonging to the OoD class;
FP: the number of misclassified examples belonging to the ID class;
FN: the number of misclassified examples belonging to the OoD class.

Based on these measures, true positive rate (TPR) (also known as Recall), false-positive rate (FPR) and precision can be further calculated as in Equations 2.3, 2.4 and 2.5.

$$TPR\ (Recall) = \frac{TP}{TP + FN} \tag{2.3}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

However, choosing the specific threshold is often regarded as a separate task that depends on the particular application and expectation of the users. In order to view the trade-off of performance achieved by various thresholds rather than just one that was chosen by the modelling technique, receiver operating characteristic (ROC) analysis (Fawcett, 2006) and precision-recall curve (PRC) analysis (Boyd et al., 2013) are often used. True positive rate (TPR) and false-positive rate (FPR) are used to build the ROC curve. Precision and recall are used to build the PRC curve. The area under ROC (AUROC) and area under PRC (AUPRC) are two single number summaries of the overall

classification performance.

Consequently, a random positive example detector corresponds to an AUROC of 50% and an AUPRC equal to the fraction of positive examples in the testing data set. For both metrics, a higher value indicates better detection performance. A "perfect" OoD detector corresponds to 100% AUROC and AUPRC.

2.2.3 Concept Drift Detection Methods

Similar to OoD detection methods, concept drift detection methods can also be categorized into classifier-based ones and data-based ones. As discussed in Section 1.1, concept drifts are commonly categorized as being real or virtual, depending on if the decision boundary $P(Y|\mathbf{X})$ is affected. Classifier-based concept drift detection methods aims to detect real drifts by monitoring $P(Y|\mathbf{X})$ directly. A drift is reported when the classification performance constantly deteriorates. Data-based methods mainly inspect changes in $P(\mathbf{X})$, which are more effective for capturing virtual drifts, but can also capture some of the real drifts simultaneously.

One of the major differences from OoD detection methods is that for concept drift detection, it is often assumed that the label space of supervised data stream remains balanced and unchanged over time. A test-then-validate scenario is considered, meaning that the respective label is revealed once the predictive label is produced. In this setting, it is more important to detect changes in the underlying environment instead of the

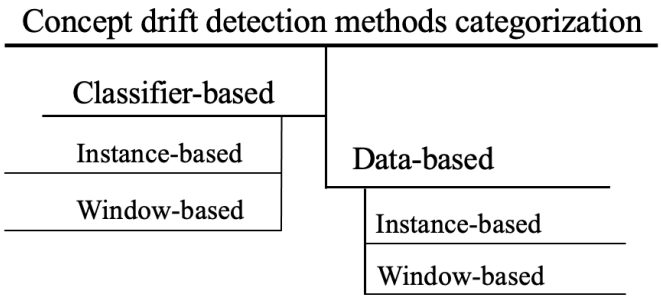


Figure 2.5: Categorization of existing concept drift detection methods.

emergence of new classes. Changes in $P(Y)$ can be considered as a particular type of concept drift, which is often referred to as concept evolution. For relevant literature dealing with concept drift in $P(Y)$, please see Antwi et al., 2012; Wang et al., 2013; Brzezinski and Stefanowski, 2014; Wang and Abraham, 2015. This thesis focuses on the detection of real and virtual drifts assuming $P(Y)$ is fixed.

Depending on the way data are processed and the application-specific memory requirement, methods can be further categorized as being instance-based and window-based as shown in Figure 2.5. Instance-based methods monitor a statistic extracted from the data stream sequentially in a one-pass learning manner. The test statistic used for detection is updated each time a single instance is received and no previous data need to be saved. Window-based methods, on the other hand, slide one or more windows along the data stream for inspection. Multiple instances within the window are used together to estimate the test statistic. The performance of window-based methods can be heavily dependent on the window size. The trade-off between detection delay and detection accuracy also needs to be carefully examined.

Classifier-based Concept Drift Detection Methods

Most existing work tackling drift in supervised data streams are classifier-based methods. They consider real drift to be the most detrimental to classification performance and focus on detection of such drifts only (Wang et al., 2017; Gama et al., 2014). As the name suggests, classifier-based methods are built upon the associated streaming data classification model. Then they monitor over time some classification performance-related indicators to decide if a drift has occurred (Gama et al., 2004; Baena-García et al., 2006; Ross et al., 2012; Harel et al., 2014; Wang and Abraham, 2015).

Assuming the true class label is revealed once the model prediction is made, an univariate stream of the error of the base classifier is generated. The label prediction can be correct ($\hat{y} = 1$) or incorrect ($\hat{y} = 0$). The prediction error of the learning algorithm

is considered as a random variable from a sequence of Bernoulli trials. The Binomial distribution gives the general form of the probability of observing an error. Based on the assumption of data independence, various statistical tests can be applied on the prediction error rate for detection. As the number of examples increases, the Binomial distribution can also be closely approximated by a Gaussian distribution, allowing a wider range of tests to be applied. Classifier-based methods are built upon the Probably Approximately Correct (PAC) learning model premise, which suggests the error rate will decrease as the number of analysed examples increases as long as the data distribution is stationary, until the generalization error rate is reached (Gama et al., 2004). Therefore, a significant increase in the classification error indicates the existence of a drift.

Instance-based methods detect changes in a single-pass through the data. The benchmark detection algorithm within this category is Drift Detection Method (DDM) (Gama et al., 2004). DDM detects abrupt drift by inspecting if the error rate exceeds a calculated threshold. It is also the first algorithm that defines a two-thresholds setting with a warning level and a drift level. After the classification error exceeds certain threshold specified by the warning level, newly arrived data will be temporarily saved. The drift is confirmed only if the error continues to rise and reaches the threshold defined by the drift level. Then data temporarily stored after warning will be used for detector reconfiguration. The thresholds for the two levels are also updated as new data are received.

This two-threshold setting is inherited by many later work. Early Drift Detection Method (EDDM) (Baena-García et al., 2006) follows a similar strategy. It improves the detection performance of DDM on gradual drifts where the change takes place at a lower rate. It monitors the distance between consecutive classification errors. EDDM is initialized and reconfigured when a minimum of 30 classification errors have taken place.

Reactive DDM (RDDM) (Barros et al., 2017) notices that the performance of DDM worsens when the concepts are very large because the accumulated error rate tend to become less sensitive. Hence, it improves DDM by adding an explicit mechanism to

discard older instances of very long concepts on a regular basis to alleviate the performance loss problem. RDDM also improves the accuracy of DDM in data streams with gradual drifts. However, it also moderately increases the number of false positives and memory consumption.

EWMA for concept drift detection (ECDD) (Ross et al., 2012) is also based on the two-threshold setting. It is adapted from Exponentially Weighted Moving Average (EWMA) control charts (Yeh et al., 2008). Instead of requiring to know the mean and variance of the error stream in advance as in EMWA, ECDD proposes a way to estimate the parameters from the data and the thresholds are set with polynomial approximations.

In addition to simple classification errors, combined error statistics can also be monitored to detect drifts. Linear Four Rates (LFR) (Wang and Abraham, 2015) monitors all four statistics associated with a confusion matrix to better exploit the error information. LFR can achieve early detection of concept drifts, high detection rate and low false alarm rate than DDM regardless of the distribution properties of the labels.

Window-based approaches also exist. For these methods, two test statistics, one from a longer frame and one from a more recent frame, are calculated and compared to each other. Window-based methods are generally based on the assumption that the accuracy of classification models should stay steady, or improve, as more instances are processed for a stationary concept. Statistical Test of Equal Proportions Detection (STEPD) (Nishida and Yamauchi, 2007) is a popular two-time window-based drift detection algorithm. STEPD detects a drift when the short term classification error is significantly higher than that of the overall window of the current concept. Fast Hoeffding Drift Detection Method (FHDDM) (Pesaranghader and Viktor, 2016), McDiarmid Drift Detection Method (MDDM) (Pesaranghader et al., 2018), and Wilcoxon Rank Sum Test Drift detector (WSTD) (Barros et al., 2018) work in a similar fashion with various modifications.

FHDDM (Pesaranghader and Viktor, 2016) compares the accuracy of the current

sliding window with the best historical accuracy observed so far. A drift is reported when a significant difference between these two values is observed according to a threshold derived from the Hoeffding's inequality (Hoeffding, 1994). MDDM (Pesaranghader et al., 2018) adopts an instance-weighting strategy. It compares the weighted mean classification accuracy of the current sliding window with the maximum weighted mean accuracy observed so far. Different weighting strategies can be applied to capture drifts of various speeds. WSTD (Barros et al., 2018) aims to restrict the high false alarm rate of STEPD by substituting the equal proportion test with Wilcoxon rank sum test (Wilcoxon, 1992) and limiting the length of the overall window. Experiments have shown that WSTD effectively reduces false alarms and is also more sensitive in detecting abrupt drifts.

In these methods, the window size is a predefined parameter. A small window can better reflect the latest data distribution and allow quicker detection at the cost of a higher computational burden. In contrast, a larger window may include examples from previous concept and delay the detection time. However, longer window also provides more data for better detector reconfiguration.

Differently, ADaptive WINdowing (ADWIN) (Bifet and Gavalda, 2007) is another benchmark method that avoids the problem of choosing the appropriate window size by incorporating an adaptive windowing strategy. ADWIN first maintains a relatively large window and examines all possible cuts to divide the large window into two sub-windows, representing old and new data. When the difference between the average values of these sub-windows is higher than a given threshold, drift is detected and the older window is dropped. The newer window is used for detector reconfiguration.

Similar to ADWIN, Hoeffding's inequality based Drift Detection Method (HDDM) (Frias-Blanco et al., 2014) is also based on a cut-point estimation procedure. HDDM does not assume the errors to follow a Bernoulli distribution. Nonetheless, it requires only independent, univariate and bounded random variables to be received, which allows the application of Hoeffding inequalities to obtain theoretical guarantees for the detection rate of various types of drifts. Two versions of HDDM are introduced by the authors to

deal with abrupt and gradual drifts with different weighting schemes.

In summary, classifier-based detectors can be used in conjunction with any classifier since they utilize only the error stream. This property allows them to be smoothly combined with any existing classification algorithms. Besides, the memory consumption of these methods is low. Even for window-based approaches where some past information may need to be stored, the memory required is still low since only the error stream needs to be saved.

However, in order to acquire the error stream, instant feedback regarding whether each prediction is correct is a necessary condition. Besides, their detection performance is heavily dependent on the chosen base classifier (Benenson et al., 2013). Various base classifiers lead to very different detection outcomes. Moreover, they neglect drifts that are not reflected on classification performance. This can cause two consequences. Firstly, this may harm the interpretation of the data and the nature of the drift. Secondly, in some situations, especially when a series of drifts is present, methods that can detect drifts before they cause performance degradation are preferred. Classifier-based methods, however, can only detect drifts after the classification performance deteriorates.

In addition, classifier-based methods regard the prediction error as a random variable following Bernoulli distribution and assume the error stream contains independent draws from the Binomial distribution. When temporal dependency exists, the prediction error stream also contains dependency (Zliobaite et al., 2015). Hence, the premise of many statistical tests is violated. Zliobaite et al. (2015) also demonstrate that when temporal dependency exists, classification error is no longer a suitable metric for assessing the performance of drift detectors because false alarms may actually decrease classification error, resulting in misleading results. Since temporal dependency commonly exists in real-world applications, it is crucial to relax the assumption of data independence for existing detectors.

Data-based Concept Drift Detection Methods

In contrast to classifier-based detection methods, data-based detection methods do not rely on an explicit classifier to make decisions. They extract and monitor some data distribution-related characteristics directly (Page, 1954; Dasu et al., 2006; Alippi et al., 2010b; Ditzler and Polikar, 2011; Lu et al., 2014; Bu et al., 2017).

Instance-based detection methods that monitor data features first stemmed from the application area of stochastic process control (SPC) and have a longer history than classifier-based methods. Control chart is generally used in industries to monitor the statistical quality of product data. The earliest control charts are parametric schemes. They assume the data to be $i.i.d$ and follow certain distributions such as the Gaussian distribution. One of the earliest control charts is the Shewhart X bar control chart (Shewhart, 1931) established in 1931 that directly monitors shifts in feature mean. Later in 1954, Cumulative Sum (CUSUM) chart (Page, 1954) is proposed to monitor the accumulated sum of deviations from the mean so that it can better detect smaller drifts. Afterwards, EWMA (Roberts, 1959) is proposed in 1959, which is similar to CUSUM but the calculation of the accumulated mean is incorporated with a predefined forgetting factor so that more weights can be assigned to the most recent data. For control charts, decisions are made based on a set of pre-defined upper and lower control limits. Time-varying control limits can also be used to improve the detection sensitivity for drifts, especially in early stages (Steiner, 1999).

Although parametric control charts are easy to use and can be very powerful when data is known to follow certain distributions, the development of non-parametric instance-based detection methods has become increasingly popular since reality is mostly pdf-free. Non-parametric methods make no distributional assumptions on the data. Computational intelligence CUSUM (CI-CUSUM) (Alippi and Roveri, 2008) is a pdf-free version of the CUSUM chart that incorporates a parameter configuration phase that allows for automatic configuration of the test parameters. The Intersection of Confidence Intervals

Change Detection Test (ICI-based CDT) (Alippi et al., 2010b) is also a non-parametric test relying on the central limit theorem for feature extraction. The ICI test carefully designs mean and variance-related features that follow a Gaussian distribution. Instead of the values themselves, the confidence intervals of these estimated values are calculated and examined. If there are no more intersections in the confidence intervals of the newly arrived and previous data, a drift is detected. The ICI-based CDT is endowed with a refinement procedure. Once a change is detected. An estimated drift starting time is then provided to help detector reconfiguration, ICI-based CDT has been shown to achieve higher detection accuracy than CI-CUSUM (Alippi et al., 2016). Although non-parametric tests do not have assumptions on the distribution of the data, the *independently-distributed* assumption still exists.

The above detection methods are designed for univariate data streams. They can be adapted to multivariate data streams by conducting univariate CDTs for each individual dimension (Alippi et al., 2010b; Alippi et al., 2016; Faithfull et al., 2019). Then a drift is reported whenever one of the dimensions raises an alarm. In order to reduce the high computational cost of dimension-wise detection methods, some researchers propose to apply principal component analysis (PCA) (Abdi and Williams, 2010) for feature extraction prior to the change detection (Alippi and Roveri, 2008; Qahtan et al., 2015). However, such approaches still tend to be problematic for high-dimensional data streams (Harel et al., 2014; Wang and Abraham, 2015). The inherent information contained within multiple dimensions is also lost.

In contrast, window-based approaches can better handle multivariate data streams by estimating and comparing the empirical density of two windows (Dasu et al., 2006; Bu et al., 2017). Window-based methods are mostly non-parametric. Different from classifier-based methods that maintain two overlapping windows, data-based methods usually treat an initial fixed window as the reference window that represents past information and keep a sliding window summarizing the most recent information. Then two-sample tests based on features extracted from these two windows are conducted.

For instance, the method proposed by Dasu et al. (2006) utilizes Kdq-tree to partition the historical and new data, estimate their empirical distributions and rely on Kullback-Leibler distance estimation for detection of potential drifts. Kdq-tree can also identify sub-regions with of the data that drifted the most based on the leaf nodes. Least squares density difference (LSDD) (Bu et al., 2017) relies on KDE methods to estimate the pre-change and the post-change pdfs and examine their difference for detection. Both methods are based on a bootstrap procedure to generate enough data to estimate the distribution of density difference on stationary concepts, which can be very time-consuming and computationally expensive, especially when the data dimension is high. So far, concept drift detection methods for high dimensional data streams are very scarce.

In summary, data-based methods may become problematic when applied on multivariate data streams with higher dimensionalities. For methods inspecting individual dimensions, high-dimensional data stream increases the possibility of false alarms. For methods estimating the data distribution, the computational burden of sequential density estimation can also be very high. Another main shortage of data-based methods is that they do not consider any label information. Thus, they cannot detect real drifts affecting the data labelling mechanisms only (e.g., a class swap) (Sobolewski and Wozniak, 2013). In addition, most of the methods monitor the overall input space, hence they tend to be insensitive to subtle drifts.

Temporal dependency can also be a potential issue for data-based detectors. For many of them, it is explicitly stated in their problem formulations that only independent random observations are to be received (Harel et al., 2014; Dong et al., 2017; Bu et al., 2016). Even if this is not explicitly stated, many algorithms relying on various statistical tests have this inherent assumption. For instance, the ICI-test (Alippi et al., 2010b) rely on the central limit theorem to conduct statistical testing, which also assumes data independence. For methods relying on PCA for feature extraction (Qahtan et al., 2015), it has also been shown that autocorrelation structure has a spurious impact on the eigenvalues of PCA (Zamprogno et al., 2020). Hence the extracted PCA components may

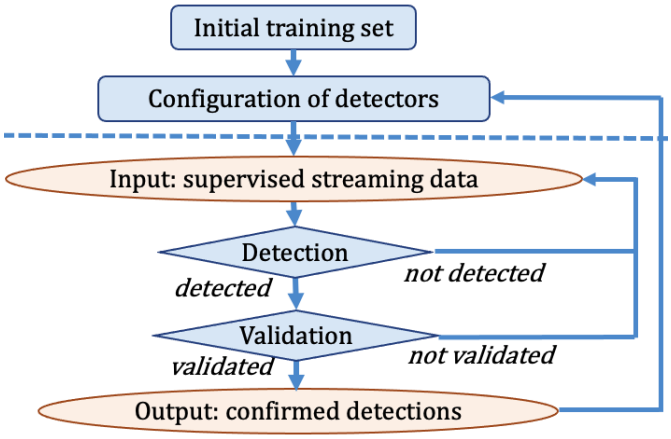


Figure 2.6: General framework of HCDT (Alippi et al., 2017).

no longer well summarize the data information. It should be noted that the existence of temporal dependency within the data streams may not necessarily hinder the performance of existing detectors in the detection of all possible types of drifts, but undoubtedly impedes the reliability of them since there will be no guarantees for optimality of the results. How well they can perform under various drift scenarios is still an issue worth investigating.

Consolidated Detection Frameworks

Based on the vast scope of individual detectors existing in the literature, more consolidated frameworks have been developed recently. For instance, hierarchical drift detection methods with multiple verification schema containing multiple layers of tests.

The first hierarchical test in the literature is the Hierarchical Change Detection Test (HCDT) presented in Figure 2.6. It is a two-layered detect-and-verify detection framework (Alippi et al., 2017). HCDT incorporates in Layer-I a simple non-parametric online detector such as the CI-CUSUM or ICI-based CDT, and in Layer-II a two-sample test such as the Hotelling T2 test (Härdle and Simar, 2007). Once a potential drift is reported in Layer-I, Layer-II is activated to compare the training set with the most recent set so as to confirm (or deny) the validity of the suspected drift. HCDT has been shown to achieve more advantageous performance than its single CDT counterpart, but it has only

been tested on non-labelled scalar data (Alippi et al., 2017). Direct application of this framework to multivariate supervised data streams still suffers from the aforementioned deficiencies of data-based detectors.

Inspired by this framework, another hierarchical framework named Hierarchical Linear Four Rate (HLFR) for supervised data streams is proposed (Yu et al., 2019). HLFR incorporates LFR (Wang and Abraham, 2015) as the base detector in Layer-I and a permutation test (Good, 2013) in Layer-II. The validation test is carried out only after a pre-defined number of examples are received following the detection. In other words, there is an unavoidable delay in drift confirmation. Even though, it has been shown to surpass the performance of many existing classifier-based detectors in terms of both detection accuracy and efficiency. HLFR is a purely classifier-based detector, hence it fails to detect virtual drifts that do not affect the decision boundary.

Another drift detection method based on Information Value and JACcard similarity (IV-Jac) (Zhang et al., 2017) is a more comprehensive detector consisting of three layers of detection tests. It addresses the label drift in $P(Y)$, feature drift in $P(\mathbf{X})$, and the decision boundary drift in $P(Y|\mathbf{X})$ with three sequential layers. With this setup, it can detect both real and virtual drifts. It extracts the Weight of Evidence (WoE) and Information Value (IV) from the available data and then detects whether a significant change exists between two consecutive windows. However, this algorithm tackles the challenge of sparseness and high dimensionality of text data streams only. It is more suitable for data streams with discrete or categorical features.

In summary, hierarchical detection frameworks have been shown to achieve high drift detection rate with lower false alarms and stronger generalization ability than those single-indicator-based methods. Existing work based on hierarchical structure have demonstrated a promising direction of research, although additional tests may bring higher computational costs.

2.2.4 Performance Evaluation Metrics for Concept Drift Detection

The performance of concept drift detectors is often evaluated from two perspectives: detection performance and the overall classification performance.

The former examines how well the detector can capture the drifts. With synthetic data streams where the ground truth of drifts is known, the detection results are easily compared with the ground truth. This can also be viewed as a bi-class problem. By treating a drift as the positive class, a confusion matrix (Figure 2.4) can be formed. However, different from OoD detection tasks, concept drift detection is associated with a time lag. Hence, it may not be so obvious to calculate each component within the confusion matrix.

Various definitions have been created for TP, FP, TN and FN in the literature reflecting various foci of the detectors. Some authors focus on if a detection is raised on a drifted sequence, instead of the number of detections raised (Alippi et al., 2011a; Kim and Park, 2017). Some authors pay attention to whether there are redundant detections after the first correct detection, but neglect the false alarms before the drifting point (Lu et al., 2014; Gu et al., 2016). A comprehensive summary of different performance evaluators used in the literature can be found in Appendix I.

Drift detector with the highest TP, the lowest FP and the lowest FN is preferred. Recall and precision are also commonly used, since they offer more insight into the ability of the model by elaborating its class-wise performance. They are defined in Equations 2.3 and 2.5 in Section 2.2.2. F-measure (also known as F-Score) is the harmonic mean of precision and recall values (Equation 2.6).

$$F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.6}$$

The definition of the evaluators TP, FP, TN and FN can heavily impact the performance analysis. There is a need for a standardized and complete definition paradigm

for detector performance evaluation. This research gap will also be addressed in Chapter 4 of this thesis.

Another way to assess detection performance is to calculate and compare the detection delay, which measures how fast the TP detection is made after its occurrence. While the quicker the detection the better, in many real-world applications there is often an acceptable delay length involved. According to Pesaranghader and Viktor (2016), an acceptable delay length is a threshold set to determine how far the detected drift could be from the true location of drift, for being considered as TP. Detection outside this range is defined to be an FP since it is considered to be too late and inaccurate.

Classification performance assessment is also commonly used to evaluate the performance of detectors in online scenarios, especially for real-world data. For real-world data streams where the ground truth of drifts is unknown, the detection performance cannot be measured directly. The most widely used metrics under this case are the number of detection and the overall classification error (or accuracy) rate. Prequential classification error is commonly used for streaming data. The prequential error at timestamp t is defined as $E_t = \frac{S_t}{B_t} = \frac{\mathcal{L}(y_t,\hat{y}_t)+\lambda S_{t-1}}{1+\lambda B_{t-1}}$ where $\mathcal{L}(y_t, \hat{y}_t)$ is the 0-1 classification loss function, $S_1 = \mathcal{L}(y_1, \hat{y}_1)$, $B_1 = 1$ and λ is a decay factor usually set to 0.999 (Gama et al., 2009). In essence, it presents the accumulated classification error rate at each time stamp with a forgetting factor. The number of detections, on the other hand, is positively related to the reconfiguration cost. Therefore, detectors leading to the highest accuracy rate at the cost of the lowest number of detections are preferred.

2.3 Chapter Summary

This chapter starts by revisiting the problem definitions for both OoD detection and concept drift detection. It has been shown that the two problems are closely related since they both concentrate on the detection of changes in classification tasks. However, the

foci are different. Concept drift detection can also be regarded as an escalated issue than OoD detection since the standard for building the detection algorithms in online scenario is also higher.

Next, existing literature for both problems is reviewed. Despite the great efforts to design various OoD detectors, most existing work still concentrate on supervised approaches. The key weakness of such methods is that in reality, the information regarding OoD data is unforeseeable. Synthesizing artificial OoD data may help to improve the detection for some types of OoD, but they can also be misleading, hence harm the generalization ability of the detectors. Therefore, the applicability of these methods in the real-world is limited. Reconstruction-based methods are the ones that are naturally unsupervised. In previous work, they have also demonstrated their good potential in OoD detection. Nonetheless, many of such methods have only been tested with single-class ID data. Their robustness against various types of OoD data also needs to be improved. For instance, their performance on semantic OoDs can be very unstable. Therefore, this thesis aims to tackle these issues and propose a purely unsupervised approach that is both reliable and robust against different types of OoDs in Chapter 3.

Different from OoD detection which endeavours to detect changes in single instances, concept drift detection aims to detect changes in the streaming environment over time. A large body of detection methods have been proposed to explicitly mark out the drifts. Nonetheless, current methods cannot well address different types of drifts simultaneously regardless of their effects on classification. Besides, existing algorithms are mainly designed for univariate or low-dimensional data streams. Since multivariate data streams are more likely to be encountered in real-world applications, this thesis proposes a new efficient detection framework to tackle these problem in Chapter 4.

After summarizing existing detection methods, it can also be noted that another potential issue that has been long neglected by existing concept drift detection work is the threat of temporal dependence. The violation of the *independently-distributed* assumption of most existing detectors may harm their reliability as well as their detection

performance. In Chapter 5, the effect of temporal dependency on existing detectors is scrutinized. After gaining a better understanding of the joint issue of temporal dependence and concept drift, a new detector that can accommodate the more complex situation is proposed.

Chapter Three

Unsupervised Out-of-distribution Detection

In the previous chapter, we introduced various existing approaches tackling OoD detection. This chapter embodies our main research findings regarding the issues in OoD detection, aiming to answer the first set of research questions proposed in Chapter 1 [1].

The rest of this chapter is organised as follows. Section 3.1 lists the motivation and novel contributions of this chapter. Section 3.2 introduces the backbone model of our algorithm as well as the related notations. Sections 3.3 and 3.4 discuss when and why existing reconstruction-based OoD detectors may fail, based on which we propose two purely unsupervised novel OoD detectors that are capable of detecting a wider range of OoDs accurately. The effectiveness of the proposed OoD detectorsis demonstrated experimentally in Section 3.5.

[1] This chapter also forms the following published research paper: Zhang, S., Pan, C., Song, L., Wu, X., Hu, Z., Pei, K., Tino, P., & Yao, X. (2021). Label-Assisted Memory Autoencoder for Unsupervised Out-of-Distribution Detection. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 795-810).

3.1 Introduction

As introduced in Chapter 2, many OoD detectors have been developed lately (Liang et al., 2017; Lee et al., 2017; Hendrycks et al., 2018; Lee et al., 2018; Ren et al., 2019). But a large body of them are supervised approaches. That is, they require the availability of some real or synthetic OoD data to tune the hyperparameters of the detectors, which is considered to be unrealistic as OoD data are not typically accessible in advance.

Reconstruction-based methods relying on generative models such as Autoencoder (AE) (Rumelhart et al., 1985) and its variations are exempted from this problem when used for OoD detection. They rely on the assumption that when trained with ID data only, an AE network produces higher reconstruction error for unseen OoD data than ID data.

Many AE-based OoD detectors have been proposed based on this assumption (Gong et al., 2019; Tuluptceva et al., 2020). However, this assumption may be violated in some scenarios. Observations have demonstrated that its validity depends on the specific characteristics of OoD examples. Sometimes AE-based OoD detector can "generalize" so well that it can also reconstruct OoD data with low reconstruction error, causing unsatisfactory detection performance (Denouden et al., 2018; Gong et al., 2019).

In addition, so far most AE-based detectors have only been tested in scenarios where the ID training data set consists of data from a single-class only. However, the difficulty in identifying semantic OoD examples does not only exist in the one-class setting. When the training dataset contains multiple classes instead of only one, which is of more practical use in the real-world, empirical studies of state-of-the-art (SOTA) AE-based OoD detectors reveal an even larger deterioration of detection performance, showing an urgent need for better solutions dealing with such learning scenario.

In addition, there has been no systematic study characterizing different types of OoD aiming for analysing the cause of performance degradation of AE-based detectors.

The only categorization of OoD is presented by Hsu et al. (2020), which is based on the semantics of ID and OoD examples. The study concluded that detection difficulty would increase when OoD examples possess the same semantic meaning as the ID examples, which coincides with our findings. Nonetheless, based on our preliminary investigation on the detection performance of various types of OoD examples, we noticed that inherent image complexity may be another factor causing OoD performance degradation. As an extreme example, a constant image (i.e., with same-valued pixels) that is of low complexity can always be reconstructed very well, even if it has never been seen during the training stage. We further noticed that most AE-based methods suffer in such a scenario. Therefore, a more thorough OoD characterization is preferred, which can not only allow us to scrutinize the reason behind the performance variation, but also help researchers to provide more targeted solutions.

To summarize, the contributions of this chapter are two OoD detectors, namely LAMAE (Label-Assisted Memory AutoEncoder) and LAMAE+, to address the above issues. Besides, a new criterion to characterize OoD scenarios is also proposed.

3.2 Memory Autoencoder

Our OoD detectors are built upon MemAE (Gong et al., 2019), which was introduced in Chapter 2. This section explains MemAE and the relevant notations in more details. MemAE endorses a memory component into the traditional AE architecture as shown in Figure 3.1. The encoder $f_e(\cdot)$ maps an input image $\mathbf{x} \in \mathcal{X}$ to a latent space $\mathcal{Z} = \mathcal{R}^C$ via $\mathbf{z} = f_e(\mathbf{x}; \theta_e)$, where θ_e represents the encoder-specific model parameter. Before the latent vector $\mathbf{z}$ is forwarded to the decoder, the memory module $\mathbf{M} \in \mathcal{R}^{N \times C}$ containing N prototypical vectors $\mathbf{m}_i$, each of dimension $1 \times C$, is put in place, where N is a predefined parameter for the memory size. $\mathbf{M}$ is designed to record the prototypical normal patterns of ID data D_{in}, which is updated at each epoch in the training phase. Once a new training example is received, cosine similarity between the encoded vector $\mathbf{z}$

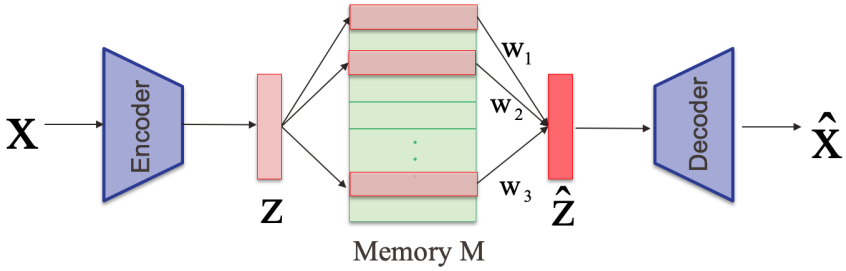


Figure 3.1: Framework of MemAE (Gong et al., 2019).

and each memory item $\mathbf{m}_i$ is calculated as $d(\mathbf{z}, \mathbf{m_i}) = \frac{\mathbf{z}\mathbf{m_i}^T}{||\mathbf{z}||\cdot||\mathbf{m_i}||}$ for $\forall\ i = \{1, \cdots, N\}$. The weight vector $\mathbf{w} = [w_i, \cdots, w_N]$ is calculated via a softmax operation $w_i = \frac{exp(d(\mathbf{z},\mathbf{m_i}))}{\sum_{j=1}^{N} exp(d(\mathbf{z},\mathbf{m_j}))}$ with $\sum_{i=1}^{N} w_i = 1$. To further limit the reconstruction ability for OoD examples, MemAE applied a hard shrinkage technique on $\mathbf{w}$, promoting the sparsity of model parameters.

The latent representation fed to the decoder is then $\hat{\mathbf{z}} = \mathbf{wM} = \sum_{i=1}^{N} w_i \mathbf{m_i}$. The reconstructed image is $\hat{\mathbf{x}} = f_d(\hat{\mathbf{z}}; \theta_d)$ where θ_d represents the decoder-specific model parameter. The MemAE loss function considers two terms: the reconstruction loss and an entropy for promoting the sparsity of $\mathbf{w}$. The network is trained with back-propagation and gradient descent. During training, the items $\mathbf{m_i}$ in memory M is updated with each pass of training and only the gradients for the items with non-zero weights w_i can be non-zero.

Once the training stage is complete, the memory M is fixed. In the testing phase, all examples are forced to be constructed with prototypical components of the ID data, resulting in significant reconstruction errors for OoDs.

3.3 Label-Assisted Memory AutoEncoder

This section explains the novel OoD detector, namely Label-Assisted Memory AutoEncoder (LAMAE). The key idea of this detector is to leverage the information of the class-labels of ID data so that the reconstruction of OoD data is constrained while the reconstruction capability for ID data can be retained. Hence, differentiation between ID

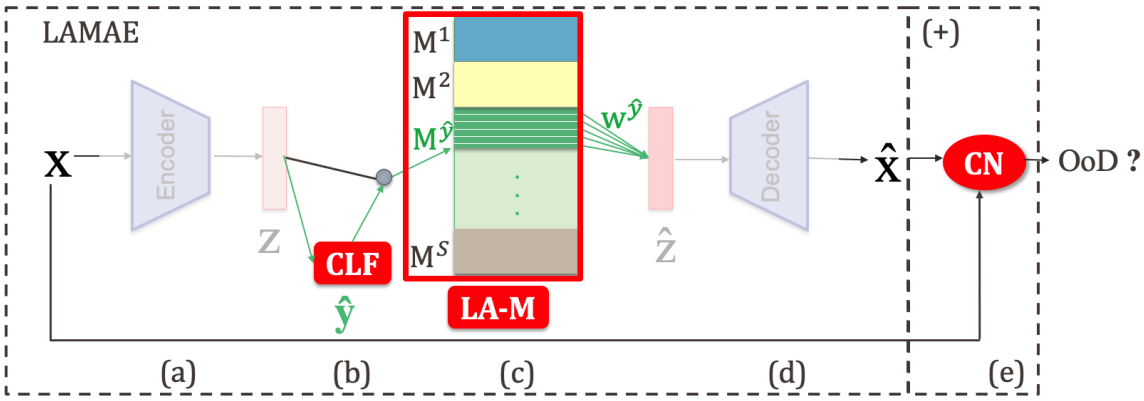


Figure 3.2: Framework of our proposed LAMAE and LAMAE+. `CLF` denotes the classifier module (section 3.3.1). `LA-M` denotes the label-assisted memory (section 3.3.2). `CN` denotes a normalizer to refine the reconstruction (section 3.4).

and OoD examples can be promoted. Figure 3.2 shows the detector architecture. Source codes of our proposed algorithms are available at DOI:10.5281/zenodo.7947400.

3.3.1 Classifier Module

For offline classification tasks, the ID data always contains multiple semantic classes. However, as mentioned earlier, existing AE-based methods have mostly been tested in scenarios where the ID data set contains only one single class and OoD data set contains multiple classes (An and Cho, 2015; Denouden et al., 2018; Gong et al., 2019). When the ID dataset contains multiple semantic classes, the information learnt by the AEs becomes more diverse. Our preliminary experiment shows that in such more practical scenarios, the performance of existing AE-based detectors can deteriorate significantly. Figure 3.3 provides an illustrative example based on the MNIST data set of handwritten digits (LeCun, 1998), where digit "7" is OoD and the rest nine digits are ID. We can see that MemAE can reconstruct both ID and OOD examples very well (Figure 3.3), such that it would be difficult to identify the OoD examples based on reconstruction error. Figure 3.4(a) shows the histograms of the reconstruction errors of ID and OoD data, further confirming the difficulty of achieving good OoD performance by using MemAE

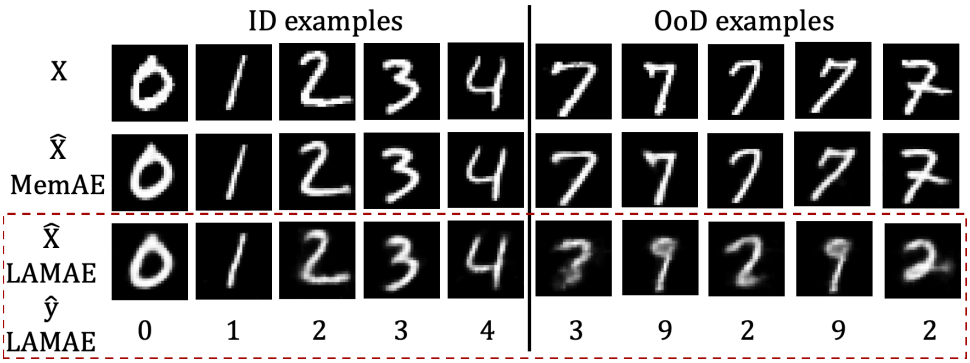


Figure 3.3: Original and reconstructed images of ID and OoD for MemAE and LAMAE. MemAE can reconstruct OoD images as good as the ID images, leading to a difficulty in OoD differentiation. LAMAE, on the other hand, reconstructs ID images much better than OoD images, allowing for a better differentiation between ID and OoD images by examining the reconstruction error.

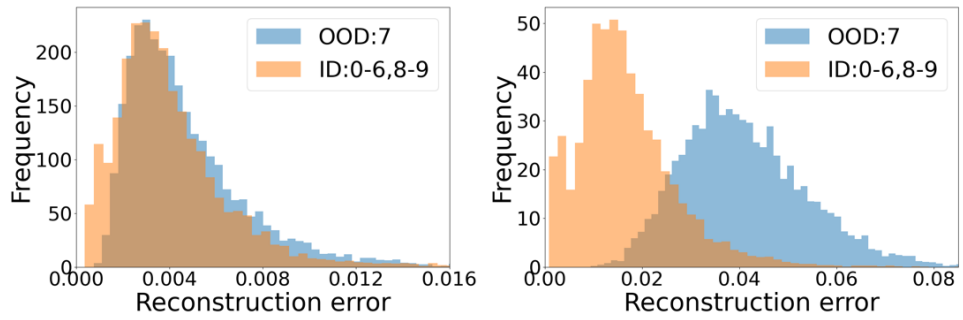


Figure 3.4: Density of reconstruction error of ID and OoD for MemAE and LAMAE. LAMAE leads to a better differentiation between ID and OoD images.

under this circumstance.

A potential reason is that the latent space learned from the multi-class ID dataset allows for a combination of features from various ID classes to reconstruct unseen OoD examples. This combination may not have much effect on the reconstruction of ID examples, but can be detrimental for OoD detection since the reconstruction error is no

longer distinguishable. To tackle this issue, we propose to regulate the reconstruction of test images by exploiting their class labels, which is implemented by placing a classifier (`CLF`) and a label-assisted memory (`LA-M`) in the AE framework as shown in Figure 3.2. The two modules are explained in Section 3.3.1 and Section 3.3.2, respectively.

A classifier $f_c(\cdot)$ is incorporated into the MemAE architecture by connecting the latent space $\mathcal{Z}$ and memory $\mathbf{M}$ as shown in Figure 3.2. $f_c(\cdot)$ can be a single-layered or multi-layered network, depending on the complexity of the application. The predicted label $\hat{y} = f_c(\mathbf{z}; \theta_c)$ of a training image $\mathbf{x}$, where θ_c denotes the classifier parameter, can then be used to guide the learning of the label-assisted memory. Given a test image, the latent representation induced by the encoder is forwarded to the classifier for the predicted label. We can see from the dotted box in Figure 3.3 that the classifier (trained purely on ID data) always assigns the OoD example with one of the existing ID labels.

3.3.2 Label-assisted Memory Module

The Label-Assisted Memory module (`LA-M`) aims to record the most representative prototypical patterns for each individual ID class. Therefore, the whole memory $\mathbf{M}$ of size N is divide into S mutually exclusive class-conditional memory chunks $\{\mathbf{M}^s | s = 1, \cdots, S\}$ where S is the number of ID classes, i.e., $\mathbf{M} = \cup_{s=1}^{S} \mathbf{M}^s$. We use N^s and $\mathbf{w}^s$ to denote the size and associated weight vector of $\mathbf{M}^s$, respectively. This study assigns the same size for $\mathbf{M}^s$, i.e., $N^1 = \cdots = N^S$.

Similar to MemAE, cosine similarity is used to calculate the weights (see section 3.2). However, thanks to the information from $f_c(\cdot)$, the latent feature $\mathbf{z}$ is only compared with each memory item $\{\mathbf{m}_i^{\hat{y}} \in \mathbf{M}^{\hat{y}} | i = 1, \cdots, N^{\hat{y}}\}$. The associated weight $\mathbf{w}^{\hat{y}}$ is calculated based on the similarity of the memory items and $\mathbf{z}$. Only $\mathbf{M}^{\hat{y}}$ and $\mathbf{w}^{\hat{y}}$ are used to formulate $\hat{\mathbf{z}}$ as

$$\hat{\mathbf{z}} = \mathbb{1}_{s=\hat{y}} \sum_{i=1}^{N^s} w_i^s \mathbf{m}_i^s. \tag{3.1}$$

Note that weight vector $\mathbf{w}$ is rather sparse since $\mathbf{w}^s = 0 \; \forall \; s \neq \hat{y}$, so we do not need to

apply the hard shrinkage technique to further increase the sparseness of $\mathbf{w}$ manually as in MemAE.

In the testing phase, $\mathbf{M}$ is fixed. Given a test image, one employs the classifier to predict its label, so that only the predicted-class-conditional memory chunk is referred to construct the latent feature according to Equation. (3.1), which is then decoded to obtain the reconstruction error. Modules (c) and (d) in Figure 3.2 demonstrate this process.

3.3.3 Training Objective

The loss function is formulated as the sum of reconstruction error and classification error on the training data as

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}[R(\mathbf{x}^t, \hat{\mathbf{x}}^t) + \beta L(y^t, \hat{y}^t|\mathbf{x}^t)], \tag{3.2}$$

where T is the training set size, β is a tuning parameter, $R(\mathbf{x}^t, \hat{\mathbf{x}}^t) = ||\mathbf{x}^t - \hat{\mathbf{x}}^t||_2^2$ is the mean-squared reconstruction error, and $L(y^t, \hat{y}^t|\mathbf{x}^t) = -\sum_{s=1}^{S} \mathbb{1}_{\hat{y}^t=s} log(p(y^t = s))$ is cross entropy classification error. In this work, softmax activation is adopted. Although the ranges of two loss terms are quite different and preliminary experiments have shown that tuning the parameter β can lead to better detection performance, the value of the most optimal β is very task-specific. During training, the loss component that can lead to the highest amount of total loss reduction is always addressed first. Preliminary results have shown that setting β to 1 can achieve sufficiently good results in most scenarios considered in our experiments. Hence, β is set to 1 for all scenarios. Finding the most optimal parameter can be one possible direction for future work. In the training process, ID examples, along with their true labels, are used to minimize the overall loss during which the predictive performance of the classifier module is also guaranteed.

3.4 LAMAE with Complexity Normalizer

This section describes the detector LAMAE+, a refined adaptation of LAMAE to further improve detection performance.

Further exploration into our experiments shows that although LAMAE generally achieves better performance than other AE-based methods, it may still fail on some specific types of semantic OoDs. For instance, the detection performance is 26% lower when digit "1" is taken as OoD compared with the case when "0" or "2" is taken. In fact, almost all existing AE-based detectors suffer in such a scenario. This section aims to investigate the reason why the detection of some types of OoD is of greater difficulty. After that, we propose a new way to categorize OoD examples. Finally, we design an image complexity-based metric (module (e) in Figure 3.2) to upgrade LAMAE, inducing LAMAE+.

3.4.1 Image Reconstruction and Complexity

According to the taxonomy with respect to data semantics (Hsu et al., 2020), our experimental setting based on handwritten digits belongs to the *semantic* OoD scenario. Nevertheless, our experimental results show that the detection performances can still vary by a large extent when different digit is treated as OoD, suggesting that it is not adequate to explain the performance variation purely from the perspective of semantics.

Based on this, we hypothesize that the inherent complexity of the image is positively correlated to its reconstruction difficulty, which impacts detection performance. To test this hypothesis, we train and test two vanilla AEs on two datasets with very different complexities: handwritten digits MNIST (LeCun, 1998) with low complexity and natural images CIFAR10 (Krizhevsky and Hinton, 2009) with much high complexity. We adopt Shannon entropy (Shannon, 1948; Tsai et al., 2008), which has a well-established

information-theoretic basis, to measure the image complexity as

$$H(S) = -\sum_{S_i=0}^{n-1} p(S_i)log(p(S_i)), \tag{3.3}$$

where n is the number of grey levels, S_i are the grey level pixel values contained in image S and $p(S_i)$ is the probability of pixel having level S_i. A Pearson correlation of 0.7952 between image entropy and reconstruction error is derived from a total of 20,000 test images (10,000 from each dataset), suggesting a strong positive correlation. Our hypothesis has been verified.

3.4.2 A Novel Characterization for OoD Data

Experimentally, we found that image complexity played an important role in OoD detection. Hence, we propose to further characterize OoD datasets by comparing the mean image complexity of the OoD datasets with the mean complexity of the ID dataset. When the OoD data set have a lower mean image complexity than that of the ID data set, we categorize this type of shift to be "plain". For an OoD data set with a higher image complexity than the ID data set, we categorize this type of shift to be "*fancy*". For instance, for an ID data set consisting handwritten digits (e.g., MNIST), an extreme example of a plain OoD data set could be a set of images with constant pixels. An example of a fancy OoD data set could be a set of images with completely random pixels. Altogether, OoDs can be categorized into six classes: *S+P*, *S+F*, *NS+P*, *NS+F*, *NS+S+P*, and *NS+S+F*, where *P, F, S, NS* stand for *plain*, *fancy*, *semantic* and *non-semantic* respectively.

By nature, the reconstruction for *plain* images should be easier than that of the *fancy* images, since fewer features are required for their description, leading to lower reconstruction errors. This property would probably mislead OoD detectors towards classifying *plain* images as ID even when they are actually OoD. Therefore, *semantic-and-plain (S+P)* OoD is the hardest to detect among all types of OoD and image complexity should be catered for when making OoD detection based on the criterion of reconstruction error.

3.4.3 Complexity-normalized Test Statistic

To tackle the challenge of detecting $S+P$ OoDs, we propose a new metric called Complexity Normalizer (CN) to adjust reconstruction error for detection. Indeed, the mechanism of CN can be used in combination with any AEs. When CN is equipped with LAMAE, we form LAMAE+.

To perform LAMAE+ for each test image, we calculate an entropy-based normalizer $CN = log(H(S) + 1)$ and re-scale the reconstruction error derived from LAMAE as:

$$\widehat{Err} = \frac{||\mathbf{x}^t - \hat{\mathbf{x}}^t||_2^2}{CN + \gamma}, \tag{3.4}$$

where $\gamma > 0$ is a tiny value to avoid the numerical problem of zero division (fixed as 1e-9). $\widehat{Err}$ is the correction of the reconstruction error, taking image complexity under consideration for OoD detection.

3.5 Experimental Studies

This section carries out two sets of experiments. Experiment 1 validates the proposed LAMAE+ by comparing with SOTA OoD detectors. Experiment 2 examines the effectiveness of each component in LAMAE+. Comparisons between LAMAE and LAMAE+ can also be found in Experiment 2.

3.5.1 Experimental Setup

Our experiments are based on the following benchmark datasets with each image standardized to [0,1] channel-wise.

1. **MNIST** (LeCun, 1998) contains gray-scale images of handwritten digits 0-9.

2. **Fashion MNIST** (FMNIST) (Xiao et al., 2017) contains gray-scale images of Za-

lando's article images from 10 classes including sneakers, trousers, pullover, etc.

3. **CIFAR10** (Krizhevsky and Hinton, 2009) contains natural color images from 10 classes including airplane, ship, dog, cat, etc.

4. **CelebA** (Liu et al., 2015) contains face images of 10,177 celebrities.

5. **notMNIST** (Bulatov, 2020) contains gray-scale images of English letters from A to J.

6. **Constant** contains images of plain color. All pixels of an image has the same value uniform-randomly drawn from the set $\{0, \cdots, 255\}$.

7. **Noise** contains images of uniform noise. Pixel values are independently drawn from the uniform distribution on the set $\{0, \cdots, 255\}$.

To show the generality and applicability of the proposed detectors, we conduct experiments on three different settings. Setting 1 is built with MNIST dataset. In each experiment, one class is used as the OoD class, and the rest 9 digits are seen as the ID data set. The procedure is repeated for all available classes. Within this setting of semantic OoDs, both *S+P* OoDs and *S+F* OoDs exist. Details of the ID and OoD data sets complexities can be found in Table 3.3. FMNIST and CIFAR10 are used as the ID dataset in settings 2 and 3 respectively. The OoD datasets being tested when FMNIST is taken as the ID dataset are MNIST, notMNIST, Constant and Noise. Within this scenario, MNIST and Constant are *plain* OoDs since they have lower mean complexities than FMNIST. OoD datasets notMNIST and Noise are considered to be *fancy*. When CIFAR10 is taken as the ID dataset, FMNIST, CelebA, Constant and Noise are adopted as OoD datasets. In this case, FMNIST and Constant are *plain* OoDs. CelebA and Noise are *fancy* OoDs. In summary, within these two settings representing *NS+S* shift, both *NS+S+P* and *NS+S+F* OoDs exist.

In this section we report the area under the receiver operating characteristic curve (AUROC) which plots the true positive rate (TPR) of ID against the false positive rate

(FPR) of OoD data by a varying threshold. The average detection performance and the standard deviation of 10 repetitive experiments are reported. Performance measured by area under the precision-recall curve (AUPRC) shows similar trends.

3.5.2 Experiment 1: Comparative Studies with SOTA Detectors

In this section we validate the proposed methods for the detection of various types of OoD. We compare LAMAE+ with unsupervised detectors including traditional AE, VAE (An and Cho, 2015), SSVAE (Berkhahn et al., 2019), MemAE (Gong et al., 2019) and the latest non-reconstruction-based detector GODIN (Hsu et al., 2020).

On MNIST and FMNIST, we implement the encoder using three convolution layers as in MemAE (Gong et al., 2019). For GODIN (Hsu et al., 2020) where MNIST and FMNIST are not used for training, we experimented with the same structure as the setting for the encoder-and-classifier component adopted for LAMAE+. On CIFAR10, with higher data complexity, deeper encoder and decoder are constructed for MemAE and LAMAE+. A skip connection from $\mathbf{z}$ to $\hat{\mathbf{z}}$ with dimension 16 is added to further assist reconstruction. Except for the last layer, each layer is followed by a batch normalization (BN) (Ioffe and Szegedy, 2015) and a Rectified Linear (ReLU) activation (Nair and Hinton, 2010). Batch size is set to 128 and we use an Adam optimization procedure.

For all methods being compared, the maximum number of training epochs is set to 200, 200 and 500 for MNIST, FMNIST and CIFAR10 respectively. CIFAR10 is assigned with a higher number of training epochs since it consists of natural images that are more difficult to reconstruct and classify. A 10% validation set is extracted from the ID training dataset. The best model that achieves the lowest loss on the validation dataset among all epochs is saved and used as the final model. Note that this validation dataset is still ID so the models are trained without access to any information about OoD examples. The class-conditional memory size N^s in LAMAE and LAMAE+ is set to 10, 10 and 50 for MNIST, FMNIST and CIFAR10 respectively. Later we demonstrate experimentally

Table 3.1: AUROC detection performance for MNIST in Experiment 1. Each time the model is trained on 9 of the 10 classes and the left-out class is considered to be the OoD class. **Bold** indicates the best scores.

OoD Class	AE	VAE(e)	VAE(p)	SSVAE(p)	MemAE	GODIN	LAMAE+
0	81.4±0.7	87.0±1.6	94.9±0.1	96.9±0.1	83.0±1.4	86.3±8.0	**98.5±0.2**
1	12.7±0.1	27.2±1.3	47.0±3.0	9.5±0.6	14.2±1.0	86.4±5.0	**90.1±2.5**
2	92.9±0.3	96.0±0.6	96.1±0.1	97.2±0.0	94.8±0.4	88.7±4.4	**99.0±0.1**
3	82.0±0.4	94.2±0.5	84.8±0.3	90.2±0.2	83.1±1.2	70.6±6.8	**97.7±0.3**
4	76.6±0.8	91.4±0.6	70.8±0.4	75.1±0.3	75.7±0.6	79.4±6.0	**95.5±0.6**
5	82.6±0.3	92.2±0.7	86.0±0.3	89.4±0.1	84.1±0.8	64.8±9.9	**97.5±0.3**
6	83.6±0.6	86.1±1.1	92.3±0.01	96.0±0.2	85.9±0.9	83.1±7.6	**97.0±0.5**
7	56.9±1.0	67.2±1.7	66.9±0.2	75.5±0.5	57.7±0.8	79.8±8.7	**94.7±1.1**
8	90.5±0.3	95.3±0.5	89.1±0.4	92.2±0.2	90.1±0.6	85.8±3.8	**97.5±0.3**
9	59.2±0.7	67.3±0.8	62.0±2.0	67.7±0.5	56.5±0.9	79.1±9.1	**89.7±2.0**

that performance is insensitive to the selection of memory size. An extra fully connected layer with softmax output is taken as the classifier component.

Performance on Semantic OoD Only

Table 3.1 reports the results of MNIST. Results of VAE(p) and SSVAE(p) based on reconstruction probability are taken from the original papers (An and Cho, 2015; Berkhahn et al., 2019). We also report the results based on reconstruction error (VAE(e)).

We can see that LAMAE+ achieves the best AUROC in all 10 cases. The improvement is especially substantial for digits "1", "4" , "7" and "9". Detailed explanations of how exactly each component in LAMAE+ contributed to this outcome is presented later in Section 3.5.3. It is also worth noting that the standard deviation of GODIN

Table 3.2: AUROC detection performance for FMNIST and CIFAR10 in Experiment 1. **Bold** indicates the best scores.

ID	OoD	AE	VAE(e)	SSVAE(e)	MemAE	GODIN	LAMAE+
FMNIST	MNIST	96.2±0.2	99.0±0.1	98.9±0.1	97.1±0.1	79.0±4.3	**99.9±0.0**
	notMNIST	99.6±0.0	99.8±0.0	**99.9±0.0**	99.8±0.0	64.0±5.8	**99.9±0.0**
	Constant	68.1±2.2	63.2±1.7	82.0±7.0	72.3±1.1	84.4±9.6	**100.0±0.0**
	Noise	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	86.5±6.3	99.9±0.0
CIFAR10	FMNIST	71.7±0.7	69.4±0.9	84.8±1.9	**98.5±0.0**	94.2±1.7	95.0±0.6
	CelebA	55.0±0.5	58.2±0.3	60.0±1.2	70.0±0.0	**75.7±2.2**	59.5±1.0
	Constant	0.0±0.0	0.0±0.0	0.0±0.0	51.2±0.0	92.7±2.0	**100.0±0.0**
	Noise	100.0±0.0	**100.0±0.0**	**100.0±0.0**	**79.6±0.0**	91.0±8.8	**100.0±0.0**

is much larger than that of AE reconstruction-based approaches, signifying that the reconstruction-based approaches can be generally more stable than softmax-based approaches.

Performance on Non-semantic and Semantic OoD

Table 3.2 reports the results of FMNIST and CIFAR10 where various OoD datasets are selected. VAE and SSVAE based on reconstruction likelihood have not been tested within these settings and the exact formulation of reconstruction likelihood is not provided. Hence, we report only VAE(e) and SSVAE(e) based on reconstruction error. We can see that LAMAE+ ranked the first in 5 out of the 8 cases. In particular, it is capable of detecting the OoD examples belonging to the Constant dataset better than the other methods, demonstrating its effectiveness in identifying the *plain* OoDs.

The performance is not as good when CIFAR10 is used as the training dataset. This may be due to the fact that the network structure is not deep enough to account for the complicated details within the CIFAR10 dataset. In addition, a single classifier layer may also be inadequate for this dataset. For instance, the backbone classifier used

Table 3.3: AUROC detection performance for MNIST in Experiment 2. The second and third columns list the average image complexity of each ID dataset and OoD dataset on 1000 images measured by Equation. (3.3). Digits "1", "4", "7" and "9" are *plain* OoDs. **Bold** indicates the best scores among each subgroup.

		Detector	1	2	3	4	5	6
OOD	ID Comp.	OoD Comp.	AE	AE+	MemAE	MemAE+	LAMAE	LAMAE+
0	1.57±0.38	1.91±0.27	**81.4±0.7**	78.3±0.8	**83.0±1.4**	79.9±1.6	**98.8±0.1**	98.5±0.2
1	1.69±0.32	0.94±0.19	12.7±0.1	**27.0±0.3**	14.2±1.0	**32.6±1.8**	72.8±4.2	**90.1±2.5**
2	1.58±0.39	1.76±0.29	**92.9±0.3**	**92.9±0.3**	**94.8±0.4**	**94.8±0.4**	98.9±0.1	**99.0±0.1**
3	1.59±0.39	1.74±0.30	**82.0±0.4**	**82.0±0.4**	**83.1±1.2**	**83.1±1.3**	97.5±0.4	**97.7±0.3**
4	1.61±0.39	1.55±0.25	76.6±0.8	**79.2±0.7**	75.7±0.6	**78.7±0.7**	93.9±0.8	**95.5±0.6**
5	1.60±0.39	1.67±0.29	82.6±0.3	**83.1±0.3**	84.1±0.8	**84.6±0.9**	97.2±0.3	**97.5±0.3**
6	1.59±0.39	1.68±0.29	83.6±0.6	**84.3±0.6**	85.9±0.9	**86.6±0.9**	96.5±0.5	**97.0±0.5**
7	1.62±0.39	1.42±0.24	56.9±1.0	**62.1±1.0**	57.5±0.8	**63.4±0.8**	91.7±1.5	**94.7±1.1**
8	1.58±0.38	1.86±0.30	**90.5±0.3**	89.1±0.4	**90.1±0.6**	88.5±0.7	**98.0±0.3**	97.5±0.3
9	1.60±0.38	1.58±0.26	59.2±0.7	**60.9±0.7**	56.5±0.9	**58.2±1.0**	87.4±2.1	**89.8±2.0**

by GODIN is Resnet-34 (Hsu et al., 2020). Increasing the complexity of network may lead to improvements in detection performance at a cost of an increasing computational burden.

3.5.3 Experiment 2: Analysis of LAMAE+

In this section, we analyse the functionality of each component in LAMAE+. We experimentally demonstrate that the combination of the classifier module, label-assisted memory and CN-adjusted test statistic helps the detector to achieve better results on the most difficult OoD type, i.e., *S+P* OoDs, with the MNIST dataset.

Effect of the Classifier and the Label-Assisted Memory

To illustrate the effectiveness of the classifier and the label-assisted memory, we compare the results for AE, MemAE and LAMAE on the MNIST dataset in Table 3.3 (Detectors 1, 3 and 5).

We can see that our results for AE and MemAE confirmed the benefit of establishing a memory component as discussed in MemAE (Gong et al., 2019), for which MemAE achieved higher AUROC than AE in 7 out of the 10 cases. Furthermore, in all 10 cases, LAMAE achieved a significant improvement in AUROC when compared with both MemAE and AE, demonstrating the dominant advantage of using a classifier and a label-assisted memory for detecting *semantic* OoDs when the ID dataset contains multiple classes. Nonetheless, for OoD digits "1", "4", "7" and "9" whose image complexity measured with Shannon entropy (Equation. 3.3) ranked the lowest four, there is still a gap when compared with the rest cases. We will address this issue with *plain* OoDs with the CN component in the following section.

Effect of CN-Adjustment

This section demonstrates that the CN-adjustment can further improve the detection performance, especially for the most difficult OoD type *S+P*. As discussed earlier, CN can be used with any AE reconstruction-based OoD detectors and an improvement in detection performance can be anticipated. We verify this conjecture experimentally by taking AE, MemAE and LAMAE as the base detectors and modify only the reconstruction error-based test statistic. We rename the CN-adjusted detectors by suffixing "+". Results are presented in Table 3.3.

It can be noted that in 8 of the 10 cases, using a CN-adjusted test statistic indeed leads to a significant improvement in detection performance for the digits "1" , "4" , "7" and "9" , which are the hardest ones to detect among all digits (Berkhahn et al., 2019)

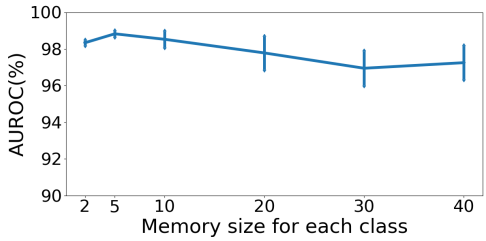


Figure 3.5: Sensitivity of detection performance to class-conditional memory size on MNIST when digit "0" is held as OoD. Similar trends can be observed for others.

and can be characterized as $S{+}P$ by us. This is true for various types of AEs. For digits "0" and "8" , CN caused a slight decrease in AUROC. This is due to the fact that these two digits are already the most complex 2 with the highest entropy values. Regarding this, we suggest that better complexity measurements may be created in the future so that the detection performance on slightly more complicated images can be maintained.

Examining the overall average performance, we conclude that the improvement in detection performance is attributed to the combination of the classifier component, the label-assisted memory and the complexity normalizer.

Sensitivity to Memory Size

This section provides a sensitivity analysis of the detection performance for LAMAE. We present the performance under different memory size settings for the MNIST experiment. Figure 3.5 suggests that LAMAE is robust to different memory sizes and for simple datasets such as MNIST, even a small memory size can achieve satisfactory performance.

3.6 Chapter Summary

In this chapter, we proposed LAMAE, a novel AE-based OoD detector with a label-assisted memory, and its refined variation LAMAE+. Both methods are purely unsupervised that do not require any assumptions on the OoD examples. Specifically, we injected

a classifier and a class-conditional memory into the traditional AE network architecture to avoid combination of features from different ID classes and thus, constrain the reconstruction of OoD examples while retaining the generalization on ID examples. LAMAE not only performs well on the non-semantic and semantic OoDs, it also significantly improves the detection performance of semantic OoD examples. We also proposed a new way to characterize OoD based on image complexity and a new metric, CN, to eliminate the bias associated with the reconstruction error induced by inherent image complexity. Thereby, the refined detector LAMAE+ is capable of detecting the most difficult type of OoD that previous work cannot handle well.

It is also worth pointing out that in the experimental set up of this work, all comparative methods are implemented with the same or at least similar network structure to realize a fair comparison. It is possible that the selected parameters may not be the most optimal ones that lead to the best results can be obtained by each method. Nonetheless, since there is no a priori knowledge regarding test data sets in our problem setting, it is hard to conduct parameter tuning for each method.

In the current work, we only used the basic Shannon entropy to measure image complexity. Many other image complexity measures also exist (Gao et al., 2018). Some measures may focus on the spatial characteristics of the images (e.g., Yu and Winkler, 2013) while some others may capture the redundancy within the images (e.g., Van Geert and Wagemans, 2017). Will the detection performance be affected if other image complexity measures are utilized? If so, how and why different complexity measures affect OoD detection performance? Besides, different functions for constructing the normalizer shall also be considered and investigated. Following this, a more suitable complexity-based normalizer may be constructed to improve detection performance further.

Chapter Four

Concept Drift Detection for Multivariate Data Streams

The last chapter focuses on the detection of out-of-distribution examples in offline classification tasks. From this chapter, the thesis starts to investigate the challenges in concept drift detection in online classification tasks .

This chapter[1] aims to provide a solution to answer the second set of research questions listed in Section 1.2. Section 4.1 emphasizes the motivations as well as the main contributions of this chapter. The solution to tackle the challenges, a hierarchical reduced-space drift detection framework for multivariate data streams, is introduced in Section 4.2. Each component of the framework is explained in detail. In Section 4.3, four sets of experiments are carried out on both synthetic and real-world data streams to demonstrate the superiority of the proposed detection framework in comparison with some state-of-the-art detectors, including both data distribution-based and classification performance-based ones. Section 4.4 summarizes this chapter.

[1]This chapter forms the following published research paper: Zhang, S., Tino, P., & Yao, X. (2021). Hierarchical Reduced-space Drift Detection Framework for Multivariate Supervised Data Streams. IEEE Transactions on Knowledge and Data Engineering (Vol 25, no.1, pp. 95-110.).

4.1 Introduction

Existing drift detection methods are mainly classifier-based or data-based. Most existing work tackling drifts in supervised data streams focus on real drifts only, since this type of drift is considered to be the most detrimental to classification accuracy (Wang et al., 2017). However, we consider the detection of all types of drift to be equally important for the following reasons. Firstly, even when a so-called virtual drift takes place and classification accuracy is not negatively affected, the optimal decision boundary is often likely to change. Retraining the classifier can still improve classification performance. Secondly, detection of such drifts provides insight of the underlying data streams, which can help understanding the behaviour of the data generation source. This information may also be beneficial when there is a pattern in a series of multiple drifts.

Therefore, this chapter aims to present a framework that is capable of detecting all types of drifts regardless of their effects on classification boundary. Then, practitioners can decide whether it is worth modifying the current classification model based on the specific application scenario.

Existing data-based methods either attempt to compare the estimated empirical density of two windows (Dasu et al., 2006; Bu et al., 2017) or conduct univariate CDTs for each individual dimension (Alippi et al., 2017; Faithfull et al., 2019) to detect drifts in multivariate data streams. These approaches tend to be problematic for higher-dimensional data streams (Harel et al., 2014; Wang and Abraham, 2015) due to increasing computational costs and possibility of false alarms as data dimensionality grows. Although the hierarchical detection framework HCDT (Alippi et al., 2017) has been shown to achieve very promising detection performance on scaler data streams, direct application of this framework to multivariate supervised data streams still suffers from the aforementioned deficiencies.

Motivated by these issues, this chapter proposes a novel drift detection framework that is capable of detecting both real and virtual drifts in multivariate supervised data

streams. The key idea is to leverage the knowledge from supervised information to discover changes that may not be detected by the existing detection methods. The contributions of this chapter include:

1) A new hierarchical detection framework named Hierarchical Reduced-space Drift Detection (HRDD) for supervised data streams that detects both real and virtual drifts simultaneously.

2) Compared with the existing HCDT framework proposed by Alippi et al. (2017) (Figure 2.6 in Section 2.2.3) , HRDD is more accurate and efficient in terms of a high number of true detections, while maintaining a low number of false alarms, when operating on higher-dimensional data streams.

3) For both real and virtual drifts, HRDD performs no worse, and in many cases better, than state-of-the-art detection algorithms, whether they are classifier-based or data-based, in terms of more true detections and fewer false alarms within any specified acceptable detection delay range.

4.2 Hierarchical Reduced-space Drift Detection Framework for Multivariate Supervised Data Streams

The new framework Hierarchical Reduced-space Drift Detection (HRDD) for multivariate data stream adopts the hierarchical structure introduced by HCDT (Alippi et al., 2017) but with three major novel components, which will be explained in sequel later in this section. The general outline of HRDD is presented in Figure 4.1. The algorithmic version of HRDD is presented in Algorithm 1. The novel components uniquely owned by HRDD compared with the existing HCDT framework are highlighted in green in both Figure 4.1 and Algorithm 1.

To tackle the challenges identified in Section 4.1, HRDD first builds a lower-

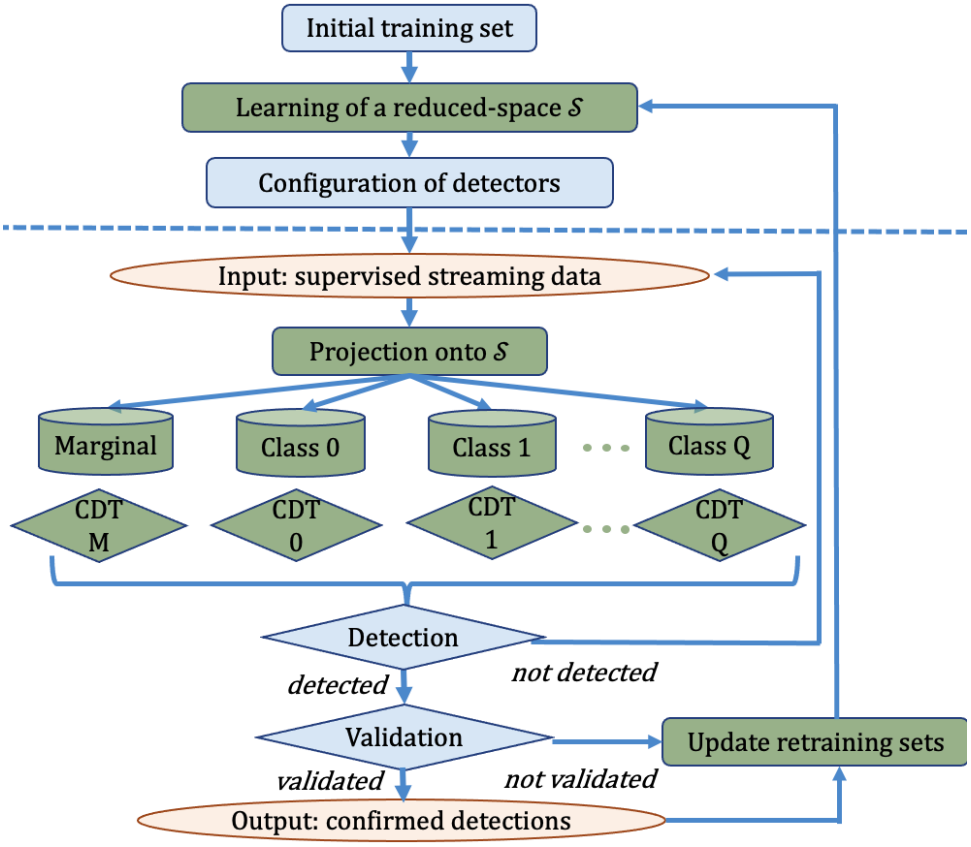


Figure 4.1: General framework of our proposed HRDD. The green-shaded boxes are three novel components that uniquely exist in HRDD. Detailed descriptions for each novel component are provided in sections 4.2.1, 4.2.2 and 4.2.3.

dimensional feature space for the given classification task using the stationary training data. Each incoming data is projected to this space upon arrival (Section 4.2.1). Next, it monitors not only the marginal distribution of the data stream, but also each individual class-conditional distribution (Section 4.2.2). Finally, a novel method to reconfigure more informative retraining datasets after each detection is also presented (Section 4.2.3).

HRDD has a high degree of flexibility and may be customized effortlessly. it can be used in conjunction with any base CDT and classifier, and the performance is independent of the choice of the classifier. Since there are no assumptions on the multivariate data streams, any detection and validation tests can be used as long as they are capable of detecting the same type of change. Although this chapter provides one possible realization for a binary classification problem as an illustrative example in the current work, it is

Algorithm 1: General framework of HRDD. The green-shaded lines represent three novel components that uniquely exist in HRDD.

input : initial training sequence TS^M for the marginal CDT and $TS^0, TS^1, ...$ for the class-conditional CDTs

output: confirmed detections

1 Find the lower-dimensional feature space $\mathcal{S}$ with TS^M;
2 Initialize the marginal and class-conditional CDTs with TS^M and $TS^0, TS^1, ...$ respectively in $\mathcal{S}$;
3 **while** *there is incoming data* **do**
4 Project data onto $\mathcal{S}$;
5 Perform concept drift detection with the marginal CDT as well as the relevant class-conditional CDT within $\mathcal{S}$;
6 **if** *a change is detected by any of the CDTs at* $\widehat{T}$ **then**
7 Estimate the potential drift starting point T_{ref};
8 Activate the validation layer on the respective stream;
9 **if** *change is validated* **then**
10 Record $\widehat{T}$ as a confirmed detection;
11 Define TS_C^M as $\{\mathbf{x}_t | t \in [T_{ref}, ..., \widehat{T}]\}$;
12 Update training sets $TS^M, TS^0, TS^1, ...$ accordingly and continue from line 1.
13 Output the confirmed changes.

worth noting that the general framework of HRDD is also suitable for multi-class data streams.

Algorithm 2: Recursive SVM (RSVM) (Tao et al., 2008)

input : training set TS^M of length l ; the desired dimension of the reduced-space R (or threshold ϵ); $r = 1$.

output: projectors $\{\mathbf{w}_r | r = 1, ..., R\} \in \mathbb{R}^{d \times R}$

1 Determine the vector $\tilde{\mathbf{w}}_1 = \sum_{i=1}^{l} \alpha_i^1 \phi(\mathbf{x}_i)$ by solving the dual optimization problem (Vapnik, 1963);

2 Let $\mathbf{w}_r = \tilde{\mathbf{w}}_r / ||\tilde{\mathbf{w}}_r||$ and generate the following training set for SVM problem by projecting the training samples onto a subspace that is orthogonal to $\mathbf{w}_r$: $\phi(\mathbf{x}_i^{r+1}) = \phi(\mathbf{x}_i^r) - \langle \phi(\mathbf{x}_i^r), \mathbf{w}_r \rangle \mathbf{w}_r$;

3 Terminate if the desired number of dimensions R has been reached (or $max\{||\phi(\mathbf{x}_i^{r+1})|| : 1 \leq i \leq l\} < \epsilon$). Otherwise, increment r by 1 and go back to line 2.

4.2.1 Learning of a Lower-dimensional Subspace

In this module, we take the information from class labels into consideration and propose a preprocessing step specifically designed for drift detection for supervised data streams. The aim of this step is to identify a lower-dimensional feature space $\mathcal{S}$ that contains the most relevant information for the given classification task. By identifying such a subspace spanned by the training samples (line 1, Algorithm 1), incoming multivariate data samples can be easily projected onto this space (line 4, Algorithm 1). Then, instead of monitoring the original input space, the detection is carried out within this reduced feature space for the particular classification task. Comparing with the existing HCDT without this step, HRDD inherently reduces the possibility of false alarms as well as the computational burden because there are fewer dimensions to examine. Meanwhile, valuable data characteristics relevant to classification are preserved.

It is worth noting that subspace selection methods have been used for change detection in signal processing applications (Blythe et al., 2012; Wu et al., 2013). However, how a change is defined in such applications is very different than that in our setting.

Consequently, the characteristics that the subspace shall possess also vary. HRDD combines the information from both original data space and the label space to identify the most appropriate subspace for concept drift detection. Besides, many subspace-based change detection algorithms for time series data make particular assumptions on their data streams (Kawahara et al., 2007; Jiao et al., 2018). For instance, in the work by Jiao et al. (2018), the data stream is assumed to follow a Gaussian distribution. HRDD does not make any assumptions on either the data stream or the underlying subspace.

As one possible realization of HRDD within a bi-class setting, we choose a recursive support vector machine (RSVM) (Tao et al., 2008) as a tool for identification of the relevant reduced-space $\mathcal{S}$. The detailed RSVM algorithm is presented in Algorithm 2, where l is the length of an initial training dataset and $\phi(\cdot)$ is the kernel function. RSVM was initially proposed for both dimensionality reduction and accuracy improvement for offline classification problems. It starts as a regular SVM (Vapnik, 1963) but can recursively derive new maximum margin features. The dimension R of the reduced-space can either be set by the practitioner a priori, or be automatically identified when $max\{||\phi(\mathbf{x}_i^{r+1})|| : 1 \leq i \leq l\} < \epsilon$. That is, when the number of components is sufficient to account for most of the differences in the classification task.

Based on an initial training set, Algorithm 2 provides us with one or several orthogonal directions $\{\mathbf{w}_r | r = 1, ..., R\}$ which can be used as projectors to an R-dimensional subspace $\mathcal{S}$. Then each newly arrived instance $\mathbf{x}_j$ can be projected to $\mathcal{S}$ as $\langle \phi(\mathbf{x}_j), \mathbf{w}_r \rangle = \sum_{i=1}^{l} \alpha_i^r \kappa(\mathbf{x}_j, \mathbf{x}_i)$ for $r = 1, ..., R$ and $i = 1, ..., l$. It is worth pointing out that all computations involved in RSVM can be based on kernel evaluation instead of the explicit $\phi(\mathbf{x}_j)$. From the second iteration, $\kappa(\mathbf{x}_i^r, \mathbf{x}_j^r)$ can be recursively computed by using the equation in step 2 of Algorithm 2 and $\kappa(\mathbf{x}_i^{r-1}, \mathbf{x}_j^{r-1})$, allowing different kernels to be adopted.

In this current work, we select $R = 1$ after some preliminary experiments. In fact, the assumption of $R = 1$ is realized by many classification models, starting from perceptrons, SVMs through to classification based on Gaussian Processes. All these models can be interpreted as imposing a single projection dimension where classification

can be performed. Since such 1-dimensional projection directions are integral part of such classification machines, they are also good candidates for 1-dimensional subspaces on which to perform statistical test regarding concept drifts. Other supervised dimensionality reduction methods may also be used. However, techniques such as PCA are inherently unsupervised, and hence do not, by definition, satisfy our requirement for a low-dimensional subspace relevant to classification.

4.2.2 Class-based Detection

While most existing detectors focus on detecting drifts by monitoring $P(Y|\mathbf{X})$ or $P(\mathbf{X})$, there has been a lack of attention paid to $P(\mathbf{X}|Y)$. Supervised information can be better utilized by class-conditional distributions because they focus on sub-regions of the input space. In HRDD, we suggest not only incorporating a distribution-based CDT to inspect data features from the perspective of marginal distribution, but also constructing one CDT for each class-conditional distribution $P(\mathbf{X}|Y = q)$, where $q \in \{0, 1, ..., Q\}$. The CDTs are initialized on its respective data stream (line 2, Algorithm 1). Note that only the marginal detector and one of the class-conditional detectors are activated each time an instance arrives.

Usually, the number of classes of a data stream is much lower than the number of dimensions. Therefore, HRDD is still expected to be computationally cheaper to implement than existing multivariate detectors that either try to estimate the distribution density or examine each dimension individually. By monitoring also the class-conditional distributions, HRDD captures both real and virtual drift, regardless of the effect on classification performance. Besides, since it synchronizes sub-regions of the input space, it is able to evaluate the effects on different classes and its detection sensitivity over smaller drifts is enhanced.

Different techniques can be chosen as the base CDT for this component. ICI-based CDT (Alippi et al., 2010b) can be used as a reference example. A dominant advantage

of this sequential CDT is that it is endowed with a refinement procedure that directly provides the estimated drift starting time T_{ref} (Alippi et al., 2010a). Thus, a new dataset representing the most recent concept is automatically identified. For other drift detectors, the method introduced by Poor (1996) is recommended to identify T_{ref}.

Comparing with IV-Jac (Zhang et al., 2017) which also monitors $P(\mathbf{X})$ and $P(\mathbf{X}|Y)$, we emphasize the following differences: a) our framework deals with continuous data features. IV-Jac cannot be directly applied to our problem setting; b) our framework can be used with various statistical CDTs and does not require prior knowledge about the drift to determine detection threshold; c) our approach works within a reduced feature subspace, hence is more robust against noise in the original data and scales better for high-dimensional data.

4.2.3 Knowledge Base Reconfiguration

Table 4.1: Construction of retraining sets after detection. Without loss of generality, we assume the last instance received belongs to Class 0. Analogous definitions can be made for Class 1. TS^M, TS^0, TS^1 are the existing training sets for the marginal, Class 0 and Class 1 detectors, respectively. TS_C^M is composed of all instances representing the current concept in $[T_{ref}, \widehat{T}]$. TS_C^0 (TS_C^1) denotes the set of Class 1 (Class 0) instances in TS_C^M.

		Layer-I and II output from the Marginal Detector		
		Detected and Validated	Detected but Invalidated	No Detection
Layer-I and II output from Class 0 Conditional Detector	Detected and Validated	M: $TS^M = TS_C^M$ C0: $TS^0 = TS_C^0$ C1: $TS^1 = [TS^1, TS_C^1]$	M: $TS^M = TS_C^M$ C0: $TS^0 = TS_C^0$ C1: $TS^1 = [TS^1, TS_C^1]$	M: $TS^M = TS_C^M$ C0: $TS^0 = TS_C^0$ C1: $TS^1 = [TS^1, TS_C^1]$
	Detected but Invalidated	M: $TS^M = TS_C^M$ C0: $TS^0 = TS_C^0$ C1: $TS^1 = TS_C^1$	M: No retraining C0: No retraining C1: $TS^1 = [TS^1, TS_C^1]$	M: $TS^M = [TS^M, TS_C^M]$ C0: No retraining C1: $TS^1 = [TS^1, TS_C^1]$
	No Detection	M: $TS^M = TS_C^M$ C0: $TS^0 = TS_C^0$ C1: $TS^1 = TS_C^1$	M: No retraining C0: $TS^0 = [TS^0, TS_C^0]$ C1: $TS^1 = [TS^1, TS_C^1]$	

Once a suspicious change is reported in the detection layer by at least one of the base detectors at time $\widehat{T}$, a potential drift starting time T_{ref} is estimated (lines 6-7, Algorithm 1). Then the validation layer is activated and an offline statistical test is used to compare the previous training set of the respective detector and instances from T_{ref} to $\widehat{T}$ to determine if the drift should be confirmed (line 8, Algorithm 1). If a drift is validated, detection time point $\widehat{T}$ is recorded. Afterwards, the existing HCDT framework discards all past data and reconfigure based on the most recent data only. This approach may be over-conservative for a supervised data stream as a drift may have uneven effects on different classes. Unnecessary rejection of data in a relatively stationary class leads to information loss, which can become problematic when available information is already scarce or expensive to obtain. Here we propose a novel and more flexible way of reconstructing the retraining sets in order to maintain as much useful information as possible for detector reconfiguration. The idea can be summarized as follows.

1) For data streams where we can confirm that a change has taken place (with a detected and validated change), the respective detectors are immediately reconfigured based on a latest dataset representing the current concept. It should be noted that when one class-conditional detector reports a validated change, it subsequently impacts the marginal distribution since $P(\mathbf{X}) = \sum_{q=0}^{Q} P(Y = q) \cdot P(\mathbf{X}|Y = q)$. Therefore in this case the marginal detector is also retrained.

2) For data streams where there is ambiguity if a change has taken place (a detected but invalidated change), we do not make any amendments to the existing detector.

3) For data streams where we are inclined to believe that no drift has taken place, all available and relevant instances are used as the new retraining set for the respective detectors. For instance, when a detection is reported by Class 0 detector but no validated detection from either the Class 1 detector or the marginal detector, we may combine the latest Class 1 instances in $[T_{ref}, \widehat{T}]$ with the previously existed Class 1 training set TS^1 to form a more informative retraining set. Hence, the performance of the detectors is expected to improve as extra relevant instances are used for retraining.

As a concrete realization under a bi-class scenario, the reconstruction scheme for all detectors after each detection can be summarized in Table 4.1. Based on the results from both the detection layer and validation layer, retraining datasets are constructed and the detectors are retrained accordingly (lines 11-12, Algorithm 1). Finally, all the validated changes are reported when there is no more data to arrive (line 13, Algorithm 1). Hotelling T2 test has been shown to be a suitable complementary validation test for ICI-based CDT in the existing HCDT framework (Alippi et al., 2017). Therefore, it is also selected in the current implementation of HRDD.

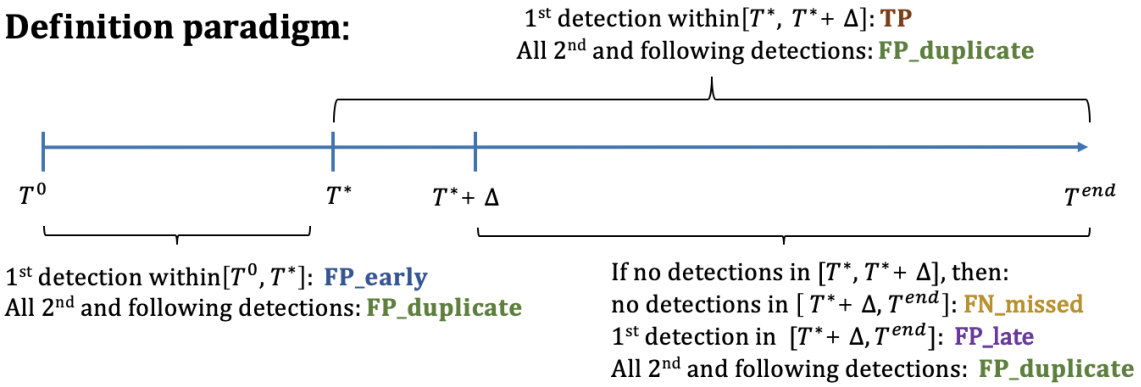
4.3 Computational Studies

This section presents four sets of experiments that evaluate the effectiveness and efficiency of HRDD. Experiment 1 aims to demonstrate the effectiveness of each component of the HRDD framework, especially when facing data streams with various dimensionalities. Experiment 2 illustrates the superiority of HRDD in drift detection on both real and virtual drifts over state-of-the-art methods. Experiment 3 validates that the superior performance provided by HRDD also benefits classification, even when integrated with a very simple classifier. Experiments 1-3 are based on datasets of synthetically generated sequences where the ground truth of drift occurrences is available. In Experiment 4, we demonstrate the role of HRDD on a real-world data stream. Finally, we provide a brief analysis on the computational time complexity of the approaches being considered in the experiments. All experiments were run on a CentOS 7.6 Computer with v4 2.20 GHz processor and 128 GB memory.

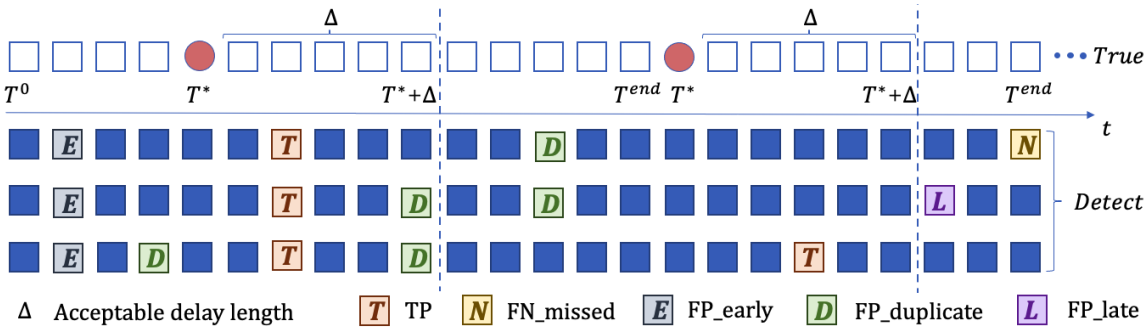
4.3.1 A New Paradigm for Performance Evaluation Metrics

A variety of performance metrics for drift detection have been used in the literature. For instance, when counting the number of True Positive (TP), False Negative (FN) and

False Positive (FP), some authors focus on if a detection is raised on a drifted sequence, instead of the number of detections raised (Alippi et al., 2011a; Kim and Park, 2017). Differently, some authors pay attention to whether there are redundant detections after a TP, and distinguish between *Detected*, *Late*, *Missed* and *False* detections based on sliding windows (Lu et al., 2014; Gu et al., 2016). False detections before the first drifting point are neglected. Differently, some other authors choose to take into account all detections raised on a stream, and each single detection is categorized into TP or FP based on a



(a)



(b)

		Predicted	
		Yes	No
Actual	Yes	TP	FN = FN_missed + FP_late
	No	FP = FP_early + FP_duplicate + FP_late	TN

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{F-measure} = \frac{2 \text{ Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

(c)

Figure 4.2: Detection performance definition paradigm.

specified window size (Harel et al., 2014). The notion of acceptable delay Δ was formally defined as a threshold set to determine how far the detected drift could be from the true location of drift (Pesaranghader and Viktor, 2016). Here, FPs are defined as detections outside of the acceptable detection interval $[T^*, T^* + \Delta]$, but extra detections within the interval are neglected.

From a practical point of view, taking into account all detections raised on a stream is important. Distinguishing between various types of false alarms also helps to make targeted modifications. Therefore, when analysing the results of a reactive detector, we propose a more realistic and comprehensive definition paradigm as in Figure 4.2a. Based on a predefined acceptable detection delay range $[T^*, T^* + \Delta]$ where T^* is the real drifting time, we define a TP as the first detection within this range, a FN_missed as a missed alarm throughout the concept. We also distinguish between three types of FPs: FP_early, FP_duplicate and FP_late. A FP_early is the first false alarm before T^* related to algorithm initialization, FP_duplicate's are redundant false alarms related to algorithm reconfiguration, and a FP_late is the first detection in $[T^* + \Delta, T^{end}]$ when there is no alarm raised in $[T^*, T^* + \Delta]$. An illustrative example is presented in Figure 4.2b.

The total number of FPs and FNs are therefore FP = FP_early + FP_duplicate + FP_late and FN = FN_late + FP_late respectively. Performance of the detector is evaluated via number of TPs, FPs, FNs or Recall, Precision and F-measure as defined in Figure 4.2c. For each synthetic dataset in the following experiments, 30 sequences are generated, and all reported figures are summations. Detection performance is measured for several acceptable lengths $\Delta = \{500, 1000, 1500, 2000\}$ so as to limit the maximum detection delay allowed.

Table 4.2: Compared detection frameworks in Experiment 1.

	detection			reconfiguration	
Algorithm	monitor $P(\mathbf{X})$	monitor $P(\mathbf{X}\|Y)$	reduced-space	single CDT	multiple CDTs
HRDD	✓	✓	✓		✓
HCDT-M (HCDT)	✓			✓	
HCDT-CC		✓		✓	
HDD	✓	✓			✓

Table 4.3: Synthetic data generation of d-dimensional hyperplane datasets.

Concept	$d-dimensional\ hyperplane$
1	$a_0^1 = -1.5; \quad a_i^1 = i \times 0.1\ \forall\ i \in \{1, ..., d\}$
2	$a_0^2 = a_0^1 - 1; \quad a_i^2 = a_i^1 - 0.5\ \forall\ i \in \{1, ..., d\}$

4.3.2 Experiment 1: Understanding HRDD

In order to better understand the novelty of HRDD relative to the existing hierarchical framework HCDT, we carry out a component-wise evaluation on data streams with varying dimensionalities. The characteristics of HRDD and several variations containing only partial components are presented in Table 4.2. HCDT-M is the existing HCDT framework which monitors the marginal input distribution only (Alippi et al., 2017). HCDT-CC is the existing HCDT framework applied to the class-conditional distributions. HDD is similar to HRDD in terms of inspection of both marginal and class-conditional distributions, but without projection to a reduced-space. All the detectors adopt ICI-based CDT as the layer I test and Hotelling T2 as the layer II test following the implementation of HCDT (Alippi et al., 2017). A set of different parameters of ICI-based CDT are considered: $\Gamma \in \{2.25, 2.5, 2.75, 3.0, 3.25\}$. A higher Γ symbolizes an decreasing level of sensitivity to drifts.

Synthetic data generated for this experiment is a set of d-dimensional moving hyperplanes $y = -a_0 + \sum_i^d a_i x_i$, $x_i \in [0, 1]$ and $y \in [0, d]$. This is a popular dataset in the field of drift detection (Minku et al., 2010; Lu et al., 2016). The generation mechanism also allows easy alteration of dataset dimensionality. To demonstrate the ability of HRDD to handle high-dimensional data, we considered $d = [5, 10, 15, 20, 30, 40]$. Data generation details can be found in Table 4.3. The data stream is balanced with 5% of class noise added. Each stream consists of $10,000$ instances with one abrupt change at timestamp 5001.

The detection results with $\Delta = 1000$ and 2000 are presented in Table 4.4. The following findings are also applicable to $\Delta = 500$ and 1500. Firstly, we notice that HRDD achieves the highest TP in almost all cases. This is true even for a tight Δ, indicating that HRDD can not only detect the drifts, but also detect them earlier than the existing HCDT and other variations being considered. Meanwhile, HRDD always reports the lowest FP. HDD, which is also based on this novel reconfiguration scheme but does not project data onto the low-dimensional space as HRDD does, always ranked second in terms of both TP and FP. In contrast, methods monitoring each dimension within the input space lead to much higher FP.

Comparing with the results of HCDT-CC, we can conclude that HRDD is very different from the existing HCDT applied on each class. The novelty of HRDD lies in not only class-based inspections, but also the projection of data onto the reduced-space, and the utilization of both marginal and class-conditional information for reconfiguration. As dimensionality increases, the superiority of HRDD becomes more dominant, confirming its ability to operate efficiently even for high-dimensional data streams. Also, comparing the performance presented in Table 4.4 horizontally, it can be seen that HRDD is relatively insensitive to the parameter of the base detector, making it a more reliable and stable approach among the compared methods.

Table 4.4: Detection performance on data streams with increasing dimensionality. Methods with high TP and low FP are preferred. Best results given the specified parameter Γ and acceptable delay length Δ are in **bold**.

	Δ =	1000											
		TP						FP					
	Γ =	2.25	2.5	2.75	3.0	3.25	3.5	2.25	2.5	2.75	3.0	3.25	3.5
5D	HRDD	**30**	**27**	**22**	**16**	11	7	**18**	**10**	**16**	**15**	19	23
	HCDT-M	5	2	3	0	0	0	31	27	27	26	21	**18**
	HCDT-CC	26	23	19	14	11	9	22	17	17	17	21	22
	HDD	23	25	19	**16**	**12**	**9**	19	11	18	16	**18**	22
10D	HRDD	**30**	**30**	28	**27**	**24**	**21**	**21**	**12**	**10**	**9**	**9**	**11**
	HCDT-M	27	27	26	21	20	18	157	66	35	13	10	17
	HCDT-CC	29	28	**29**	25	20	13	130	83	57	56	47	53
	HDD	25	28	28	25	**24**	**21**	114	81	59	21	24	15
15D	HRDD	**30**	**30**	**30**	**30**	**29**	**27**	**11**	**12**	**9**	8	**4**	**4**
	HCDT-M	23	27	28	28	28	24	344	203	83	**4**	16	6
	HCDT-CC	22	23	26	21	18	12	134	106	82	60	58	61
	HDD	21	25	28	29	28	**27**	216	176	119	40	40	13
20D	HRDD	**30**	**30**	**30**	**30**	**30**	**28**	**15**	**10**	**9**	**9**	**9**	7
	HCDT-M	19	21	24	26	28	27	405	211	123	40	18	**3**
	HCDT-CC	10	13	9	16	10	14	107	95	56	36	38	30
	HDD	17	19	18	20	22	27	238	164	96	68	58	22
30D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**31**	**28**	**28**	**22**	**12**
	HCDT-M	29	25	22	16	11	12	1052	781	506	361	242	132
	HCDT-CC	**30**	**30**	**30**	**30**	**30**	**30**	288	213	164	123	97	86
	HDD	**30**	**30**	29	29	**30**	**30**	517	342	214	155	95	76
40D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**21**	**14**	**10**	**5**	**4**	**3**
	HCDT-M	**30**	27	20	16	12	7	1304	986	697	489	305	256
	HCDT-CC	**30**	**30**	**30**	**30**	29	29	353	234	162	109	68	64
	HDD	**30**	**30**	**30**	**30**	29	29	601	407	328	217	139	69

Table 4.5: Detection performance on data streams with increasing dimensionality. Methods with high TP and low FP are preferred. Best results given the specified parameter Γ and acceptable delay length Δ are in **bold**. (Continued)

	Δ =	2000											
		TP						FP					
	Γ =	2.25	2.5	2.75	3.0	3.25	3.5	2.25	2.5	2.75	3.0	3.25	3.5
5D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	18	7	8	**1**	**0**	**0**
	HCDT-M	24	23	24	11	8	4	**12**	**6**	**6**	15	13	14
	HCDT-CC	**30**	**30**	**30**	**30**	**30**	**30**	18	10	6	**1**	2	1
	HDD	29	26	29	**30**	**30**	**30**	13	10	8	2	**0**	1
10D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**21**	**12**	**8**	**6**	3	**2**
	HCDT-M	28	29	**30**	28	29	29	156	64	31	6	**1**	6
	HCDT-CC	**30**	**30**	**30**	**30**	**30**	**30**	129	81	56	51	37	36
	HDD	27	29	**30**	29	29	**30**	112	80	57	17	19	6
15D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**11**	**12**	**9**	8	**3**	1
	HCDT-M	27	29	29	29	**30**	**30**	340	201	82	**3**	14	**0**
	HCDT-CC	25	25	**30**	28	29	29	131	104	78	53	47	44
	HDD	27	28	**30**	29	**30**	**30**	210	173	117	40	38	10
20D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**15**	**10**	**9**	**9**	**9**	5
	HCDT-M	23	23	26	26	29	**30**	401	209	121	40	17	**0**
	HCDT-CC	16	20	13	19	17	24	101	88	52	33	31	20
	HDD	23	21	21	22	25	29	232	162	93	66	55	20
30D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**29**	**31**	**28**	**28**	**22**	**12**
	HCDT-M	**30**	**30**	**30**	**30**	**30**	29	1051	776	498	347	223	115
	HCDT-CC	**30**	**30**	**30**	**30**	**30**	**30**	288	213	164	123	97	86
	HDD	**30**	**30**	**30**	**30**	**30**	**30**	517	342	213	154	95	76
40D	HRDD	**30**	**30**	**30**	**30**	**30**	**30**	**21**	**14**	**10**	**5**	**3**	**2**
	HCDT-M	**30**	29	**30**	26	23	15	1304	988	687	479	294	248
	HCDT-CC	**30**	**30**	**30**	**30**	**30**	**30**	353	234	162	109	67	63
	HDD	**30**	**30**	**30**	**30**	**30**	**30**	601	407	328	217	138	68

4.3.3 Experiment 2: Drift Detection Ability

In this section we aim to compare the drift detection ability of HRDD on a wide range of drifts with the latest hierarchical detection methods, HCDT (Alippi et al., 2017) and HLFR (Yu et al., 2019) introduced in section 2.1.2 of chapter 2. These consolidated frameworks have already been shown to perform better than their individual base detector counterparts. We also compare HRDD with two classic performance-based benchmarks, DDM (Gama et al., 2004) and EDDM (Baena-García et al., 2006), which have not been used as base change detectors in the above-mentioned frameworks. Since the detection result from performance-based detectors is contingent on the choice of classifier, two classifiers are adopted: an SVM and a decision tree. All hyper-parameters of the comparative algorithms were taken directly from the original papers. The setting of HRDD follows the experimental setting for HCDT (Experiment B in the original paper (Alippi et al., 2017)). Detection Recall, Precision, and F-measure are reported for $\Delta = \{500, 1000, 1500, 2000\}$.

In this experiment we first test on data streams with one abrupt drift only. With drift affecting $P(Y|\mathbf{X})$ or not and its magnitude being small or large, there are 4 possible scenarios for a single drift. These cases will be examined individually. Afterwards, data streams with multiple drifts are used for testing. The following synthetic datasets are generated for this experiment:

1) **4D Multivariate Gaussian** (Figure 4.3): this dataset contains sequences with one drift only. We synthetically generate drifts affecting the target concept differently by changing one class-conditional distribution independently. Possible drift scenarios are visualized in Figure 4.3. In order to reflect the 4 scenarios, 4 subsets of 4D Multivariate Gaussian streams are generated. Each data stream consists of $10,000$ observations, and a single abrupt change takes place at 5001. The magnitude of drift is controlled by the change in within-class distance d_w. The effect on the target concept is controlled by the change in between-class distance d_b. The (d_w, d_b) pair for the initial concept is always

Table 4.6: Synthetic data generation of 4D Multivariate Gaussian. The is given in Figure 4.3.

Concept	$4D\_Gaussian$			
	(a)	(b)	(c)	(d)
1	$\boldsymbol{\mu}^1_{C_0} = [0, 0, 0, 0]$ $\boldsymbol{\mu}^1_{C_1} = [0.8, 0.8, 0.8, 0.8]$ $\boldsymbol{\Sigma}^1_{C_0} = \boldsymbol{\Sigma}^1_{C_1} = \mathbb{1}_4$			
2	$\boldsymbol{\mu}^2_{C_0} = [-0.2, 0.1, -0.2, 0.1]$	$\boldsymbol{\mu}^2_{C_0} = [-0.2, -0.2, -0.2, -0.2]$	$\boldsymbol{\mu}^2_{C_0} = [0.4, -0.3, 0.4, 0.4]$	$\boldsymbol{\mu}^2_{C_0} = [-0.3, -0.4, -0.4, 0.4]$
	$\boldsymbol{\mu}^2_{C_1} = \boldsymbol{\mu}^1_{C_1}$ $\boldsymbol{\Sigma}^2_{C_0} = \boldsymbol{\Sigma}^2_{C_1} = \mathbb{1}_4 + 0.2 \times (\mathbb{J}_4 - \mathbb{1}_4)$			

Table 4.7: Synthetic data generation of 6D Multivariate Gaussian datasets. The illustration is given in Figure 4.4.

Concept	$6D\_Gaussian$	Concept	$6D\_Gaussian$
1	$\boldsymbol{\mu}^1_{C_0} = [2, 2, 3, 3, 4, 4]$ $\boldsymbol{\mu}^1_{C_1} = [1, 1, 2, 2, 3, 3]$; $\boldsymbol{\Sigma}^1_{C_0} = \boldsymbol{\Sigma}^1_{C_1} = \mathbb{1}_6$	4	$\boldsymbol{\mu}^1_{C_0} = [2.8, 2.8, 3.4, 3.4, 3.8, 3.8]$ $\boldsymbol{\mu}^4_{C_1} = \boldsymbol{\mu}^3_{C_1}$; $\boldsymbol{\Sigma}^4_{C_0} = \boldsymbol{\Sigma}^4_{C_1} = \mathbb{1}_6$
2	$\boldsymbol{\mu}^1_{C_0} = [2.6, 2.6, 3.8, 3.8, 4.2, 4.2]$ $\boldsymbol{\mu}^2_{C_1} = \boldsymbol{\mu}^1_{C_1}$; $\boldsymbol{\Sigma}^2_{C_0} = \boldsymbol{\Sigma}^2_{C_1} = \mathbb{1}_6$	5	$\boldsymbol{\mu}^1_{C_0} = [2.6, 2.6, 3.0, 3.0, 3.4, 3.4]$ $\boldsymbol{\mu}^5_{C_1} = \boldsymbol{\mu}^4_{C_1}$; $\boldsymbol{\Sigma}^5_{C_0} = \boldsymbol{\Sigma}^5_{C_1} = \mathbb{1}_6$
3	$\boldsymbol{\mu}^1_{C_0} = [2.2, 2.2, 3.4, 3.4, 4.4, 4.4]$ $\boldsymbol{\mu}^3_{C_1} = \boldsymbol{\mu}^2_{C_1}$; $\boldsymbol{\Sigma}^3_{C_0} = \boldsymbol{\Sigma}^3_{C_1} = \mathbb{1}_6$		

(0,0). For scenarios (a-d) in Figure 4.3, the (d_w, d_b) pair are set to (0.5, -0.9), (0.5, 0.8), (1.0, -0.8), and (1.0, 1.1), respectively. Data generation details can be found in Table 4.6.

2) **6D Multivariate Gaussian** (Figure 4.4): this dataset contains multiple-drift streams. We consider a scenario where a series of drifts is not detrimental to classification at the beginning, but eventually impairs the accuracy after several evolutions. A simple illustration of this situation is presented in Figure 4.4. Each sequence is of length $25,000$ and contains 5 concepts. The evolution of concept can be summarized as the (d_w, d_b) pair being (0, 0), (0.4, 2.4), (0.4, 1.9), (0.4, -1.4), and (0.4, -2.1) for each drift. Details of the data generation process can be found in Table 4.7.

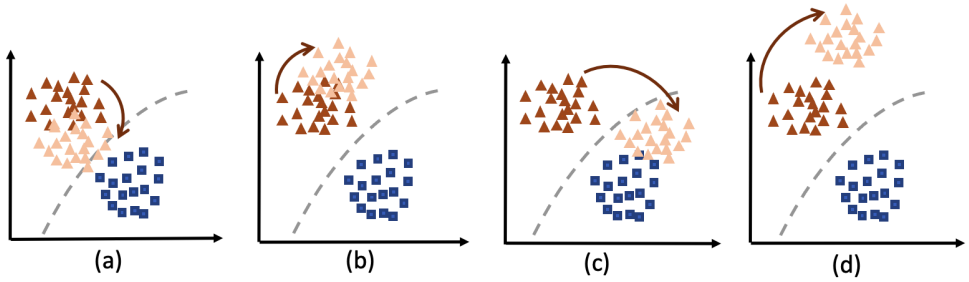


Figure 4.3: Illustration of various drift types of 4D Multivariate Gaussian. (a) small drift affecting $P(Y|\mathbf{X})$; (b) small drift not affecting $P(Y|\mathbf{X})$; (c) large drift affecting $P(Y|\mathbf{X})$; (d) large drift not affecting $P(Y|\mathbf{X})$. Data generation details are given in Table 4.6.

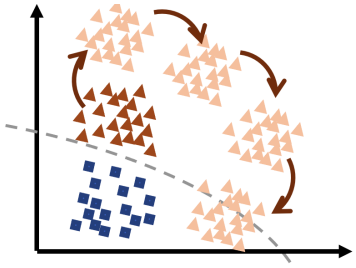


Figure 4.4: Illustration of 6D Multivariate Gaussian. Data generation details are given in Table 4.7.

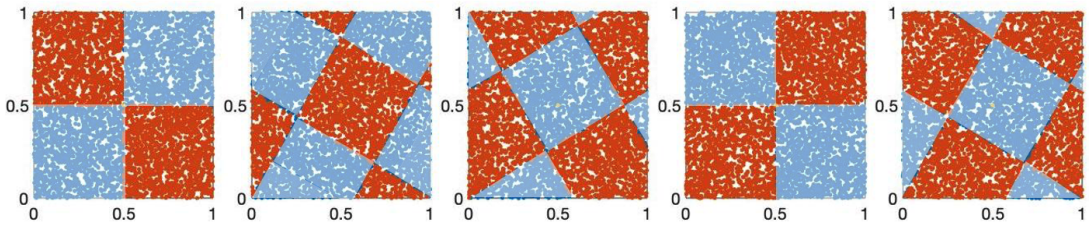
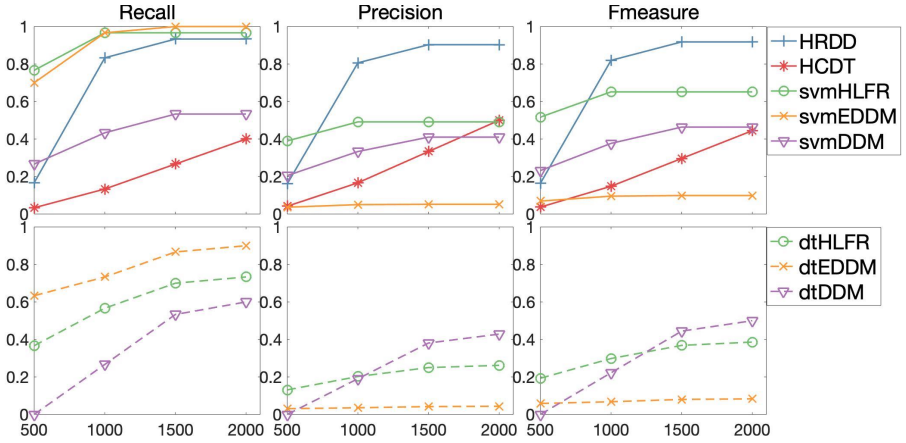


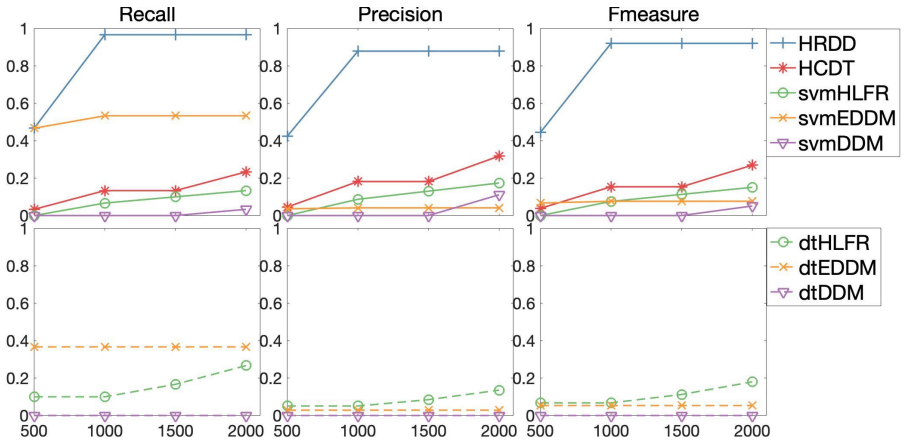
Figure 4.5: Illustration of Rotating Checkerboard.

3) **Rotating Checkerboard** (Figure 4.5): in this multiple-drift benchmark dataset (Elwell and Polikar, 2011), all 4 drifts lead to a strong change in classification boundary. Each stream is of length $25,000$ and contains 5 concepts. Examples are sampled uniformly from the unit square with a dimensionality of 2, and the labels are set by a checkerboard with 0.5 tile width. At each concept drift, the checkerboard is rotated by an angle of $\pi/6$ radians.

Detection performance for **4D Multivariate Gaussian** is summarized in Figure 4.6. Overall, HRDD ranked first in 14 out of the 16 cases (4 datasets and 4 Δ's) in

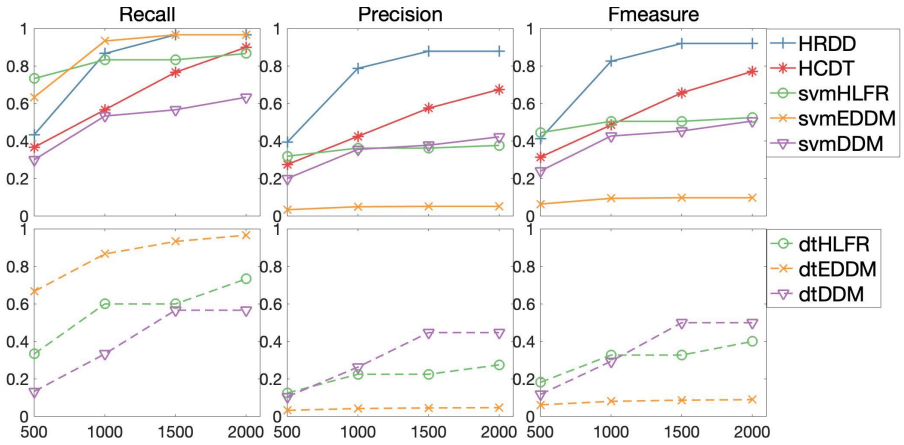


(a) Magnitude: small; Type: affecting $P(Y|\mathbf{X})$

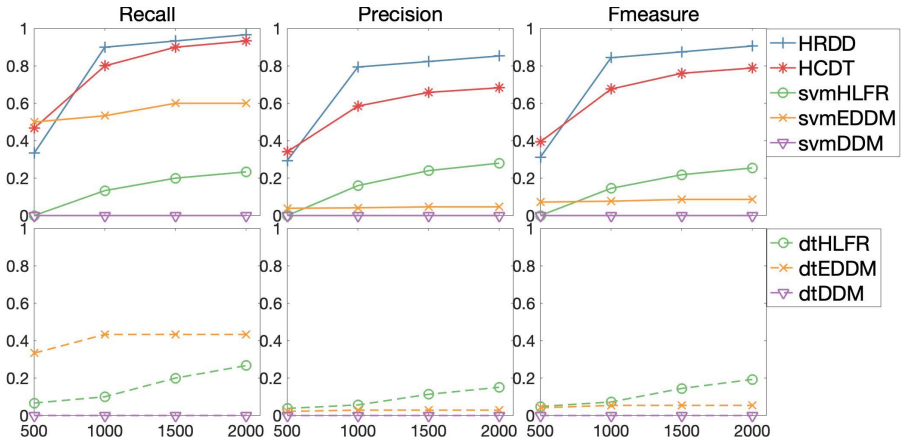


(b) Magnitude: small; Type: not affecting $P(Y|\mathbf{X})$

Figure 4.6: Detection performance for 4D Multivariate Gaussian against acceptable delay lengths. Subfigures (a-b) correspond to scenarios (a-b) in Figure 4.3, respectively.



(c) Magnitude: large; Type: affecting $P(Y|\mathbf{X})$



(d) Magnitude: large; Type: not affecting $P(Y|\mathbf{X})$

Figure 4.6: Detection performance for 4D Multivariate Gaussian against acceptable delay lengths. Subfigures (c-d) correspond to scenarios (c-d) in Figure 4.3, respectively. (Continued.)

terms of F-measure, indicating its ability to achieve the best trade-off between recall and precision. Performance-based detectors HLFR, EDDM and DDM only secure high recall values for real drifts affecting $P(Y|\mathbf{X})$, which cause an evident degradation in classification accuracy (figures 4.6a and 4.6c). For drifts not harming classification performance, i.e., drifts not affecting $P(Y|\mathbf{X})$ (figures 4.6b and 4.6d), performance-based detectors fail and the distribution-based detector HCDT becomes the second best detector after HRDD in terms of detection F-measure. In addition, HRDD also surpasses HCDT by a great amount when drift magnitude is small (figures 4.6a and 4.6b). This is due to the fact that the detection mechanism monitoring class-conditional distributions makes HRDD more sensitive to even a lightest change in the overall input space. For drifts with greater magnitude (figures 4.6c and 4.6d), the performance of HCDT improves, but it still falls behind HRDD in all but one case.

Table 4.8 presents how many times each individual detector is activated among all 30 TP detections. When a drift is caused by Class 0 only, the class-conditional distribution of Class 0 and the marginal distribution are both affected. Results in Table 4.8 shows that 28 out of the 30 drifts can be captured by the respective detector or the marginal detector, which comes in line with our expectation. For scenarios b) and d), Class 0 moves away from Class 1, leading to a relatively greater change in the marginal distribution comparing with scenarios a) and c), hence these two scenarios result in more activations of the marginal detector. This also demonstrates that the combination of both marginal and class-conditional inspections in HRDD is indeed helpful.

Moving to the multiple-drift scenarios, HRDD also outperforms its competitors in all 8 cases in terms of F-measure as shown in figures 4.7 and 4.8. For **6D Multivariate Gaussian** (Figure 4.7), since the magnitude of each single drift is relatively small, HCDT requires two or more consecutive drifts in order for the effect of the drift series to be sufficiently noticeable on the marginal distribution. Performance-based detectors HLFR, EDDM and DDM are only able to detect the last one or two drifts in Figure 4.4, since earlier drifts do not deteriorate classification performance.

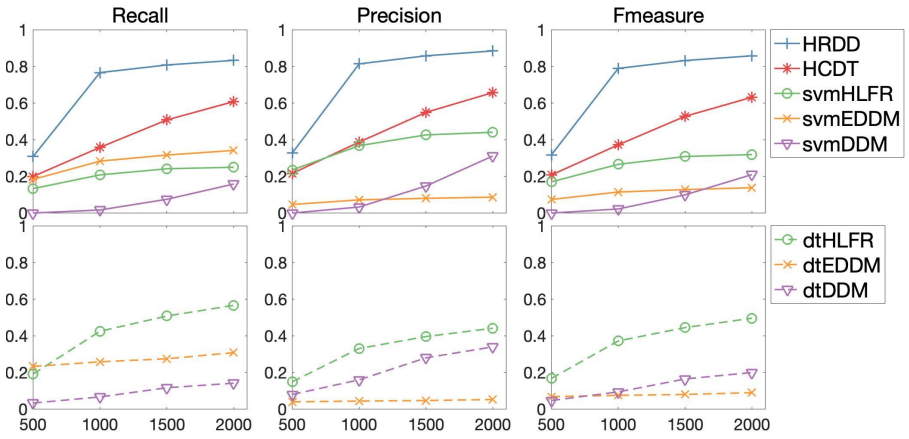


Figure 4.7: Detection performance for 6D Multivariate Gaussian. For detectors HLFR, EDDM and DDM: Linear SVM as the base classifier (top); decision tree as the base classifier (bottom).

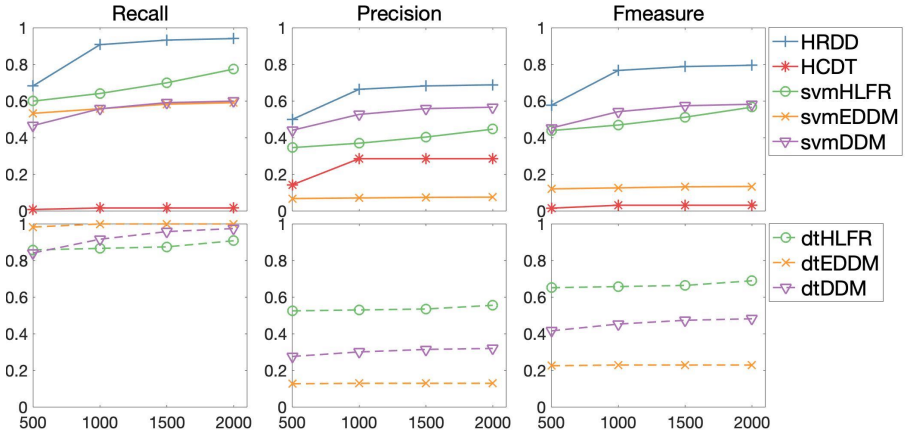


Figure 4.8: Detection performance for Rotating Checkerboard. For detectors HLFR, EDDM and DDM: RBF SVM as the base classifier (top); decision tree as the base classifier (bottom).

Table 4.8: Number of activations of each CDT for the TP detections on 30 sequences of 4D Multivariate Gaussian. Drift types are shown in Figure 4.3. Class 0 is the drifted class.

Drift type	Marginal	Class 0	Class 1
a)	7	21	2
b)	11	17	2
c)	7	21	2
d)	10	18	2

On the **Rotating Checkerboard** dataset, the effectiveness of HRDD can also be clearly identified in Figure 4.8. As expected, HCDT does not perform well because purely distribution-based detectors fail to detect changes affecting the labelling mechanism only (Alippi et al., 2013). The distribution of overall input space of this dataset remains unchanged. This phenomenon demonstrates that detecting concept drift by monitoring the class-conditional distributions is helpful. For this dataset, $P(Y|\mathbf{X})$ is significantly affected by all drifts, allowing the performance-based detectors to capture the drifts more acutely. Therefore HLFR, EDDM and DDM achieved very high recall values. However, the precision plot reveals that significantly more false alarms are triggered. Therefore, HRDD, which secures the highest F-measure, is still the most reliable choice. Another interesting finding from figures 4.7 and 4.8 is that when a decision tree is used as the base classifier, HLFR and EDDM achieve much better than when an SVM classifier is used. This confirms that the choice of classifier plays an important role in performance-based drift detection. In contrast, the performance achieved by HRDD is irrelevant to the base classifier.

Based on the above analysis, it can be concluded that for real drifts affecting $P(Y|\mathbf{X})$, HRDD performs no worse, and in many cases better than existing performance-based detectors. For virtual drifts not directly affecting $P(Y|\mathbf{X})$, HRDD performs better than both distribution-based and performance-based detectors. HRDD also performs

particularly better than the comparative methods when the changes have minor effect on the overall input distribution.

4.3.4 Experiment 3: Role in Classification

The focus of this work is to propose a new drift detector framework HRDD. Intuitively, accurate detection and localisation of drifts would help to improve classification because it leads to just-in-time model-retraining. What classification model and retraining techniques achieve the lowest classification error in a reactive streaming environment is a matter for future work. However, in order to evaluate the role of a more accurate and efficient detector in streaming data classification environments, we present the prequential classification error at the end of each sub-concept for **6D Multivariate Gaussian** and **Rotating Chekerboard** datasets. Experiments are carried out with two classifiers, SVM and decision tree. For the purpose of a fair comparison, all classifiers are trained and retrained within the original input space. Although it should be noted that for HRDD, it is also possible to train the classifiers within the reduced space derived with RSVM, which may lead to potentially better classification performance since RSVM serves the goal of both dimensionality reduction and classification accuracy improvement.

For performance-based detectors (HLFR, EDDM and DDM), the classifier is always retrained on a fixed-length recent window. For distribution-based detectors (HCDT and HRDD), a simple detect-then-retrain technique is adopted. Instances from the estimated drift starting point T_{ref} to detection point $\widehat{T}$ are used as the retraining set.

Tables 4.9 and 4.10 demonstrate that HRDD helps to achieve a lower classification error on both datasets no matter which base classifiers is adopted. For the **6D Multivariate Gaussian** dataset, recall that the first two drifts are virtual. The classification task actually becomes easier as the classes move further away from each other. Performance-based detectors consider these drifts to be irrelevant and do not detect such drifts. Even in these cases, HRDD, which accurately detects all types of drifts, leads to

Table 4.9: Classification error for 6D Multivariate Gaussian. Average prequential classification error (standard deviation in parenthesis) at the end of each sub-concept is presented. Best results are in **bold**.

	SVM-linear				
Concept	HRDD	HCDT	HLFR	EDDM	DDM
1	**0.12** (0.01)	**0.12** (0.01)	0.13 (0.02)	0.19 (0.05)	**0.12** (0.01)
2	**0.06** (0.01)	0.07 (0.01)	0.07 (0.01)	0.09 (0.04)	0.07 (0.01)
3	**0.04** (0.01)	0.05 (0.01)	0.06 (0.01)	0.07 (0.02)	0.06 (0.01)
4	**0.05** (0.01)	**0.05** (0.01)	0.07 (0.02)	0.08 (0.02)	0.07 (0.01)
5	**0.09** (0.01)	0.11 (0.01)	0.10 (0.02)	0.15 (0.05)	0.10 (0.02)

	Decision Tree				
Concept	HRDD	HCDT	HLFR	EDDM	DDM
1	**0.23** (0.02)	0.24 (0.02)	0.24 (0.02)	0.29 (0.05)	**0.23** (0.02)
2	**0.15** (0.01)	0.17 (0.02)	0.17 (0.02)	0.19 (0.05)	0.17 (0.02)
3	**0.11** (0.02)	0.13 (0.02)	0.15 (0.02)	0.16 (0.04)	0.15 (0.02)
4	**0.11** (0.01)	0.12 (0.02)	0.14 (0.02)	0.19 (0.03)	0.17 (0.02)
5	**0.16** (0.02)	0.19 (0.02)	0.18 (0.03)	0.25 (0.04)	0.17 (0.02)

Table 4.10: Classification error for Rotating Checkerboard. Average prequential classification error (standard deviation in parenthesis) at the end of each sub-concept is presented. Best results are in **bold**.

	SVM-rbf				
Concept	HRDD	HCDT	HLFR	EDDM	DDM
1	**0.12** (0.04)	**0.12** (0.04)	**0.12** (0.05)	0.21 (0.05)	0.14 (0.04)
2	**0.42** (0.04)	0.49 (0.02)	0.46 (0.04)	0.47 (0.04)	0.44 (0.05)
3	**0.13** (0.10)	0.84 (0.16)	0.27 (0.17)	0.31 (0.16)	0.31 (0.15)
4	**0.43** (0.09)	0.52 (0.07)	0.48 (0.03)	0.48 (0.05)	0.47 (0.07)
5	**0.11** (0.07)	0.16 (0.16)	0.35 (0.20)	0.37 (0.15)	0.33 (0.16)
	Decision Tree				
Concept	HRDD	HCDT	HLFR	EDDM	DDM
1	**0.03** (0.02)	0.04 (0.02)	0.04 (0.03)	0.04 (0.03)	0.05 (0.02)
2	**0.21** (0.06)	0.49 (0.06)	0.25 (0.09)	0.26 (0.06)	0.23 (0.11)
3	0.07 (0.09)	0.90 (0.20)	**0.05** (0.05)	**0.05** (0.06)	0.13 (0.15)
4	**0.21** (0.12)	0.53 (0.10)	0.26 (0.13)	0.25 (0.08)	0.22 (0.09)
5	**0.05** (0.07)	0.09 (0.20)	0.09 (0.14)	**0.05** (0.05)	0.10 (0.13)

an even lower error than the performance-based detectors. This supports the hypothesis that when the optimal decision boundary has shifted but performance is not deteriorated, retraining the classifier can still be beneficial. For the **Rotating Chekerboard** dataset, recall that the drifts affect the labelling mechanism only. The data distribution-based detector HCDT fails to make accurate detections, hence the much worse classification performance than the performance-based detectors.

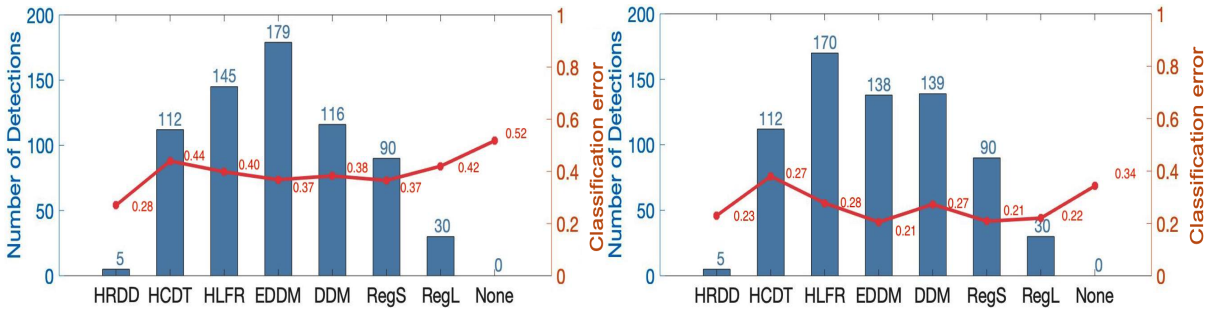
It is also worth noting that the standard deviations of the reported errors presented in tables 4.9 and 4.10 lead to large overlaps between HRDD and its comparative methods, which may signify that the improvement achieved by HRDD may not be statistically significant. Nonetheless, with a similar standard deviation and a lower mean classification error in most cases, it can still be concluded that HRDD can help in reducing classification error regardless of the drift type. For both real and virtual drifts, incorporating HRDD in a classification model can achieve a lower or at least comparable classification error than both performance-based and distribution-based detectors.

4.3.5 Experiment 4: Real-world Scenarios

In the above experiments, synthetic data streams are used to better understand the functionality, efficiency, and effectiveness of HRDD. For real-world data streams, there is no ground truth regarding the existence or location of drifts. Therefore, the performance metrics used for synthetic datasets cannot be employed. Here we report the number of detections and prequential classification error to compare the methods. A classification system that achieves the lowest number of detections as well as the lowest classification error is preferred.

Electricity (Harries and Wales, 1999) is a dataset collected from the Australian New South Wales Electricity Market. It contains 45,312 instances and each example is described by 8 features. The class label identifies the change of the price relative to a moving average of the last 24 hours. (i.e., up and down). We note that there has been

a dispute regarding the usage of this dataset for concept drift detection analysis due to the temporal correlation within the data (Zliobaite, 2013). Nonetheless, it is still one of the most commonly used real-world data streams in this area of research (Bifet et al., 2013a). In order to mitigate this issue, we also compare with situations where regular retraining takes place and where no detector is adopted.



(a) SVM-rbf as the base classifier

(b) Decision tree as the base classifier

Figure 4.9: Detection and classification performance for the Electricity data stream. The bar plot represents the number of detections raised by each method, and the line plot records the prequential classification error at the end of the data stream.

The number of detections and the classification error obtained by the methods concerned are presented in Figure 4.9. *RegS* and *RegL* stand for regular retraining with small and large intervals (500 and 1500), respectively. The selection of interval length directly affects classification performance as well as the computational cost. *None* indicates the no-detector situation. From the line plot representing the classification error, it can be seen that adding a drift detector always help reducing the classification error since all methods lead to lower error when comparing with the no-detector situation.

From the bar plot representing the number of detections, it can be seen that HRDD always ranked first, with only 5 detections regardless of the choice of base classifier. In contrast, its competitors all raise many more detections, causing higher overhead cost. HRDD not only bears a very low computational burden from retraining, but also helps to maintain a satisfactory classification performance. The line plots in Figure 4.9 demonstrate that when an SVM is used as the base classifier, the error obtained by

HRDD ranked first, and is much lower than what has been achieved by its competitors. When a decision tree is adopted, HRDD ranked fourth with a classification error of 0.23, being only 0.02 higher than the best result achieved by EDDM. Examining the number of detections and classification error together, we may conclude that in summary, HRDD achieves the best trade-off between classification performance and computational cost on this real-world data stream.

Table 4.11: Average runtime for each reported detection (s.).

Dataset	HRDD	HCDT	HLFR	EDDM	DDM
4D Gaussian	0.2496	0.1044	49.0637	0.2645	2.1117
6D Gaussian	0.6129	0.2989	63.3693	0.5735	3.0722
Rotating Checkerboard	0.2004	0.4088	36.2308	1.1619	6.9635

4.3.6 Computational Time Complexity Analysis

DDM (Gama et al., 2004) and EDDM (Baena-García et al., 2006) have a constant time complexity ($\mathcal{O}(1)$) at each time point, since they monitor a single error-rate based statistic. Although the base detector LFR (Wang and Abraham, 2015) in HLFR (Yu et al., 2019) also has complexity ($\mathcal{O}(1)$), the validation layer requires extra training of P classifiers (P=1000 in the original paper). Assuming $\mathcal{O}(K)$ is the computational complexity of training a new classifier, the time complexity for HLFR is $\mathcal{O}(KP)$, which is usually much higher than ($\mathcal{O}(1)$). HCDT (Alippi et al., 2017) adopts an univariate test on each dimension in the detection layer and one offline test in the validation layer. If the complexity of the base detector is ($\mathcal{O}(1)$), the complexity of the overall framework is ($\mathcal{O}(d)$) where d is the dimensionality of input space. HRDD has a similar structure but adopts a univariate test on each dimension of the reduced-space for each class in the detection layer. The time complexity is ($\mathcal{O}(RQ)$) ($R = 1$ and $Q = 2$ in this work so ($\mathcal{O}(RQ)$) is close to ($\mathcal{O}(1)$)).

For multivariate data streams of higher dimensionality, the advantage of HRDD

will become more significant since the number of classes is usually much lower than the number of dimensions. The average runtime for a reported detection is summarized in Table 4.11. It is worth noting that performance-based detectors generally have longer runtime since they also include a classifier training procedure which data distribution-based detectors do not. Practitioners should take this into consideration when choosing the appropriate detector depending on the application scenario.

4.4 Chapter Summary

This chapter starts to investigate the detection of changes in online classification tasks. The main contribution of this chapter is the hierarchical drift detection framework HRDD for multivariate supervised data streams. The proposed framework first maps the data to a lower dimensional subspace, and then detects drifts in that space relevant to the given classification task. It utilizes information from both marginal distribution and class-conditional distributions of the supervised data stream. Based on the effect of drift on each class, a novel reconfiguration scheme aiming to maintain as many relevant instances for retraining as possible is incorporated within the algorithm.

HRDD is computationally light and efficient when operating on higher-dimensional data streams. For both real and virtual drifts, HRDD performs no worse, and in many cases better, than state-of-the-art detection algorithms, whether they are performance-based or distribution-based. Compared with the existing data-based detectors, it is a more accurate and efficient approach in terms of a high number of true detections, while maintaining a low number of false alarms, when operating on higher-dimensional data streams. It is also capable of detecting subtle drifts which can hardly be captured by existing data-based detectors.

In this chapter, the parameters of the existing drift detectors were the same as the ones used in their original papers. Although these are recommended parameters

to be used by the original authors, it is still worth noting that they may not be the optimal parameters for our testing data streams, which can be regarded as a threat to validity. Implementing the comparative methods with different parameter settings may lead to potentially better results. However, in the scenario of online classification, the only information available during the training stage is the sequence of data representing the current concept, making it difficult to carry out more careful parameter tuning for better detection.

We have only showed one possible implementation of HRDD. In our current realization, data projection is carried out with RSVM (Tao et al., 2008) due to its advantageous ability to conduct dimensionality reduction and improve classification performance simultaneously. Other tools with the same properties can also be considered. Besides, in the current work, we only projected data onto a single-dimensional linear subspace. How various characteristics of the subspace affect the detection performance should be investigated in future work. In fact, HRDD can also be used with many different settings. Which combination of detection and validation tests will achieve the best detection performance for various data streams is another topic worth further exploration. The choice can also be application-specific. For instance, with a suitable choice of base detection tests, extension can also be made to accommodate multi-class data streams and even imbalanced-class data streams.

Chapter Five

Concept Drift Detection in Data Streams with Temporal Dependency

As can be seen in the last chapter, with accurate detections of concept drifts, the violation of the *identically-distributed* assumption in online classification can be handled, in-time modifications can be made to the operating systems to restore the overall performance. However, existing drift detection methods still function under the *independently-distributed* assumption, which can hardly be satisfied in real-world applications. This assumption also has to be further relaxed since temporal dependency is often involved. This chapter aims to understand and tackle the joint issue of concept drift and temporal dependency, attempting to provide answers to the set of research questions listed in Section 1.2.3.

Section 5.1 briefly introduces the motivations of this chapter and the contributions. Section 5.2 provides a supplementary analysis of existing concept drift detection methods and summarizes the related work dealing with temporal data streams in online classification. Section 5.3 presents a systematic formulation of the joint issue, a novel taxonomy to categorize different forms of drifts, a synthetic data generator as well as a simple benchmark detector algorithm to manage the joint issue. In Section 5.4, two sets of experiments are carried out to demonstrate the deficiency of existing state-of-the-art detectors, and the improved detection ability with our newly proposed detector. Sec-

tion 5.5 summarizes the chapter and highlights the urgency for further research in this direction.

5.1 Introduction

Temporal dependency itself is not a new issue (Beck et al., 1998). However, its effects on data stream classification and concept drift are relatively unexplored. As shown in Section 2.2.3, most existing drift detection methods choose to assume data independence. The threat of temporal dependency poses to these concept drift detectors is yet to be investigated. Nonetheless, the combined issue of concept drift and temporal dependency commonly exists in reality. For instance, the most popular real-world data streams used for the evaluation of concept drift detectors such as KDD'99 (Archive, 2022) and Electricity (Harries and Wales, 1999) are all known to subject to somewhat extreme temporal dependency (Zliobaite et al., 2015).

In fact, not till very recently, have researchers started to notice the challenge of the independence assumption in online classification. Zliobaite (2013) are the first group of authors pointing out this issue (Zliobaite, 2013; Bifet et al., 2013b). However, their work only target at correcting and improving the predictive modelling performance, and are not related to concept drifts or their detections.

So far, there is still a lack of comprehensive discussion on the new challenges in concept drift brought by temporal dependency. In two of the most recent review papers on supervised data stream learning and concept drift (Lu et al., 2020; Wares et al., 2019), the lack of attention paid to temporal dependency is also highlighted. It is also stated that there is a lack of established base datasets and the impact of temporal dependence on concept drift detection Wares et al., 2019. With the work presented in this chapter, we aim to fill in the gap by providing a systematic study of the joint issue, discussing how their interactions bring new challenges to the current state of

research and demonstrating our findings with simulated data sets. Besides, we also verify experimentally that the existence of temporal dependency itself, under the premise of precise and accurate extraction, may be used as a tool for more efficient drift detection. In this chapter, we aim to make the following 4 contributions:

1) We formally show that temporal dependency is indeed a generic problem which also affects the concept drift detection problem. We provide a novel taxonomy for various types of concept drifts that may exist in temporal-dependent data streams.

2) We propose a new and flexible data stream generator to simulate different types of temporal-dependent data streams.

3) We show experimentally that when taking temporal dependency into consideration, existing detectors may still work in some cases, but fail in some others. We list out what forms of drifts require urgent attention and conclude that concept drift detection for temporal dependent data streams is an issue worth further investigation.

4) We provide a benchmark solution named Concept Drift detection for Temporally Dependent data streams (CDTD) to detect the wide range of drifts that can possibly occur in temporal dependent data streams.

5.2 Why Temporal Dependency May Be an Issue

In this section we restate why and how existing concept drift detection methods may be negatively affected by temporal dependency. Afterwards, existing work related to the issue of temporal dependency in general online classification scenarios are also be summarized.

As stated in Section 2.2.3, concept drift detection methods can be divided into classifier-based ones and data-based ones. Existing classifier-based detection methods are based on the rationale that an increase in classification error is likely to represent the

occurrence of a concept drift. Under the 0–1 loss function, the error is a Bernoulli trial, described by a binomial distribution. When the number of observations is large enough, the error rate can be approximated using a Gaussian distribution (Gama et al., 2004; Baena-García et al., 2006; Ross et al., 2012; Harel et al., 2014). Then various statistical tests can be applied to the specified test statistics to detect drifts. However, these tests are based on the assumption that each repeated trial is an independent event. When temporal dependency exists, the prediction error stream itself also contains dependency (Zliobaite et al., 2015). Hence, the premise of many statistical tests is violated. Previous work also demonstrate that when temporal dependency exists in the label stream, classification error is no longer a suitable metric for assessing drift because false alarms may actually decrease classification error, resulting in misleading results (Zliobaite et al., 2015). Therefore, the detection performance of classifier-based methods may be hindered.

For data-based detectors, temporal dependency can also be a potential issue. In many such detectors, it is explicitly stated in their problem formulations that only independent random observations drawn from an unknown probability density function are received (Harel et al., 2014; Lu et al., 2014; Bu et al., 2016). Even if this is not explicitly stated, the various statistical tests and theorems forming the basis of the detectors rely on this inherent assumption (Alippi et al., 2010b; Alippi and Roveri, 2008; Qahtan et al., 2015).

It should be noted that the existence of temporal dependency within the data streams may not necessarily hinder the performance of existing detectors in the detection of all possible forms of drifts, but undoubtedly impedes the reliability of them since there will be no guarantees for optimality of the results. How well existing methods can perform in such situation is worth investigating. However, all existing methods have been tested on synthetic data streams assuming independence only (Minku et al., 2010; Elwell and Polikar, 2011; Lu et al., 2016). In this work, we wish to experimentally demonstrate the ability of existing drift detectors to cope with the environment where temporal dependency exists.

In fact, non-stationary data stream mining with temporal dependency has long been neglected. As a pioneering work discussing temporal dependency in concept drifting data streams, Zliobaite (2013) first spots the existence of temporal dependency in the label stream of the Electricity data stream (Harries and Wales, 1999). The author proposes a new evaluation measure named Kappa-Temporal to correct the bias brought by temporal dependency in classification performance evaluation. However, it is only a statistical metric and does not offer any mechanism to handle temporal dependency during the online classification process (Wares et al., 2019). In their following work, two generic meta-learning approaches that can be used to wrap state-of-the-art classifiers to account for temporal dependency during model training are proposed (Zliobaite et al., 2015; Mayo and Bifet, 2016). However, these methods are solely for the purpose of correcting and improving the predictive modelling performance, and are not related to concept drifts or their detections. It has also been briefly mentioned in their work that although temporal dependency is likely to violate the assumptions of current drift detection methods, in practice the impact of this issue is small (Zliobaite et al., 2015). However, we reckon that this conclusion is tenable only for a subset of existing detectors and limited forms of drifts.

Candidate Change Point Detector (CCPD) (Duong et al., 2018) is a recent as well as the only detector making effort to exploit and utilize the temporal dependency between data within the stream for change detection. However, it is assumed that the observations consisting the data stream are drawn from a Gaussian distribution, and the only form of drift considered is a change in the feature mean, which cannot fully reflect the consequential implications of the combined issue. Other forms of change such as drifts affecting decision boundary only are not detected. Although having some limitations, this work conveys the idea that temporal dependency may also bring possibilities to the development of more advanced concept drift detectors.

As mentioned earlier, temporal dependency itself is not a new problem and has been investigated in the field of time series analysis. However, the problem setting of

such problems are different from what is being considered in this work. Firstly, time series analysis mainly deals with regression modelling tasks. Time series classification tasks typically refers to sequence classification, where the task is to predict a single label that applies to an entire input sequence. Secondly, change detection for time series data generally requires strong assumptions. It is often assumed that at least partial information, if not the entire sequence, of the time series is known a-priori (Aminikhanghahi and Cook, 2017). For instance, some work employ the AR model to represent a time series for detection (Yamanishi and Takeuchi, 2002), some others assume that time series can be represented satisfactorily by a single autoregressive ARIMA model (Hilas et al., 2013; Srivastava et al., 2016). OAR-DLSTM (Sun et al., 2021) is a very relevant work that utilizes temporal dependency to detect change in time series prediction tasks. However, it requires the information of all possibly-occurring concept in the offline pre-training stage, which makes it unsuitable for our problem scenario.

5.3 How Important is Temporal Dependency in Concept Drift Detection

This section first provides a novel problem formulation for describing the online classification learning scenario as well as concept drift detection in temporally dependent data streams. Based on this formulation, a novel taxonomy emphasizing the interaction of the joint issue is presented. Then a novel synthetic data generator is proposed to simulate various forms of drifts that can possibly occur in temporal dependent data streams, including the ones that are neglected by existing work. Lastly, a simple baseline drift detector targeting the new scenario is introduced.

5.3.1 A More Detailed Problem Formulation

In a streaming environment, a supervised data stream is formed by observations $\{(\mathbf{x}_t, y_t), t \in \mathbb{Z}^+\}$. $\mathbf{x}_t \in \mathbb{R}^d$ represents the d-dimensional feature vector at time stamp t and y_t is its class label. $y_t \in \{0, 1, ..., Q\}$ where $Q + 1$ is the number of available classes. In the current problem formulation, each data point $(\mathbf{x}_t, y_t)$ is considered to be an independent realization of the random variable $Z = (\mathbf{X}, Y)$ generated by $P(Z) = P(\mathbf{X}, Y)$ at time stamp t. A concept drift is said to occur when there is a change in the joint distribution $P(\mathbf{X}, Y)$. As described in Section 2.1.2, this distribution can be further decomposed as $P(\mathbf{X}, Y) = P(Y|\mathbf{X}) \cdot P(\mathbf{X}) = P(Y) \cdot P(\mathbf{X}|Y)$, symbolizing four possible sources of a drift. A change in any of these probabilities leads to a concept drift.

In order to better describe the problem by taking temporal dependency into consideration, we provide a new formulation for concept drift in online classification as follows. Rather than fixed random variable Z, the data stream can be considered as a joint stochastic process $\mathcal{Z} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the data feature process and label process, respectively. A stochastic process is a mathematical model that describes a sequence of random variables. We define $\mathbb{P}\{\mathcal{Z}\}$ as the law of the joint process for $\mathcal{Z}$, and $P(\mathcal{Z})$ as a joint probability of the random variables constituting $\mathcal{Z}$. $\mathbb{P}\{\mathcal{X}\}$, $P(\mathcal{X})$ and $\mathbb{P}\{\mathcal{Y}\}$, $P(\mathcal{Y})$ can be defined analogously for processes $\mathcal{X}$ and $\mathcal{Y}$, respectively.

We now introduce some further definitions taking the process $\mathcal{X}$ as an example. $\mathcal{X}$ denotes the stochastic process for the data feature stream, and is consisted of random variables $\mathbf{X}_{t,s}$ where t is the time index and s is the number of realization. t can approach ∞ as size of the data stream grows. For simplicity, we assume that t represents the time stamp index taken at a regular rate and only one single data stream is examined at each t, i.e., $s = 1$. Therefore, $\mathbf{X}_{t,s}$ can be simplified to $\mathbf{X}_t$. Note that in an online classification scenario, at some time stamp t, only the information up to t is available. Therefore, in our scenario, we have $\mathcal{X} = \{\mathbf{X}_t : t \in \mathbb{Z}^+\}$ where $\mathbf{X}_t$ is the random variable at time stamp t of the process. Following this, $\mathbf{X}_{\{1:t\}} = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_t)$ hence represents a

Table 5.1: A summary and comparison of the new formulation, which takes into account temporal dependency, and the existing formulation, which assumes data independence, for describing the supervised data streams.

	Notation	Definition
New formulation	$\mathcal{X} = \{\mathbf{X}_t : t \in \mathbb{Z}^+\}$	The stochastic process for data feature stream
	$\mathbf{X}_t$	The random variable for data features at time t
	$\mathbf{X}_{\{1:t\}} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$	A random vector representing a sub-process of $\mathcal{X}$ from time stamp 1 to t
	$\mathbf{x}_{\{1:t\}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t)$	One possible realization of $\mathbf{X}_{\{1:t\}}$
	$\mathbb{P}\{\mathcal{X}\}$	The law of stochastic process $\mathcal{X}$
	$P(\mathbf{X}_{\{1:t\}})$	The joint probability of $\mathbf{X}_{\{1:t\}}$ for some t
	$\mathcal{Y} = \{Y_t : t \in \mathbb{Z}^+\}$	The stochastic process for the data label stream
	Y_t	The random variable for data label at time t
	$\mathbf{Y}_{\{1:t\}} = (Y_1, \ldots, Y_t)$	A random vector representing a segment of $\mathcal{Y}$ from time stamp 1 to t
	$\mathbf{y}_{\{1:t\}} = (y_1, y_2, \ldots, y_t)$	One possible realization of the data label stream vector $\mathbf{Y}_{\{1:t\}}$
	$\mathbb{P}\{\mathcal{Y}\}$	The law of stochastic process $\mathcal{Y}$
	$P(\mathbf{Y}_{\{1:t\}})$	The joint probability of $\mathbf{Y}_{\{1:t\}}$ for some t
	$\mathcal{Z} = \{\mathcal{X}, \mathcal{Y}\}$	The joint stochastic process representing the supervised data stream
	$\mathbb{P}\{\mathcal{Z}\} = \mathbb{P}\{\mathcal{X}, \mathcal{Y}\}$	The underlying generation mechanism of $\mathcal{Z}$
	$P(\mathbf{X}_{\{1:t\}}, \mathbf{Y}_{\{1:t\}})$	The joint probability distribution of $P(\mathbf{X}_{\{1:t\}}, \mathbf{Y}_{\{1:t\}})$
Existing formulation	$\mathbf{X}$	The random variable for data features
	$P(\mathbf{X})$	The underlying probability distribution of $\mathbf{X}$
	$\{\mathbf{x}_t : t \in \mathbb{Z}^+\}$	t i.i.d realizations of the random variable $\mathbf{X}$
	Y	The random variable for data labels
	$P(Y)$	The underlying probability distribution of Y
	$\{y_t : t \in \mathbb{Z}^+\}$	t i.i.d realizations of the random variable Y
	$Z = (\mathbf{X}, Y)$	The joint random variables representing the i.i.d supervised data stream
	$P(Z) = P(\mathbf{X}, Y)$	The joint probability distribution of Z

random vector of $\mathcal{X}$ from time stamp 1 to t. $\mathbf{x}_{\{1:t\}} = (\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_t)$ is used to denote one possible realization of $\mathbf{X}_{\{1:t\}}$. $\mathbf{x}_t \in \mathbb{R}^d$ represents the d-dimensional feature vector of the t^{th} observation as before. In other words, we consider $\mathcal{X}$ to be discrete-time continuous process. The stochastic process of the labels $\mathcal{Y} = \{Y_t : t \in \mathbb{Z}^+\}$ can be defined in a similar way. $y_t \in \{0, 1, ..., Q\}$ where $Q + 1$ is the number of available classes. Therefore, $\mathcal{Y}$ can be regarded as a discrete-time discrete process. Please see Table 5.1 for full details of the definitions.

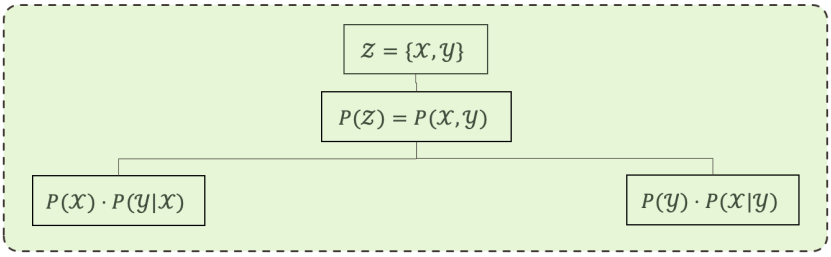


Figure 5.1: A newly proposed taxonomy for drift categories taking into account temporal dependency. The underlying data stream is described by the stochastic process $\mathcal{Z} = \{\mathcal{X}, \mathcal{Y}\}$, instead of a random variable $Z = (X, Y)$.

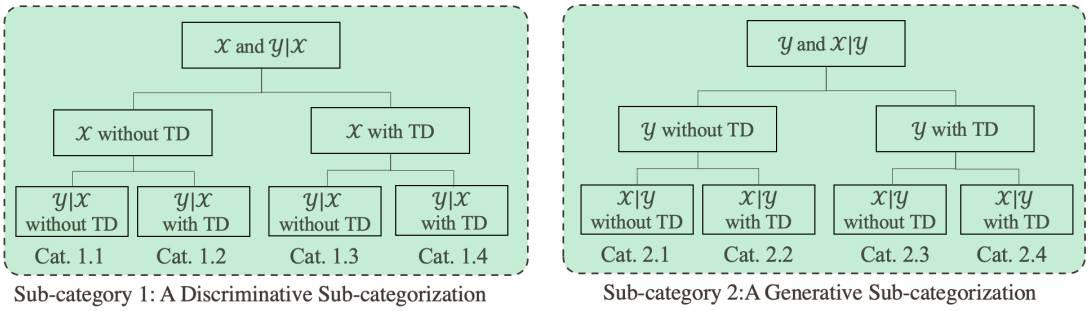


Figure 5.2: An extended sub-taxonomy of the overall categorization taking into account temporal dependency (TD) in Figure 5.1 from both discriminative perspective (left) and generative perspective (right).

Under this new problem formulation, a concept drift in this case is essentially equivalent to a change that occurs in the law of the joint stochastic process $\mathbb{P}\{\mathcal{X}, \mathcal{Y}\}$, or from a probabilistic point of view, the joint probability of $P(\mathcal{X}, \mathcal{Y})$. This probability

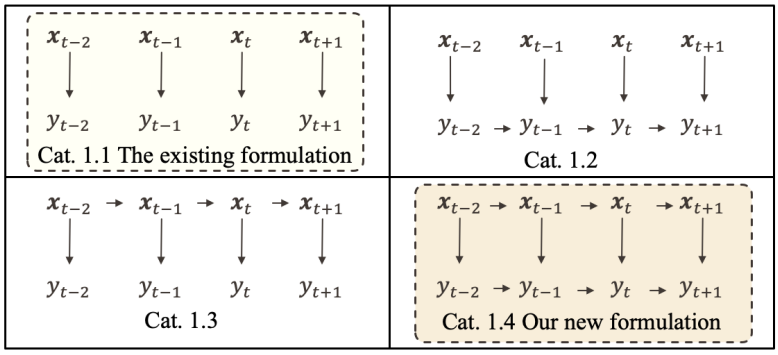


Figure 5.3: An illustration of temporal dependency within the supervised data stream for sub-category 1 with an order of 1. Arrows from A to B represents the dependency of B on A. Cat. 1.1 is the case considered by existing problem setting. Cat 1.4 is the most general case that our new formulation considers. Examples for sub-category 2 can be presented analogously.

can also be decomposed further from either a discriminative or generative perspective as shown in Equation 5.1. Figure 5.1 presents the new categorization of the possible sources of concept drift under the new problem formulation.

$$P(\mathcal{X}, \mathcal{Y}) = P(\mathcal{Y}|\mathcal{X}) \cdot P(\mathcal{X}) = P(\mathcal{X}|\mathcal{Y}) \cdot P(\mathcal{Y}). \tag{5.1}$$

The benefit of this new taxonomy is not limited to the more practical definitions from random variables to stochastic processes. More essentially, it demonstrates the possible existence of a wider range of concept drift forms than what have been considered by the existing work.

It should be noted that temporal dependency can also manifest itself in various forms. A stochastic process can also consists of independent random variables only. Following the broad taxonomy presented in Figure 5.1, a further sub-categorization can be constructed as in Figure 5.2 to show where the temporal dependency can exist. To explain the sub-categorization, Figure 5.3 provides some examples for Cat.1.1-1.4 in sub-category 1. Cat 1.1 is a special case under the new formulation where $\mathcal{X}$ and $\mathcal{Y}$ are both independent stochastic processes, which is also the case considered by existing work. Cat 1.4 is the most general form of temporal dependency existence considered by our

formulation and categorization.

The more complicated sources of drift also bring new challenges to the detection of drifts. We aim to theoretically demonstrate that existing methods are likely to fail in detecting some forms of drifts in temporal dependent data streams. Taking the discriminative approach as an example, the drift detection task is to find out if and when there is a change in $P(\mathcal{X})$ and $P(\mathcal{Y}|\mathcal{X})$. Equations 5.2 and 5.3 show the expanded form of the two probabilities under the chain rule. At each time stamp t, the change may take place in $p(y_t|y_{\{1:t-1\}}, \mathbf{x}_{\{1:t\}})$ and/or $p(\mathbf{x}_t|\mathbf{x}_{\{1:t-1\}})$. Nonetheless, existing methods detect drifts by monitoring $p(\mathbf{x}_t)$ and $p(y_t|\mathbf{x}_t)$ at each time stamp t, which is only sufficient in the special case of the new categorization where the $independently-distributed$ assumption is satisfied. That is, the stochastic processes are temporally independent. Hence, with current concept drift detection methods, changes related to temporal dependence cannot be well captured.

Stochastic processes are usually associated with some order $\tau \in \mathbb{Z}_0^+$. τ can be regarded as the number of previously seen instances that are considered when calculating the probabilities for the current instance. For simplicity, we assume the processes $\mathcal{X}$ and $\mathcal{Y}|\mathcal{X}$ have the same order. Taking τ into consideration, expressions in equations 5.2 and 5.3 can be further simplified as $p(\mathbf{x}_t|\mathbf{x}_{\{1:t-1\}}) = p(\mathbf{x}_t|\mathbf{x}_{\{t-\tau:t-1\}})$ and $p(y_t|y_{\{1:t-1\}}, \mathbf{x}_{\{1:t\}}) = p(y_t|y_{\{t-\tau:t-1\}}, \mathbf{x}_{\{t-\tau+1:t\}})$, respectively. τ can also evolve over time.

$$
\begin{aligned}
&P(\mathbf{X}_{\{1:t\}} = \mathbf{x}_{\{1:t\}}) \\
&= p(\mathbf{x}_1, \ldots, \mathbf{x}_t) \\
&= p(\mathbf{x}_1) \cdot p(\mathbf{x}_2|\mathbf{x}_1) \cdot p(\mathbf{x}_3|\mathbf{x}_{\{1:2\}}) \ldots \\
&\cdot p(\mathbf{x}_t|\mathbf{x}_{\{1:t-1\}}) \\
&\overset{\text{i.i.d}}{=} \prod_{i=1}^{t} p(\mathbf{x}_i)
\end{aligned}
\tag{5.2}
$$

$$
\begin{aligned}
&P(\mathbf{Y}_{\{1:t\}} = \mathbf{y}_{\{1:t\}} | \mathbf{X}_{\{1:t\}} = \mathbf{x}_{\{1:t\}}) \\
&= p(y_1, ..., y_t | \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t) \\
&= p(y_1 | \mathbf{x}_1) \cdot p(y_2 | y_1, \mathbf{x}_{\{1:2\}}) \dots \\
&\cdot p(y_t | y_{\{1:t-1\}}, \mathbf{x}_{\{1:t\}}) \\
&\overset{\text{i.i.d}}{=} \prod_{i=1}^{t} p(y_i | \mathbf{x}_i)
\end{aligned}
\tag{5.3}
$$

5.3.2 A Novel Synthetic Data Generator

Although real-world data streams often contain temporal dependency, without ground truth information, it is not possible to know if there really is a drift, when exactly the drift occurs, or which form of drift is present when working with such data streams. Therefore, it is not possible to perform a detailed analysis of the behaviour of various detection algorithms using only real-world data sets.

Various synthetic drift generators have been proposed in the literature, for instance, hyperplane (Lu et al., 2016), SINE (Minku et al., 2010), rotating checkerboard (Elwell and Polikar, 2011), and interchanging RBF (Losing et al., 2016). However, the generation mechanisms are all based on the assumption of data independence. As far as we know, there is no established benchmark generator for simulating temporally dependent data streams containing concept drifts. Hence, we propose a novel data generator with the purpose of not only filling in this research gap, but also promoting further studies following this research direction.

The data generator presented in Table 5.2 is built based on cosine and sine functions. We assume a bi-class supervised data stream with two-dimensional input is to be generated. x_t^1 and x_t^2 are the feature values of the two dimensions at time t, respectively. A, B, C are random non-negative numbers summing to 1. δ is an added constant to differentiate the feature mean. ϵ is a Gaussian noise with small standard deviation (e.g., $1e-6$). Based on x_t^1, x_t^2, and some previously obtained values of features and

Table 5.2: Details of the data generator for two-dimensional temporal dependent data streams. x_t^1 and x_t^2 are the feature values of the first and second dimension of the data stream at time t, respectively. $A, B, C, a, b, \delta, \theta$ and τ are the generator parameters. A, B, C are parameters controlling the temporal dependency of $\mathcal{X}$; δ is the parameter that determines the mean of the features of $\mathcal{X}$; a, b are parameters controlling the the temporal dependency of process $\mathcal{Y}|\mathcal{X}$; τ and θ represent the order of dependency and the decision threshold of process $\mathcal{Y}|\mathcal{X}$, respectively; ϵ is a small Gaussian noise added to the data streams.

Drift generator for temporal dependent data streams
$x_t^1 = A \cdot sin(A \cdot \pi \cdot t) - B \cdot cos(B \cdot \pi \cdot t) + C \cdot sin(C \cdot \pi \cdot t) + \delta + \epsilon$
$x_t^2 = A \cdot cos(A \cdot \pi \cdot t) - B \cdot sin(B \cdot \pi \cdot t) + C \cdot cos(C \cdot \pi \cdot t) + \delta + \epsilon$
$\tilde{y}_t = a \cdot (x_t^1 + x_t^2) + 0.8a \cdot \sum_{i=t-\tau+1}^{t-1}(x_i^1 + x_i^2) - b \cdot \tilde{y}_{t-1} - 0.8b \cdot \sum_{i=t-\tau}^{t-2} \tilde{y}_i$
if $sigmoid(\tilde{y}_t) > \theta, y_t = 1;$ otherwise $, y_t = 0.$

label streams from time stamp $t - \tau$ to t with order τ, a latent variable $\tilde{y}_t$ is calculated. After applying the sigmoid transformation on $\tilde{y}_t$, the actual label y can be determined by comparing $sigmoid(\tilde{y}_t)$ to a fixed decision criterion θ.

We choose to use sine and cosine functions to construct the generator for the following reasons: 1) The addition (subtraction) of sine and cosine waves is a more complex times series than standard time series such as ARIMA, which can better imitate real-world data streams. 2) Sine and cosine functions are bounded and so are their additions (subtractions). Other time series generation functions such as ARMA and ARIMA, on the other hand, are more likely to experience explosive growth when eigenvalues lie outside the unit circle.

By inserting various parameter values, the generator can be used to simulate various drifts, corresponding to possible forms of change that fall into different categories listed in Figure 5.2. It should be noted that some forms of drift may occur only when certain dependencies exist. For instance, a change in the temporal dependency of $\mathcal{X}$ can

Table 5.3: Some examples of possible drift forms that may occur in temporal dependent data streams with our data simulator. Details of the parameters involved and their meaning are presented in 5.2. Details of different categories (Cat.) are presented in Figure 5.2.

Drift in	Form	Name	Change in parameter	Drift cannot occur in	Our CDTD detector
$P(\mathcal{X})$	1	**TD-XX**	A, B, C	Cat. 1.1, 1.2	✓
$P(\mathcal{X})$ and $P(\mathcal{Y}\|\mathcal{X})$	2	**NUM-X**	δ $(, \theta)$	[ ]	✓
$P(\mathcal{Y}\|\mathcal{X})$	3	**TD-XY**	a	Cat. 1.1, 1.3	✓
	4	**TD-YY**	b	Cat. 2.1, 2.3	✓
	5	**TD-OY**	τ	Cat. 1.1, 1.3	✓
	6	**CLF**	θ	[ ]	✓

not be realized for Cat. 1.1 and 1.2 where $\mathcal{X}$ is an independent process only.

Table 5.3 provides some possible examples of simulated drifts. Some plots demonstrating the simulated drifts are presented for the synthesized drift examples. Due to space limitation, only the plots for **TD-XX** (Figure 5.4), **NUM-X** (Figure 5.5) and **CLF** (Figure 5.6) are presented. The drift occurs at timestamp 300. The upper plot is a demonstration of the change in the first feature of the generated data stream. The middle plot is an autocorrelation (ACF) plot that calculates the correlation of a stochastic process, say $\mathcal{Y}$, with itself k lag away (Equation 5.4). It is used here for the visualization of temporal dependency and the change in it. The bottom plot is a scatter plot showing the data points from different classes within the input space.

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2} \tag{5.4}$$

With these synthetic data streams, researchers can test the functionality and performance of existing detectors. After identifying the forms of drift in which the existing algorithms are weak, more sophisticated strategies can be further designed to solve the particular challenging cases. These methods can also be combined with existing ones to improve detector generalization power and better resolve real-world issues.

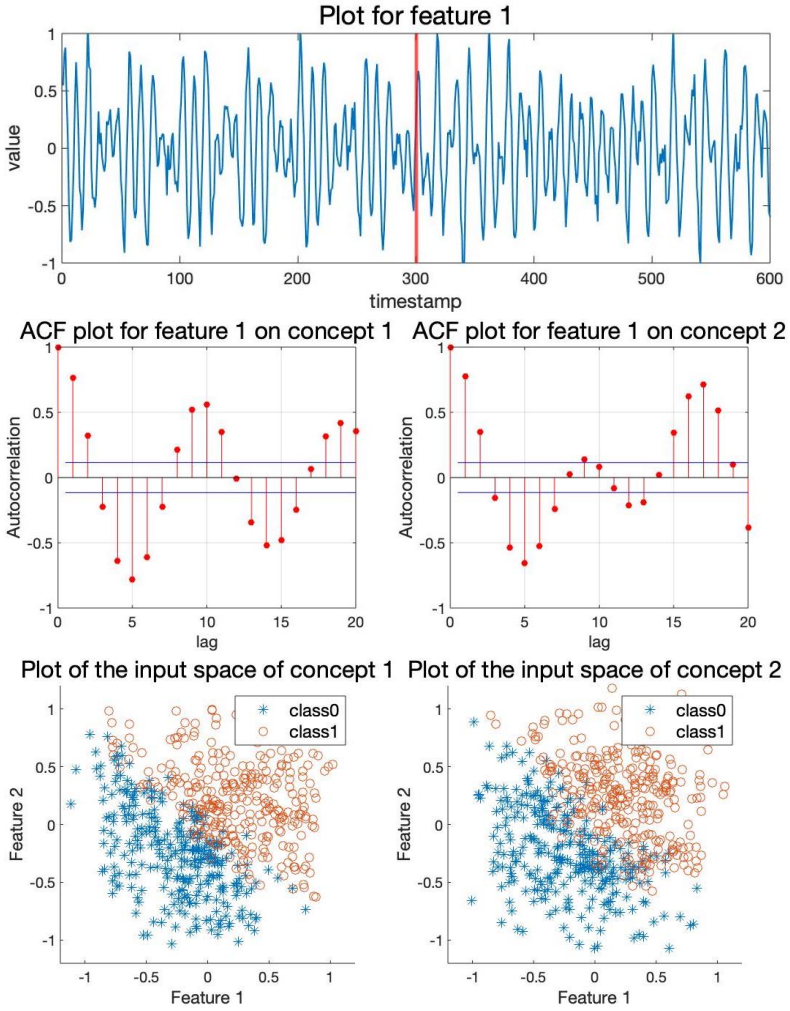


Figure 5.4: An example for **TD-XX** drift (Drift form 1 in Table 5.3). No obvious change can be spotted from either the feature plot (upper plot) or the input space plot revealing the class-conditional distributions (bottom plot). The change is only noticeable from the ACF plot representing the temporal dependency only (middle plot).

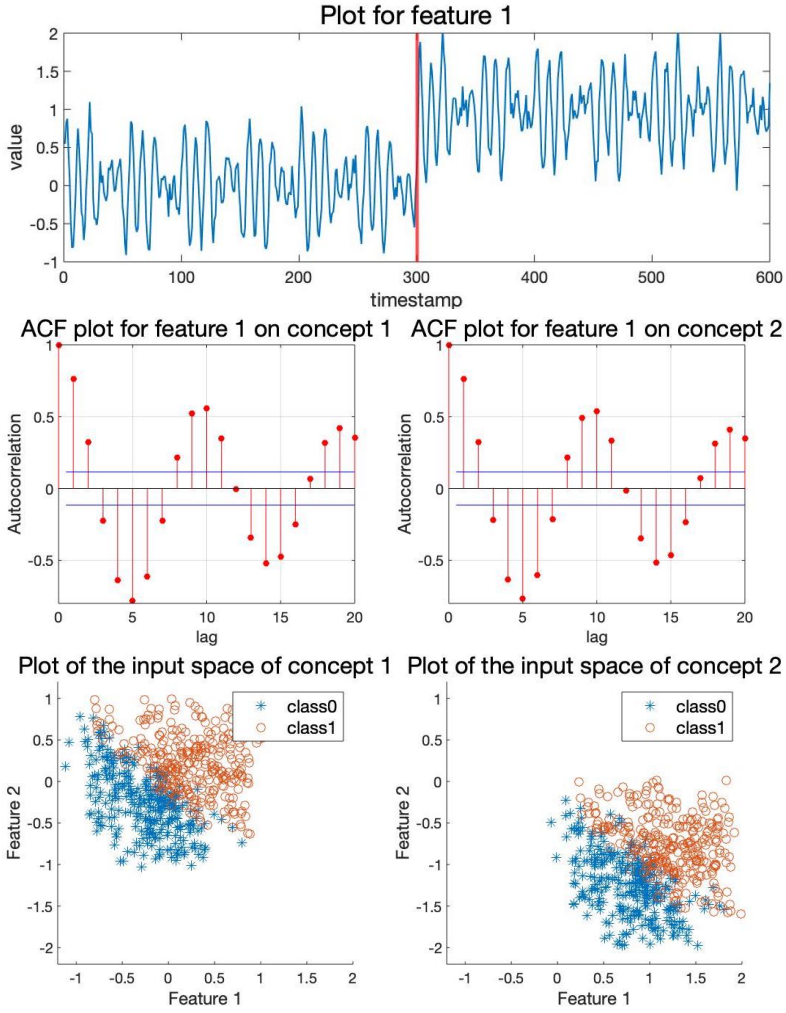


Figure 5.5: An example for **NUM-X** drift (Drift form 2 in Table 5.3). An obvious change can be spotted from the feature plot (upper plot) and the input space plot revealing the class-conditional distributions (bottom plot). The change is not noticeable from the ACF plot representing the temporal dependency only (middle plot).

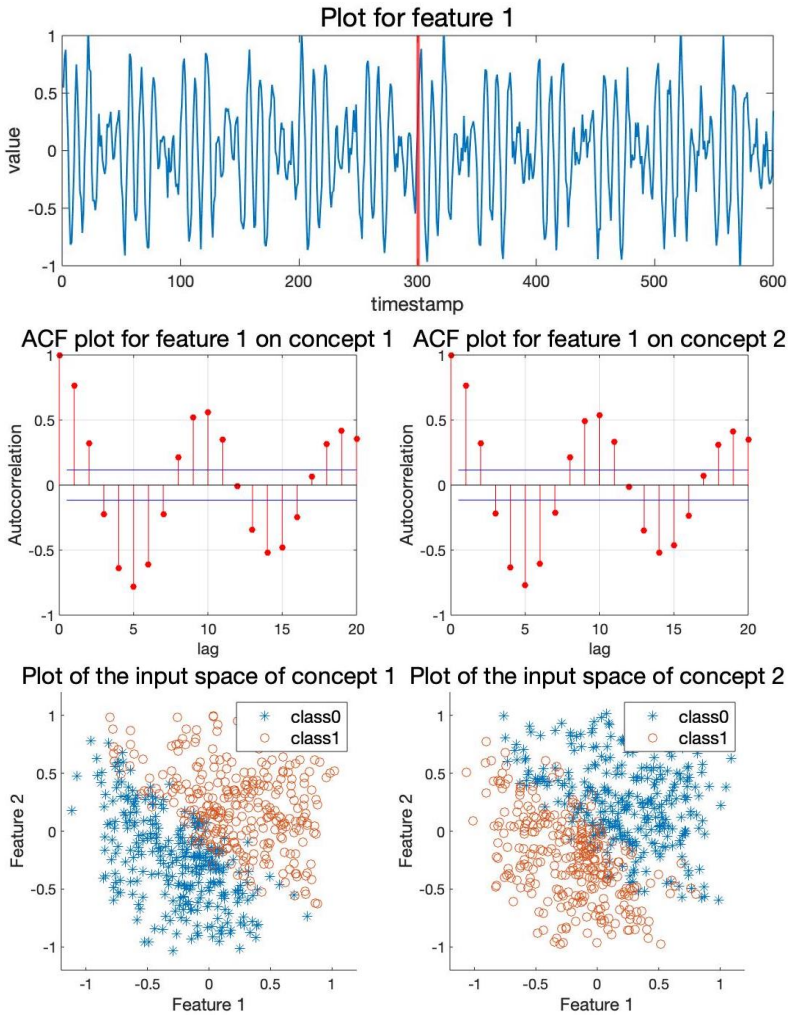


Figure 5.6: An example for **CLF** drift (Drift form 6 in Table 5.3). No obvious change can be spotted from either the feature plot (upper plot) or the ACF plot representing the temporal dependency only (middle plot). The change is only noticeable from the input space plot revealing the class-conditional distributions (bottom plot).

5.3.3 A Temporal Dependency-based Detector CDTD

After discussing the possible sources of drifts under our new problem formulation in Section 5.3.1 and giving some examples of these drifts in Section 5.3.2. In this section, we aim to develop a novel detector named Concept Drift detection for Temporally Dependent data streams (CDTD) that exploits the temporal dependency to detect the wider range of drifts.

Our derivations in equations 5.2 and 5.3 have shown that in order to capture all possible drifts, dependency-related information should be monitored for change detection in this situation. From the discriminative viewpoint, $p(\mathbf{x}_t|\mathbf{x}_{\{t-\tau:t-1\}})$ and $p(y_t|y_{\{t-\tau:t-1\}}, \mathbf{x}_{\{t-\tau+1:t\}})$ should be monitored. It can be noted that the latter probability captures the full dependency, which also contains relevant information of the former probability. Various feature extraction methods can be used in order to keep track of such information. Recurrent neural network (RNN) is a state-of-the-art method used for modelling spatio-temporal information (Wang et al., 2020). A simplified RNN structure, reservoir computing (RC), has been very popular since it is able to represent a wide variety of dynamical characteristics of the data streams (Chen et al., 2014). It can also approach any non-linear mapping with only linear training (Tanaka et al., 2019). They can be trained fast and run in real-time, which makes them suitable tools for online learning scenario.

Given the input signals $\mathbf{s}(t)$ and output signals $o(t)$, the RC model with a reservoir of N units can be formulated as follows: $\mathbf{r}(t) = tanh(W\mathbf{R}(t-1) + U\mathbf{s}(t))$; $f(\mathbf{r}(t)) = V\mathbf{r}(t) + \mathbf{b}$, where $r(t)$ is the state vectors of the reservoir activations, $tanh(\cdot)$ is the state-transition function of the reservoir and $\mathbf{W}$ is an $N \times N$ reservoir weight matrix. U and V are the input and output weight matrices, respectively. $f(\mathbf{r}(t))$ is the output of the RC model and $\mathbf{b}$ is the bias vector for output nodes.

However, the traditional random RC model is largely driven by a series of randomized model building stages, which could lead to very unstable results (Quevedo et al., 2014). Inspired by the fault detection algorithm proposed by Chen et al. (2014), we

Algorithm 3: General framework of CDTD. The green-shaded lines represent three novel modifications made to the existing fault detection algorithm by Chen et al. (2014).

input : initial sequence of k sliding windows, sliding window size m, order of the input τ', counter window size l, decision threshold c^*.

output: confirmed detections

1 **for** *the* i^{th} *sliding window in the initial training sequence* **do**

2 Prepare the input-output data stream to train the reservoir model: at time stamp t, the input is a re-arranged one-dimensional vector $\mathbf{s}(t) = (y_{\{t-\tau':t-1\}}, \mathbf{x}_{\{t-\tau'+1:t\}})$ and the output is $o(t) = y_t$;

3 Fit the reservoir model and record the linear readout f_i.

4 Fit a one-class classifier OCC with $TS = (f_1, ..., f_k)$;

5 **while** *there is incoming data* **do**

6 **for** *each new sliding window arrived at time* t **do**

7 Prepare the input-output stream as in step 2 to derive f_t;

8 Check if f_t belongs to the current concept within OCC;

9 **if** *an* f_t *does not belong to the current concept* **then**

10 Put f_t in the candidate pool that only retains abnormal f_i for $i \in [t-l, t]$;

11 **if** *size of candidate pool* $> l \times c^*$ **then**

12 Record $\widehat{T} = t$ as a confirmed detection;

13 Use members in the candidate pool or wait until enough data (e.g., k linear readouts) are collected for retraining and update TS;

14 Continue from line 4.

15 Output the confirmed detections.

adopt a deterministic reservoir as a tool for temporal dependency extraction and conduct detection within the model space with a one-class classifier. Since we are dealing with supervised data streams, three major modifications are made to adapt our scenario. Firstly, the input-output pair is modified to contain information from both data features stream and the label stream, whereas the existing fault detection algorithm by Chen et al. (2014) does not contain supervised information. Secondly, The RC structure is switched to simple cycle reservoir instead of cycle reservoir with jumps to further reduce instability. Lastly, the candidate pool size is also limited to reduce the possibility of alarms. More detailed description of the exact modifications is presented in sequel.

CDTD also adopts a sliding window-based strategy. One RC model is learnt from each window. Each RC model in CDTD is trained with input $\mathbf{s}(t) = (y_{\{t-\tau':t-1\}}, \mathbf{x}_{\{t-\tau'+1:t\}})^T$ and output $o(t) = y_t$, where τ' can be seen as a pre-defined short-term order the user wishes to consider. The long-term memory is automatically learnt with the reservoir model. Information of both features and labels are used to form a model representation, aiming to capture the full dependency within the data stream. In addition, in order to reduce the number of parameters involved in the model and improve model stability, a simple cycle reservoir (SCR) topology (as shown in Figure 5.7) rather than cycle reservoir with jumps is used. In the SCR reservoir, all cyclic connections have the same weight r_c. The input weights have the same absolute value u. Finally, the existing method by Chen et al. (2014) saves every unknown point in a candidate pool and acknowledge the emergence of a new concept after the size of pool reaches certain threshold. This strategy may lead to an increasing number of false detections, especially when the stable concept is long. In fact, when there is a concept drift, abnormal RC models are expected to appear at a much higher rate. Therefore, to reduce false alarms, we specify a counter window of size l (e.g., 100) which represents the scale of smoothness, and a percentage threshold c^* (e.g., 70%) that represents the internal level of consistency tolerance. At each time stamp t, the candidate pool maintains only the identified unknown points within the latest time frame $[t-l, t]$. A drift is confirmed when the size of the candidate pool exceeds $l \times c^*$.

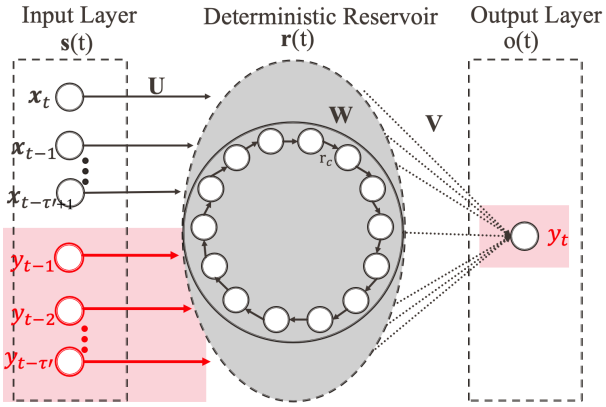


Figure 5.7: Reservoir computing model with simple cycle reservoir (SCR) topology adopted in our proposed CDTD detector.

The pseudo-algorithm is presented in Algorithm 3. CDTD adopts a rolling-window-based approach. One linear readout (model space projection) is learnt with each window. In the beginning, a series of model space projections representing the stationary concept are derived based on an initial stationary segment. Then, a one-class classifier OCC will be constructed based on there projections. With the rolling window moving forward, we continually apply the OCC model to judge whether a new concept is taking place. If the new projection is deemed to come from a possibly different concept by the OCC model, the projection will be sent to the candidate pool. Then, a drift is reported when the size of the candidate pool exceeds the pre-defined threshold $l \times c^*$. After the confirmation of a new concept, a new OCC model is learnt with members in the candidate pool. Note that if the candidate pool size is not large enough to learn a new concept, the reconstruction OCC model may wait for a while until the minimum size for retraining is met. This is also a reasonable approach in the literature based on the assumption that concept drifts will not happen too frequently (e.g., Baena-García et al., 2006).

5.4 Computational Studies

In this section we conduct two sets of experiments. First, based on the synthetically generated data streams, we demonstrate experimentally how well different existing detectors work when they encounter the possible forms of drifts described in the earlier sections. Second, we test the ability of our newly proposed CDTD detector to detect the synthesized drifts, and conduct an ablation study to discuss the effect of various RC model input-output settings on the detection performance.

5.4.1 Experimental Protocols

The synthetic data streams are generated according to the examples of various concept drift forms introduced in Table 5.3. For each form of drift, 30 sequences, each containing 5 concepts with a length of 5000 time stamps, are tested. The details for data generation is as follows. For **TD-XX**, the parameters A, B and C are simply re-sampled from the uniform distribution for each drift and normalized so that the sum of A, B, C is always 1. For **NUM-X**, δ is increased by 1 for each drift. For **TD-XY**, a is randomly sampled from the standard uniform distribution for the first concept, and then incremented by 0.2 for each drift. A similar strategy is applied on b to generate **TD-YY** drift. For **TD-OY**, the order τ is first set to 10 and re-sampled again within $[\tau, 5\tau]$ for each drift. For **CLF**, the change in θ is simply achieved by a class swap after each drift.

The evaluation measures we adopt follow the evaluation definition paradigm in Section 4.3. Detection performance is measured for two acceptable lengths $\Delta = \{1000, 2000\}$ so as to limit the maximum detection delay allowed. All reported recall, precision and F-measure values are calculated with the average TP, FP and FN over the 30 sequences.

It is assumed that the first 1000 data points in the stream come from a stationary concept. For our newly proposed detector CDTD, the SCR topology is used as shown in

Figure 5.7. The connection weight r_c, input weight v and reservoir size N are selected based on a 5-fold cross validation on the first 1000 normal data points. The grid search ranges for the first two weight parameters are both [0.05: 0.05 :1] [1]. Reservoir size is selected from $[150, 200, 250]$. The window size should be larger than the largest temporal scale as well as the number of recurrent units in the reservoir. If the window size is too small, the RC model cannot be learnt well, which may lead to unsatisfactory detection performance. On the other hand, although larger windows can train more accurate RC models, they lead to longer delays in detection times. The selection of the most appropriate window size can be task specific. Preliminary experiments have shown that a window size between 300 and 800 leads to acceptable results in terms of both precision and recall for most drift forms we consider. In the current work, the size of rolling window m is set to 500 with a rolling size of 1. A small rolling size can help detect drifts more quickly at the cost of higher computational cost. Since we do not have any prior knowledge of the actual order τ of the data stream, for simplicity τ' is set to 1, meaning that we mostly account on the reservoir to keep track of the memory information. Then the input and output of the RC model are simply $\mathbf{s}(t) = (x_t^1, x_t^2, y_{t-1})^T$ and $o(t) = y_t$. One-class SVM (Khan and Madden, 2010) is used as the one-class classifier in CDTD. Different comparative methods are utilized for each experiment and will be introduced later. Configuration of the comparative methods follow the setting in their original papers unless otherwise stated.

5.4.2 Experiment 1: How Well Existing Methods Perform

In this section, we first apply some existing detection methods on our newly synthesized data streams with temporal dependency (Table 5.3). For data-based detectors, HCDT (Alippi et al., 2016) and Kdq-tree (Dasu et al., 2006) are selected. HCDT with the ICI-based CDT as the base detector has been shown to achieve very accurate and efficient in

[1] specified in MATLAB notation: [s: d: e] denotes a series of numbers starting from s, increased by increments of d, until the ceiling e is reached.

Table 5.4: Detection performance for CDTD and other existing detection methods with an acceptable delay length of 1000.

Detector	CDTD			Kdq-Tree			HCDT		
Drift Type	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
1. **TD-XX**	0.8657	0.9167	0.9016	0.0000	0.0000	NaN	0.3684	0.0583	0.1007
2. **NUM-X**	0.9254	1	0.9191	0.5178	0.8507	0.6438	0.9524	1.0000	0.9756
3. **TD-XY**	0.8657	0.9667	0.9134	NaN	0.0000	NaN	0.0000	0.0000	NaN
4. **TD-YY**	0.8197	0.9083	0.8617	NaN	0.0000	NaN	0.2500	0.0084	0.0163
5. **TD-OY**	0.8516	0.9730	0.8906	NaN	0.0000	NaN	0.2500	0.0084	0.0163
6. **CLF**	0.8138	0.9833	0.9375	NaN	0.0000	NaN	1.0000	0.0250	0.0488
Detector	DDM			HLFR			HRDD		
Drift Type	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
1. **TD-XX**	0.1000	0.0583	0.0737	NaN	0	NaN	0.5789	0.4583	0.5116
2. **NUM-X**	0.7101	1	0.8305	1.0000	0.9000	0.9744	0.9600	1.0000	0.9796
3. **TD-XY**	0.1000	0.1417	0.1173	NaN	0.0000	NaN	0.6154	0.1333	0.2191
4. **TD-YY**	0.1711	0.1083	0.1326	NaN	0.0000	NaN	0.3704	0.0833	0.1360
5. **TD-OY**	0.3696	0.5083	0.4280	1.0000	0.1000	0.1818	0.4576	0.2250	0.3017
6. **CLF**	0.7843	1.0000	0.8791	0.9744	0.9500	0.9620	0.7480	0.7667	0.7572

previous works. Kdq-tree based method (Dasu et al., 2006) is a distance-based approach that does not explicitly require the data to be independent. For classifier-based methods, among the two benchmark detectors DDM and EDDM, DDM is picked since it is more suitable for the detection of abrupt drifts (Baena-García et al., 2006). The latest HFLR (Sun et al., 2021) introduced in Section 2.1.2 is also used. Note that the detection performance can also be affected by the choice of base classifier, hence two classifiers, SVM and decision tree, are tested. The classifier-detector combination achieving the best result for F-measure is presented. In addition to existing methods in the literature, HRDD from Chapter 4 is also selected as a comparative method.

Table 5.4 shows the precision, recall and F-measure of the comparative methods

with an acceptable delay length of 1000 time stamps. For comparison purposes, the outcome of CDTD is also presented here, but more detailed analysis of CDTD will be presented in Experiment 2. Similar results are obtained with an acceptable delay length of 2000. The only form of change that can be accurately identified by data-based methods is **NUM-X**. For this form of drift, although the HCDT is operating under conditions where the independence assumption is violated, this does not necessarily implies that it will always fail. The change in the feature mean is so strong that it can still be easily captured by existing data-based methods.

Although **TD-XX** is also a change affecting the data feature stream, this form of drift affects the temporal dependency only and leave the mean and variance of the features unchanged. Existing data-based detectors such as HCDT and Kdq-tree are not designed to capture this form of change, hence they are more likely to fail. This observation illustrates that data-based detectors monitoring $p(\mathbf{x}_t)$ are not appropriate for temporally dependent data streams. For the rest forms of change, the feature streams are not altered in any ways so detection always fails.

Classifier-based methods fail for 4 of the 6 forms of drifts. This result coincides with our expectation that monitoring $p(y_t|\mathbf{x}_t)$ is also insufficient for drift detection in temporally-dependent data streams. We also notice that the issue of temporal dependency is not limited to the violation of the independence assumption of the error stream. More importantly, the errors themselves are no longer trustworthy. Due to the existence of temporal dependency, data points with exactly the same features $\mathbf{x}_t$ at time stamp t may have different true labels y_t since the influence of previous data points are also taken into account. Such results confuse the classifier during training. Thus, the pattern of classification outcomes is no longer reliable even for a stationary data stream. However, they may still be able to detect partial drifts when the there is a noticeable change in the learnt decision boundary. In other words, as long as drifts can be reflected by a significant change in classification performance, no matter how poor or good the original performance is, they can still be captured. This is especially the case when the change

severely affect the decision boundary, e.g., for **NUM-X** and **CLF** drifts. These two forms of drifts are the most typical drifts considered in existing work. For particular forms of drifts, our finding coincide with the conclusion drawn by Zliobaite et al. (2015) that temporal dependency does not do much harm to the detection performance. In a temporally dependent environment, they can still be detected with reasonable precision and recall, even if the theoretical guarantees of the tests (if any) are not valid any more. Nonetheless, this conclusion is not true for other forms of drifts that can possibly occur.

Although HRDD has been shown to surpass the comparative methods in scenarios with independent data streams by a great amount in Chapter 4, the advantages no longer hold for drift detection in temporally dependent data streams. Viewing the performance on all 6 forms of drifts, it can be seen that HRDD may still detect some drifts that cannot be detected by purely classifier-based or data-based detectors. However, the detection sensitivity is still far behind CDTD.

In summary, existing methods, no matter data-based methods or classifier-based methods, are not suitable for concept drift detection in temporally-dependent data streams. They cannot cope with this more complicated situation very well. It should also be noted that among the drift examples provided in Table 5.3, the forms of drifts that existing methods always fail are all **TD**-related drifts. This finding signals the necessity of investigating new concept drift detection tools that can accommodate this more realistic situation. Our newly proposed baseline detection method CDTD is capable of detecting all 6 forms of drifts presented and performs the best in terms of F-measure among all the methods being compared. The next set of experiment will have a closer look at the functionality of CDTD and the key requirement to guarantee its success.

5.4.3 Experiment 2: Functionality of CDTD

This experiment examines the performance of CDTD to handle various forms of drifts in temporally-dependent data streams. As introduced in Section 5.3.3, the current input-

Table 5.5: Various input-output setting for the reservoir model in CDTD under comparison in Experiment 2.

Method	Reservoir input	Reservoir output
CDTD	$\mathbf{s}(t) = (y_{t-1}, ...y_{t-\tau'}, x_t^1, ..., x_t^d, ..., x_{t-\tau'+1}^1, ..., x_{t-\tau'+1}^d)^T$	$o(t) = y_t$
CDTD-XX	$\mathbf{s}(t) = (x_{t-1}^1, ..., x_{t-1}^d, ..., x_{t-\tau'}^1, ..., x_{t-\tau'}^d)^T$	$o(t) = (x_t^1, ..., x_t^d)^T$
CDTD-XY	$\mathbf{s}(t) = (x_t^1, ..., x_t^d)^T$	$o(t) = y_t$

output setting for the reservoir model in CDTD is designed to capture the full temporal dependency $p(y_t|y_{\{t-\tau:t-1\}}, \mathbf{x}_{\{t-\tau+1:t\}})$ within each window. The reservoir takes some previous features, current features as well as some previous labels as inputs. In this experiment, we also demonstrate that capturing the full dependency within the data stream is necessary for the detection of the wide range of possible drifts. We compare the current setting with two different input-output settings of CDTD (Table 5.5). CDTD-XX considers input and output based on the features only. CDTD-XY takes the usual input-output setting of existing classification models and can be regarded to be a similar realization of the work by Chen et al. (2014). To be fair, τ' is set to 1 for all the settings compared here.

Table 5.6: Detection performance for various input-output settings of the reservoir in CDTD with an acceptable delay length of 1000.

Detector	CDTD			CDTD-XX			CDTD-XY		
Data Name	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
1. **TD-XX**	0.8657	0.9167	0.9016	0.8096	0.7083	0.7556	0.6087	0.7000	0.6512
2. **NUM-X**	0.9254	1.0000	0.9191	0.8440	0.9917	0.9119	0.7581	0.7833	0.7705
3. **TD-XY**	0.8657	0.9667	0.9134	NaN	0.0000	NaN	0.5053	0.8083	0.6219
4. **TD-YY**	0.8197	0.9083	0.8617	NaN	0.0000	NaN	0.6087	0.5833	0.5957
5. **TD-OY**	0.8516	0.9730	0.8906	NaN	0.0000	NaN	0.6667	0.6333	0.6496
6. **CLF**	0.8138	0.9833	0.9375	NaN	0.0000	NaN	0.7143	0.1000	0.8333

Table 5.6 presents the detection results for the methods compared. The first conclusion from this table is that the current input-output setting of CDTD exploiting the full dependency within the data stream performs the best among the three settings being compared in terms of F-measure.

CDTD-XX and CDTD-XY can be regarded as data-based and classifier-based detectors, respectively. Nonetheless, it should be noted that they are different from existing methods since the reservoir model structure processes and records temporal patterns. Comparing Table 5.6 and Table 5.4, it can be seen that by relying on the temporal dependency-related information for detection, CDTD-XX surpasses existing data-based detectors in the detection of **TD-XX** drifts and achieves comparative results as existing data-based detectors in the detection of **NUM-X** drifts. This result evidently shows the importance of utilizing dependency for drift detection.

Similarly, CDTD-XY can also perform better than existing classifier-based detectors for all **TD**-related drifts. Nonetheless, there is still a large gap between the performance of CDTD-XY and CDTD for all 6 forms of drifts. The results confirmed the importance of incorporating previous information from both the features stream and the label stream for precise drift detection. Hence, we conclude that recording and utilizing the full dependency for drift detection is the key to guarantee the superiority of CDTD. Moreover, the improved precision achieved by CDTD also demonstrates that our modified drift detection rule can help lead to more precise drift detection by reducing false alarms.

5.5 Chapter Summary

Initially, we demonstrate that temporal dependency is a crucial problem that significantly impacts the current understanding of concept drift. Our study then introduces a new problem formulation that considers the interaction between these two issues. This

formulation also leads to a novel taxonomy of various forms of concept drift that can possibly occur. Subsequently, we conduct experiments to reveal some of the situations where existing detectors fail in the presence of temporal dependency. In order to simulate different forms of drifts on temporal-dependent data streams, we devise a new and adaptable data stream generator. The results of this chapter highlighted the need for immediate attention to certain forms of drift, and led us to conclude that detecting concept drift in temporal-dependent data streams is a subject that needs further exploration. Finally, we propose a benchmark solution for drift detection in temporal-dependent data streams and evaluated its effectiveness on multiple sets of data streams.

As in Chapter 4, the parameters of the existing drift detectors were the same as the ones used in their original papers. The nature of online classification scenarios does not allow parameter tuning since no post-drift information is available. Future work may consider testing and comparing the methods with various parameter settings. Another piece of crucial future work is to conduct experimentations with real-world data streams. In existing literature, the number of detections and the prequential classification error are usually examined for assessing concept drift detectors applied on real-world data streams. However, it should be noted that a low number of detections may no longer be appreciated since this may be caused by the neglection of TD-related drifts. Experiments with real-world data streams have to be carefully designed and evaluated. In addition, in the current work, a RC model and a simple one-class SVM are selected to form CDTD, which we regard as a simple baseline detector only. Other effective combinations may also be considered for various application systems. As long as the full dependency within the data stream is captured, TD-related drifts should also be detected.

Chapter Six

Conclusions and Future Work

The focus of this thesis is the detection of changes in both offline and online classification tasks. Conventional classification is based on the identically and independently distributed ($i.i.d$) assumption. However, the requirement of a static environment may not be satisfied once the model is deployed in real-world applications. This thesis mainly investigates *How to accurately detect a change that violates the identically-distributed assumption of classification models?*. The detection of such changes is crucial to ensure the safety deployment of machine learning systems, since it allows in-time actions to be taken to maintain the expected performance.

Since classification can be conducted in both offline and online forms, this thesis divides the main target into two research areas, namely OoD detection and concept drift detection for offline and online scenarios respectively. The key research question in the offline scenario is *How to accurately detect testing examples that do not come from a same distribution as the training data set?*. For online scenario, this thesis investigates *How to accurately detect if and when a change has occurred in the underlying streaming environment?*. In order to find the solutions for these two questions, several specific research questions have been listed in the Introduction Chapter.

This chapter summarizes the contributions of the thesis, with an emphasis on how each proposed research question is addressed. Some directions for future work are also

identified at the end of this chapter.

6.1 Conclusions

The thesis starts with the offline scenario. In offline scenarios, OoD testing examples that do not belong to the distribution of the training data set need to be identified.

Although a plethora of detection OoD methods have been developed, many of the methods have assumptions on the OoD data and rely on some synthetic or real OoD examples for training the detector. Purely unsupervised methods are very limited. Besides, the stability and reliability of existing unsupervised methods have been shown to be poor for some particular types of OoD examples. For instance, semantic OoDs have been shown by existing work to be the hardest type of OoDs to detect. Therefore, the thesis aims to provide solutions to the following research questions listed in Section 1.2.1: *How to detect various types of OoD examples more accurately in a purely unsupervised manner? How to characterize different types of OoDs more comprehensively and improve the detection performance for the more difficult OoDs?*.

In Chapter 3, we present our new OoD detection method LAMAE based on a memory AE to answer the first question. The key idea is to regularize the AE network architecture with a classifier and a label-assisted memory. The information of the class-labels of ID data is leveraged to confine the reconstruction of OoD data while retaining the reconstruction ability for ID data. Thus, the OoD detection performance can be improved. In order to answer the second question, we first refine the existing categorization with respect to OoD semantics only by adding a new dimension accounting inherent image complexity. Testing examples that are less complex than the ID training examples are more likely to be missed by the reconstructed-based detector. By noticing the strong impact of image complexity on the reconstruction errors, we propose to adjust the test statistic by applying a complexity-based normalizer before using it for detection, forming

the detector LAMAE+. Experimental studies show that the proposed OoD detectors can perform well on a wider range of OoD scenarios, including the difficult semantic OoDs.

After viewing the possible change in offline case, the online scenario is closely examined. Data streams often arrive from an evolving and dynamic environment. Changes in this case are generally more difficult to detect than in the offline case. Existing concept drift detection tools focus on either real drifts deteriorating the classification performance or virtual drifts altering the input space only. While both types of drifts are worth detection since they signal different information about the data stream, there is a lack of method to detect both types simultaneously. Besides, existing detection methods targeting virtual drifts cannot well accommodate high-dimensional data streams due to high computational costs and low efficiency. Hence, the following research questions are raised in Section 1.2.2: *How to detect both real and virtual drifts in supervised data streams regardless of their effect on classification performance? How to improve the efficiency of data distribution-based detector for high-dimensional data streams? How to improve detection performance to achieve high true detections and low false alarms within a specified delay range for all types of drifts even when the magnitude of drift is small?*

In Chapter 4, we present the HRDD detection framework to detect both real and virtual drift accurately and efficiently for multi-dimensional data streams to answer the listed research questions. The key idea is to leverage the knowledge from supervised information to discover changes that may not be detected by the existing detection methods. To achieve this goal, firstly, a lower-dimensional feature space for the given classification task is explicitly constructed. Each incoming data is projected to this space upon arrival, where more efficient detection can be carried out. Next, we monitor not only the marginal distribution of the data stream, but also each individual class-conditional distribution so that even the subtle drifts affecting partial input space can be detected. Finally, a novel method to reconfigure more informative retraining datasets after each detection is presented. HRDD can be used in conjunction with any base CDT and classifier, and the performance is independent of the choice of the classifier.

After proposing HRDD and testing it on real-world data streams, we notice that the validity of many existing concept drift detectors is based on the assumption of data independence. Nonetheless, most real-world data streams contain temporal structures spanning over different period range. In fact, the problem of temporal dependency and concept drift in data streams cannot be separately investigated. In order to better understand the joint issue, the following research questions are proposed in Section 1.2.3 and answered in Chapter 5: *If and how temporal dependency affects existing concept drift detection methods? What types of drifts can possibly occur under such scenario? How to detect the wide range of possible drifts for temporal dependent data streams?*

With this current work, we not only formally show the reasons behind possible failures of existing methods with theoretical explanations, but also demonstrate the results experimentally with the simulated data streams, answering the first research question. Then, to answer the second research question, we provide a novel problem formulation for drift detection within the temporal-dependent environment taking into account the interplay between the two issues. Following this, we discuss how their interactions bring new challenges to the current state of research and also construct a novel taxonomy to illustrate that a wider range of various forms of concept drifts are likely to take place. Finally, to answer the last research question, we develop a baseline detector that relies on reservoir computing to extract temporal dependency, and utilize this dependency to achieve better drift detection performance in temporally dependent data streams.

6.2 Future Work

This section lists out several directions for future research in both offline (Section 6.2.1) and online (sections 6.2.2, 6.2.3, 6.2.4) scenarios that may improve and extend work presented in this thesis.

6.2.1 The Role of Image Complexity in AE Reconstruction

Chapter 3 has demonstrated that for image datasets, the impact of image complexity on reconstruction-based OoD detection methods is unneglectable. One possible avenue is to explore why image complexity plays such an important role in AE reconstructions. Recall that we have demonstrated with the extreme example of constant OoD images (i.e., with same-valued pixels) that images with such a low complexity can almost always be reconstructed well, even if they have never been seen by the AE model before. Although we noticed and summarized this phenomenon based on the observations from an empirical study, and also provided a trick to remove the bias before carrying out OoD detection, an in-depth explanation of the reason behind the phenomenon is still absent.

One possible reason is that AEs are designed for accurate pixel-level image reconstruction, which may conflict with the goal of distinguishing between ID and OoD examples. When and why will the basic assumption of AE-based detectors fail? What is the key reason behind the failure? Is there any theoretical implication behind the phenomenon?

Future work may start with carefully analysing the manifolds learnt with AEs and evaluating their relationship with OoD examples of various degrees of complexity. A suitable latent space that is sensitive to change, rather than the latent space for the best reconstruction, may be constructed for better OoD detection. Once this investigation is complete, more task-specific amendments can be made to the AE structure and loss function to make them more suitable for OoD detection. Then, new solutions as to the design of more explainable and robust OoD detectors that can detect OoD examples with various complexities equally well can be proposed.

6.2.2 Further Analysis of Concept Drift in Temporal-dependent Data Streams

We have verified in Chapter 5 of this thesis that the interplay between temporal dependence and concept drift detection is a research area worth further exploration. As the first work discussing the joint issue, there is still a lot of room to further discuss and improve the current analysis.

Firstly, we have only provided a novel taxonomy to categorize the different sources of drifts in this scenario in Section 5.3.1. It can be clearly identified that the source of drifts being considered here can be very different from the previous work neglecting the dependency. We would like to investigate whether there are other systematic ways to categorize drifts in temporal dependent data streams. Is it possible to propose quantitative measures for such drifts? Is it possible to extend existing categorizations evaluating various characteristics of the drifts to this situation? For examples, can drifts in this particular scenario also be divided into real and virtual drifts?

Secondly, we only proposed one data generator based on sine and cosine functions in Section 5.3.2. Although a noise is added to the data streams, it may be argued that these data streams are too simple due to their periodic nature. Besides, we only provided one set of examples of possible drifts that may occur. In order to promote the research in this direction, we will design various data generator aiming to simulate real-world data streams even better.

Thirdly, the current detection method proposed in Section 5.3.3 is only a baseline method, revealing the importance and effectiveness of considering temporal dependency in concept drift detection. There should be other ways of constructing the detector for various application systems as well, which is also part of our future work. For instance, we aim to propose methods that can detect drifts regardless of whether temporal dependency exists or not. Such a detector would be more useful in the real world where not enough data is available to analyse the dependency within the data stream. This goal can be

achieved, for example, with an ensemble detector consisting of both our newly proposed CDTD and HRDD.

6.2.3 Drift Detection on Spatio-temporally Dependent Data Streams

In Chapter 5, the focus of attention is only drawn to temporal dependency that summarizes within-feature dependency. Many real-world data streams, by their very nature, also contain correlated features (i.e., between-feature dependency). Within-feature and between-feature dependency can coexist, which is named as the spatio-temporal dependency. In this case, the problem formulation in Section 5.3.1 describing data streams with stochastic processes may need to be further adjusted.

In addition, we would also like to examine if any new forms of drifts can possibly occur in spatio-temporally dependent supervised data streams. If so, what forms of existing detectors are likely to fail when dealing with this scenario? How to accurately detect such drifts? Future research can also be carried out based on this set of research questions.

6.2.4 Recurrent Concept Drift Detection and Drift Prediction

It is common in real-world data streams that previously seen concepts reappear, which is known as recurring concept drift. Existing strategies dealing with general concept drifts mainly discard previous information and rebuild new models when drift occurs, which may lead to unnecessarily high computational costs and longer time for performance restoration when recurring concepts exist. Concept recurrence poses new challenges for online learning as well as new opportunities to develop more efficient learning algorithms.

The advantage of detecting recurrent concept drift is not limited to a better understanding of the system generating the data, but also a smoother and quicker adaptation to restore the system performance. A probabilistic model can be used to learn drift trends

so that drift prediction can be carried out. Although there have been some studies on recurrent drift detection (Geilke et al., 2015; Sakthithasan et al., 2015; Chen et al., 2016; Reis et al., 2018), none of them are designed for temporally dependent data streams. Hence, they cannot capture drifts related to temporal dependency change. In the future, we will investigate if existing recurrent drift detection methods can be extended to handle temporally-dependent data streams, and design suitable drift prediction methods to further improve overall performance.

Appendix One

A Summary of Performance Metrics for Assessing Concept Drift Detectors

Table A.1 summarizes some of the performance metrics and their definitions of existing methods. It can be shown that various definitions have been used even for the same performance metrics, symbolizing various focus of the proposed detectors.

Table A.1: Summary of different definitions of concept drift detection performance evaluation

Reference	Performance metrics	Definition
Kim and Park, 2017	TP	the first detection after the actual drift
	FP	detections before the actual drift
	FN	no detection after the actual drift
Alippi et al., 2011a	TP	at least one detection is after the actual drift and before end of concept
	FP	at least one detection is before the actual drift
	FN	no detection at all

Table A.1 Summary of different definitions of concept drift detection performance evaluation (Continued.)

Kifer et al., 2004 Dasu et al., 2006 Lu et al., 2014 Gu et al., 2016	Correct	at least one detection is raised within a fixed number of window length (1 or 2) after the actual drift
	Late	a detection raised after the above range
	False	detections raised before the actual change
	Missed	no detection at all
Harel et al., 2014	TP	a detection within a fixed number of window's range after the change time
	FP	a detection outside the range or an extra detection in the range
	FN	no detection within the delay range defined as above
Bifet and Gavalda, 2007 Huang et al., 2015	TP	a detection is after the real drifting time
	FP	a detection on a stationary data stream
	time to FP	time till the occurrence of the first FP on a stationary data stream
Pesaranghader and Viktor, 2016	TP	a detection is within the acceptable detection interval $[T^* - \triangle, T^* + \triangle]$
	FP	detections raised outside the acceptable detection interval before real drifting time i.e. outside $[T^* - \triangle, T^* + \triangle]$
	FN	the first detection for the concept is beyond the acceptable detection interval $[T^* - \triangle, T^* + \triangle]$

Table A.1 Summary of different definitions of concept drift detection performance evaluation (Continued.)

Dehghan et al., 2016	TP	the first detection after the actual drift
	FP	detections where no drift occurred (detections before the drift or after TP)

References

Abdelzad, Vahdat et al. (2019). "Detecting out-of-distribution inputs in deep neural networks using an early-layer output". In: *arXiv preprint arXiv:1910.10307*.

Abdi, Hervé and Lynne J Williams (2010). "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4, pp. 433–459.

Alippi, C, G Boracchi, and M Roveri (2016). "Hierarchical change-detection tests". In: *IEEE Transactions on Neural Networks & Learning Systems* 28.2, pp. 246–258.

Alippi, Cesare and Manuel Roveri (2008). "Just-in-time adaptive classifiers?Part I: Detecting nonstationary changes". In: *IEEE Transactions on Neural Networks* 19.7, pp. 1145–1153.

Alippi, Cesare, Giacomo Boracchi, and Manuel Roveri (2010a). "Adaptive classifiers with ICI-based adaptive knowledge base management". In: *International Conference on Artificial Neural Networks*. Springer, pp. 458–467.

— (2010b). "Change detection tests using the ICI rule". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–7.

— (2011a). "A hierarchical, nonparametric, sequential change-detection test". In: *The 2011 International Joint Conference on Neural Networks*. IEEE, pp. 2889–2896.

— (2011b). "A just-in-time adaptive classification system based on the intersection of confidence intervals rule". In: *Neural Networks* 24.8, pp. 791–800.

— (2013). "Just-in-time classifiers for recurrent concepts". In: *IEEE transactions on neural networks and learning systems* 24.4, pp. 620–634.

— (2017). "Hierarchical change-detection tests". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.2, pp. 246–258.

Aminikhanghahi, Samaneh and Diane J. Cook (2017). "A survey of methods for time series change point detection". In: *Knowledge and Information Systems* 51.2, pp. 339–367.

An, Jinwon and Sungzoon Cho (2015). "Variational autoencoder based anomaly detection using reconstruction probability". In: *Special Lecture on IE* 2.1, pp. 1–18.

Andrews, Jerone TA, Edward J Morton, and Lewis D Griffin (2016). "Detecting anomalous data using auto-encoders". In: *International Journal of Machine Learning and Computing* 6.1, p. 21.

Antwi, Daniel K, Herna L Viktor, and Nathalie Japkowicz (2012). "The PerfSim algorithm for concept drift detection in imbalanced data". In: *2012 IEEE 12th International Conference on Data Mining Workshops*. IEEE, pp. 619–628.

Archive, UCI KDD (2022). *Information and Computer Science, University of California, Irvine, CA, USA*. https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Online; accessed 2022].

Baena-García, Manuel et al. (2006). "Early Drift Detection Method". In: *Proceedings of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS'06)*.

Barros, Roberto SM, Danilo RL Cabral, Paulo M Gonçalves Jr, and Silas GTC Santos (2017). "RDDM: reactive drift detection method". In: *Expert Systems with Applications* 90, pp. 344–355.

Barros, Roberto Souto Maior de, Juan Isidro González Hidalgo, and Danilo Rafael de Lima Cabral (2018). "Wilcoxon rank sum test drift detector". In: *Neurocomputing* 275, pp. 1954–1963.

Bartlett, Peter L, Shai Ben-David, and Sanjeev R Kulkarni (2000). "Learning changing concepts by exploiting the structure of change". In: *Machine Learning* 41.2, pp. 153–174.

Beck, Nathaniel, Jonathan N Katz, and Richard Tucker (1998). "Taking Time Seriously: Time-series-cross-section Analysis with a Binary Dependent Variable". In: *American Journal of Political Science* 42.4, pp. 1260–1288.

Benenson, Rodrigo, Markus Mathias, Tinne Tuytelaars, and Luc Van Gool (2013). "Seeking the strongest rigid detector". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3666–3673.

Berkhahn, Felix et al. (2019). "Augmenting Variational Autoencoders with Sparse Labels: A Unified Framework for Unsupervised, Semi-(un) supervised, and Supervised Learning". In: *arXiv preprint arXiv:1908.03015*.

Bifet, Albert and Ricard Gavalda (2007). "Learning from time-changing data with adaptive windowing". In: *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, pp. 443–448.

Bifet, Albert et al. (2013a). "Pitfalls in benchmarking data stream classification and how to avoid them". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 465–479.

Bifet, Albert et al. (2013b). "Pitfalls in Benchmarking Data Stream Classification and How to Avoid Them". In: *Proceedings of Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*. Ed. by Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezný. Vol. 8188. Lecture Notes in Computer Science. Springer, pp. 465–479.

Blythe, Duncan AJ, Paul Von Bunau, Frank C Meinecke, and Klaus-Robert Muller (2012). "Feature extraction for change-point detection using stationary subspace analysis". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.4, pp. 631–643.

Boult, Terrance E et al. (2019). "Learning and the unknown: Surveying steps toward open world recognition". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 9801–9807.

Boyd, Kendrick, Kevin H Eng, and C David Page (2013). "Area under the precision-recall curve: point estimates and confidence intervals". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer, pp. 451–466.

Brzezinski, Dariusz and Jerzy Stefanowski (2013). "Reacting to different types of concept drift: the accuracy updated ensemble algorithm". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.1, pp. 81–94.

Brzezinski, Dariusz and Jerzy Stefanowski (2014). "Prequential AUC for classifier evaluation and drift detection in evolving data streams". In: *International Workshop on New Frontiers in Mining Complex Patterns*. Springer, pp. 87–101.

Bu, Li, Cesare Alippi, and Dongbin Zhao (2016). "A pdf-free change detection test based on density difference estimation". In: *IEEE Transactions on Neural Networks and Learning Systems*.

Bu, Li, Dongbin Zhao, and Cesare Alippi (2017). "An incremental change detection test based on density difference estimation". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.10, pp. 2714–2726.

Bucilua, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). "Model compression". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541.

Bulatov, Yaroslav (2020). "NotMNIST dataset, 2011". In: *URL http://yaroslavvb. blogspot. com/2011/09/notmnist-dataset. html*.

Cai, Mu and Yixuan Li (2023). "Out-of-distribution detection via frequency-regularized generative models". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5521–5530.

Chen, Huanhuan, Peter Tiňo, Ali Rodan, and Xin Yao (2014). "Learning in the Model Space for Cognitive Fault Diagnosis". In: *IEEE Trans. Neural Networks Learn. Syst.* 25.1, pp. 124–136.

Chen, Jiefeng et al. (2021). "Atom: Robustifying out-of-distribution detection using outlier mining". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 430–445.

Chen, Kylie, Yun Sing Koh, and Patricia Riddle (2016). "Proactive drift detection: predicting concept drifts in data streams using probabilistic networks". In: *IJCNN*. IEEE, pp. 780–787.

Chen, Wenhu, Yilin Shen, Hongxia Jin, and William Wang (2018). "A variational dirichlet framework for out-of-distribution detection". In: *arXiv preprint arXiv:1811.07308*.

Dasu, Tamraparni, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi (2006). "An information-theoretic approach to detecting changes in multi-dimensional data streams". In: *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer.

Dehghan, Mahdie, Hamid Beigy, and Poorya ZareMoodi (2016). "A novel concept drift detection method in data streams using ensemble classifiers". In: *Intelligent Data Analysis* 20.6, pp. 1329–1350.

Denouden, Taylor et al. (2018). "Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance". In: *arXiv preprint arXiv:1812.02765*.

DeVries, Terrance and Graham W Taylor (2018). "Learning Confidence for Out-of-Distribution Detection in Neural Networks". In: *arXiv preprint arXiv:1802.04865*.

Diers, Jan and Christian Pigorsch (2022). "Out-of-Distribution Detection Using Outlier Detection Methods". In: *International Conference on Image Analysis and Processing*. Springer, pp. 15–26.

Ditzler, Gregory and Robi Polikar (2011). "Hellinger distance based drift detection for nonstationary environments". In: *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*. IEEE, pp. 41–48.

Dong, Fan, Jie Lu, Kan Li, and Guangquan Zhang (2017). "Concept drift region identification via competence-based discrepancy distribution estimation". In: *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. IEEE, pp. 1–7.

Drummond, Nick and Rob Shearer (2006). "The open world assumption". In: *eSI Workshop: The Closed World of Databases meets the Open World of the Semantic Web*. Vol. 15, p. 1.

Duong, Quang-Huy, Heri Ramampiaro, and Kjetil Nørvåg (2018). "Applying temporal dependence to detect changes in streaming data". In: *Appl. Intell.* 48.12, pp. 4805–4823.

Elwell, Ryan and Robi Polikar (2011). "Incremental learning of concept drift in nonstationary environments". In: *IEEE Transactions on Neural Networks* 22.10, pp. 1517–1531.

Erdil, Ertunc, Krishna Chaitanya, and Ender Konukoglu (2020). "Unsupervised out-of-distribution detection using kernel density estimation". In: *arXiv preprint arXiv:2006.10712*.

Faithfull, William J, Juan J Rodríguez, and Ludmila I Kuncheva (2019). "Combining univariate approaches for ensemble change detection in multivariate data". In: *Information Fusion* 45, pp. 202–214.

Fawcett, Tom (2006). "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8, pp. 861–874.

Frias-Blanco, Isvani et al. (2014). "Online and non-parametric drift detection methods based on Hoeffding's bounds". In: *IEEE Transactions on Knowledge and Data Engineering* 27.3, pp. 810–823.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR, pp. 1050–1059.

Gama, Joao, Pedro Medas, Gladys Castillo, and Pedro Rodrigues (2004). "Learning with drift detection". In: *Brazilian Symposium on Artificial Intelligence*. Springer, pp. 286–295.

Gama, João, Raquel Sebastião, and Pedro Pereira Rodrigues (2009). "Issues in evaluation of stream learning algorithms". In: *Proceedings of the 15th ACM SIGKDDO International Conference on Knowledge Discovery and Data Mining*, pp. 329–338.

Gama, João et al. (2014). "A survey on concept drift adaptation". In: *ACM computing surveys (CSUR)* 46.4, p. 44.

Gao, Jing, Wei Fan, Jiawei Han, and Philip S Yu (2007). "A general framework for mining concept-drifting data streams with skewed distributions". In: *Proceedings of the 2007 siam international conference on data mining*. SIAM, pp. 3–14.

Gao, Peichao, Zhilin Li, and Hong Zhang (2018). "Thermodynamics-based evaluation of various improved Shannon entropies for configurational information of gray-level images". In: *Entropy* 20.1, p. 19.

Geilke, Michael, Andreas Karwath, and Stefan Kramer (2015). "Modeling recurrent distributions in streams using possible worlds". In: *DSAA*. IEEE, pp. 1–9.

Gong, Dong et al. (2019). "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1705–1714.

Good, Phillip (2013). *Permutation tests: a practical guide to resampling methods for testing hypotheses*. Springer Science & Business Media.

Goodfellow, Ian et al. (2014). "Generative adversarial networks". In: *Communications of the ACM* 63.11, pp. 139–144.

Gu, Feng, Guangquan Zhang, Jie Lu, and Chin-Teng Lin (2016). "Concept drift detection based on equal density estimation". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 24–30.

Guo, Jia, Guannan Liu, Yuan Zuo, and Junjie Wu (2018). "An anomaly detection framework based on autoencoder and nearest neighbor". In: *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*. IEEE, pp. 1–6.

Haque, Ahsanul et al. (2016). "Efficient handling of concept drift and concept evolution over stream data". In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, pp. 481–492.

Härdle, Wolfgang and Léopold Simar (2007). *Applied multivariate statistical analysis*. Vol. 22007. Springer.

Harel, Maayan, Shie Mannor, Ran El-Yaniv, and Koby Crammer (2014). "Concept drift detection through resampling". In: *International Conference on Machine Learning*, pp. 1009–1017.

Harries, Michael and New South Wales (1999). "Splice-2 comparative evaluation: Electricity pricing". In: *Technical Report*.

Hart, Peter E, David G Stork, and Richard O Duda (2000). *Pattern classification*. Wiley Hoboken.

Heer, Matthäus et al. (2021). "The ood blind spot of unsupervised anomaly detection". In: *Medical Imaging with Deep Learning*. PMLR, pp. 286–300.

Hendrycks, Dan and Kevin Gimpel (2017). "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *ICLR*.

Hendrycks, Dan, Mantas Mazeika, and Thomas Dietterich (2018). "Deep Anomaly Detection with Outlier Exposure". In: *International Conference on Learning Representations*.

Hilas, Constantinos S, Ioannis T Rekanos, and Paris Ast Mastorocostas (2013). "Change Point Detection in Time Series Using Higher-Order Statistics: A Heuristic Approach". In: *Mathematical Problems in Engineering* 2013.

Hoeffding, Wassily (1994). "Probability inequalities for sums of bounded random variables". In: *The collected works of Wassily Hoeffding*. Springer, pp. 409–426.

Hoens, T Ryan, Robi Polikar, and Nitesh V Chawla (2012). "Learning from streaming data with concept drift and imbalance: an overview". In: *Progress in Artificial Intelligence* 1.1, pp. 89–101.

Hsu, Yen-Chang, Yilin Shen, Hongxia Jin, and Zsolt Kira (2020). "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10951–10960.

Huang, David Tse Jung, Yun Sing Koh, Gillian Dobbie, and Russel Pears (2014). "Detecting volatility shift in data streams". In: *2014 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 863–868.

Huang, David Tse Jung, Yun Sing Koh, Gillian Dobbie, and Albert Bifet (2015). "Drift detection using stream volatility". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 417–432.

Huang, Yujia et al. (2019). "Out-of-distribution detection using neural rendering generative models". In: *arXiv preprint arXiv:1907.04572*.

Hulten, Geoff, Laurie Spencer, and Pedro Domingos (2001). "Mining time-changing data streams". In: *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 97–106.

Hwang, Jenq-Neng, Shyh-Rong Lay, and Alan Lippman (1994). "Nonparametric multivariate density estimation: a comparative study". In: *IEEE Transactions on Signal Processing* 42.10, pp. 2795–2810.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. PMLR, pp. 448–456.

Jain, Siddhartha, Ge Liu, Jonas Mueller, and David Gifford (2020). "Maximizing overall diversity for improved uncertainty estimates in deep ensembles". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 4264–4271.

Jiao, Yuchen, Yanxi Chen, and Yuantao Gu (2018). "Subspace change-point detection: A new model and solution". In: *IEEE Journal of Selected Topics in Signal Processing* 12.6, pp. 1224–1239.

Katz-Samuels, Julian, Julia B Nakhleh, Robert Nowak, and Yixuan Li (2022). "Training ood detectors in their natural habitats". In: *International Conference on Machine Learning*. PMLR, pp. 10848–10865.

Kawahara, Yoshinobu, Takehisa Yairi, and Kazuo Machida (2007). "Change-point detection in time-series data based on subspace identification". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, pp. 559–564.

Khan, Shehroz S and Michael G Madden (2010). "A survey of recent trends in one class classification". In: *Artificial Intelligence and Cognitive Science: 20th Irish Conference, AICS 2009, Dublin, Ireland, August 19-21, 2009, Revised Selected Papers 20*. Springer, pp. 188–197.

Kifer, Daniel, Shai Ben-David, and Johannes Gehrke (2004). "Detecting change in data streams". In: *VLDB*. Vol. 4. Toronto, Canada, pp. 180–191.

Kim, Youngin and Cheong Hee Park (2017). "An efficient concept drift detection method for streaming data under limited labeling". In: *IEICE Transactions on Information and Systems* 100.10, pp. 2537–2546.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kingma, Durk P and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in neural information processing systems* 31.

Kolter, J Zico and Marcus A Maloof (2007). "Dynamic weighted majority: An ensemble method for drifting concepts". In: *Journal of Machine Learning Research* 8.Dec, pp. 2755–2790.

Kotsiantis, Sotiris B (2013). "Decision trees: a recent overview". In: *Artificial Intelligence Review* 39, pp. 261–283.

Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In:

Kubat, Miroslav and Kubat (2017). *An introduction to machine learning*. Vol. 2. Springer.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2016). "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *arXiv preprint arXiv:1612.01474*.

Langley, Pat (1996). *Elements of machine learning*. Morgan Kaufmann.

Lazarescu, Mihai M, Svetha Venkatesh, and Hung H Bui (2004). "Using multiple windows to track concept drift". In: *Intelligent data analysis* 8.1, pp. 29–59.

LeCun, Y. (1998). *The mnist database of handwritten digits*.

Lee, Kimin, Honglak Lee, Kibok Lee, and Jinwoo Shin (2017). "Training confidence-calibrated classifiers for detecting out-of-distribution samples". In: *arXiv preprint arXiv:1711.09325*.

Lee, Kimin, Kibok Lee, Honglak Lee, and Jinwoo Shin (2018). "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in neural information processing systems* 31.

Liang, Shiyu, Yixuan Li, and Rayadurgam Srikant (2017). "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *arXiv preprint arXiv:1706.02690*.

Lin, Yih-Kai, Chu-Fu Wang, Ching-Yu Chang, and Hao-Lun Sun (2021). "An efficient framework for counting pedestrians crossing a line using low-cost devices: the benefits of distilling the knowledge in a neural network". In: *Multim. Tools Appl.* 80.3, pp. 4037–4051.

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). "Deep learning face attributes in the wild". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738.

Losing, Viktor, Barbara Hammer, and Heiko Wersing (2016). "KNN classifier with self adjusting memory for heterogeneous concept drift". In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, pp. 291–300.

Lu, Jie et al. (2018). "Learning under concept drift: A review". In: *TKDE* 31.12, pp. 2346–2363.

Lu, Jie et al. (2020). "Learning under Concept Drift: A Review". In: *CoRR* abs/2004.05785.

Lu, Ning, Guangquan Zhang, and Jie Lu (2014). "Concept drift detection via competence models". In: *Artificial Intelligence* 209, pp. 11–28.

Lu, Ning, Jie Lu, Guangquan Zhang, and Ramon Lopez De Mantaras (2016). "A concept drift-tolerant case-base editing technique". In: *Artificial Intelligence* 230, pp. 108–133.

Malinin, Andrey and Mark J. F. Gales (2018). "Predictive Uncertainty Estimation via Prior Networks". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al., pp. 7047–7058.

Malinin, Andrey, Bruno Mlodozeniec, and Mark Gales (2019). "Ensemble distribution distillation". In: *arXiv preprint arXiv:1905.00076*.

Masana, Marc et al. (2018). "Metric learning for novelty and anomaly detection". In: *arXiv preprint arXiv:1808.05492*.

Mayo, Michael and Albert Bifet (2016). "Deferral Classification of Evolving Temporal Dependent Data Streams". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*. Ed. by Sascha Ossowski. ACM, pp. 952–954. URL: https://doi.org/10.1145/2851613.2851890.

Mehrtens, Hendrik Alexander, Camila González, and Anirban Mukhopadhyay (2022). "Improving robustness and calibration in ensembles with diversity regularization". In: *arXiv preprint arXiv:2201.10908*.

Minku, Fernanda L and Xin Yao (2009). "Using diversity to handle concept drift in online learning". In: *2009 International Joint Conference on Neural Networks*. IEEE, pp. 2125–2132.

Minku, Leandro L and Xin Yao (2012). "DDD: A new ensemble approach for dealing with concept drift". In: *IEEE Transactions on Knowledge and Data Engineering* 24.4, pp. 619–633.

Minku, Leandro L, Allan P White, and Xin Yao (2010). "The impact of diversity on online ensemble learning in the presence of concept drift". In: *IEEE TKDE* 22.5, pp. 730–742.

Morningstar, Warren et al. (2021). "Density of states estimation for out of distribution detection". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 3232–3240.

Museba, Tinofirei, Fulufhelo Nelwamondo, Khmaies Ouahada, and Ayokunle Akinola (2021). "Recurrent adaptive classifier ensemble for handling recurring concept drifts". In: *Applied Computational Intelligence and Soft Computing* 2021, pp. 1–13.

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *International Conference on Machine Learning*.

Nalisnick, Eric et al. (2018). "Do deep generative models know what they don't know?" In: *arXiv preprint arXiv:1810.09136*.

Netzer, Yuval et al. (2011). "Reading digits in natural images with unsupervised feature learning". In:

Nguyen, Anh, Jason Yosinski, and Jeff Clune (2015). "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436.

Nguyen, Khanh Xuan and Brendan O'Connor (2015). "Posterior calibration and exploratory analysis for natural language processing models". In: *EMNLP*.

Nishida, Kyosuke and Koichiro Yamauchi (2007). "Detecting concept drift using statistical testing". In: *International conference on discovery science*. Springer, pp. 264–269.

Oord, Aaron Van den et al. (2016). "Conditional image generation with pixelcnn decoders". In: *Advances in neural information processing systems* 29.

Orabona, Francesco and Koby Crammer (2010). "New adaptive algorithms for online classification". In: *Advances in neural information processing systems* 23.

Oza, Nikunj C and Stuart J Russell (2001). "Online bagging and boosting". In: *International Workshop on Artificial Intelligence and Statistics*. PMLR, pp. 229–236.

Page, Ewan S (1954). "Continuous inspection schemes". In: *Biometrika* 41.1/2, pp. 100–115.

Perera, Pramuditha and Vishal M Patel (2019). "Learning deep features for one-class classification". In: *IEEE Transactions on Image Processing* 28.11, pp. 5450–5463.

Pesaranghader, Ali and Herna L Viktor (2016). "Fast hoeffding drift detection method for evolving data streams". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer, pp. 96–111.

Pesaranghader, Ali, Herna L Viktor, and Eric Paquet (2018). "McDiarmid drift detection methods for evolving data streams". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–9.

Poor, H. Vincent (1996). *Detection of abrupt changes: Theory and application : By Michèle and Igor V. Nikiforov. PTR Prentice-Hall, Englewood Cliffs, NJ (1993). ISBN 0-13-126780-9*. Vol. 32. 8, pp. 1235–1236.

Qahtan, Abdulhakim Ali, Basma Alharbi, Suojin Wang, and Xiangliang Zhang (2015). "A PCA-based change detection framework for multidimensional data streams: change detection in multidimensional data streams". In: *Proceedings of the 21th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining. Ed. by Longbing Cao et al. ACM, pp. 935–944.

Quevedo, Joseba et al. (2014). "Combining learning in model space fault diagnosis with data validation/reconstruction: Application to the Barcelona water network". In: *Engineering Applications of Artificial Intelligence* 30, pp. 18–29.

Ramírez-Gallego, Sergio et al. (2017). "A survey on data preprocessing for data stream mining: Current status and future directions". In: *Neurocomputing* 239, pp. 39–57.

Rasmussen, Carl Edward and Joaquin Quinonero-Candela (2005). "Healing the relevance vector machine through augmentation". In: *Proceedings of the 22nd international conference on Machine learning*, pp. 689–696.

Reis, Denis Moreira dos, André Maletzke, Diego F Silva, and Gustavo EAPA Batista (2018). "Classifying and counting with recurrent contexts". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1983–1992.

Ren, Jie et al. (2019). "Likelihood Ratios for Out-of-Distribution Detection". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al., pp. 14680–14691.

Roberts, SW (1959). "Control chart tests based on geometric moving averages". In: *Technometrics* 1.3, pp. 239–250.

Ross, Gordon J, Niall M Adams, Dimitris K Tasoulis, and David J Hand (2012). "Exponentially weighted moving average charts for detecting concept drift". In: *Pattern recognition letters* 33.2, pp. 191–198.

Ruff, Lukas et al. (2018). "Deep one-class classification". In: *International conference on machine learning*. PMLR, pp. 4393–4402.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Saito, Kuniaki, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada (2018). "Maximum classifier discrepancy for unsupervised domain adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3723–3732.

Sakthithasan, Sripirakas, Russel Pears, Albert Bifet, and Bernhard Pfahringer (2015). "Use of ensembles of Fourier spectra in capturing recurrent concepts in data streams". In: *IJCNN*. IEEE, pp. 1–8.

Salganicoff, Marcos (1997). "Tolerating concept and sampling shift in lazy learning using prediction error context switching". In: *Lazy learning*. Springer, pp. 133–155.

Sarafijanovic-Djukic, Natasa and Jesse Davis (2019). "Fast distance-based anomaly detection in images using an inception-like autoencoder". In: *International Conference on Discovery Science*. Springer, pp. 493–508.

Schlimmer, Jeffrey C and Richard H Granger (1986). "Incremental learning from noisy data". In: *Machine learning* 1.3, pp. 317–354.

Sen, Pratap Chandra, Mahimarnab Hajra, and Mitadru Ghosh (2020). "Supervised classification algorithms in machine learning: A survey and review". In: *Emerging technology in modelling and graphics*. Springer, pp. 99–111.

Shafaei, Alireza, Mark Schmidt, and James J Little (2018). "A less biased evaluation of out-of-distribution sample detectors". In: *arXiv preprint arXiv:1809.04729*.

Shafaei, Alireza, Mark Schmidt, and James Little (2019). "A less biased evaluation of ood sample detectors". In: *Proceedings of the British Machine Vision Conference (BMVC)*.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3, pp. 379–423.

Shewhart, Walter Andrew (1931). *Economic control of quality of manufactured product*. Macmillan And Co Ltd, London.

Sinha, Samarth et al. (2021). "Dibs: Diversity inducing information bottleneck in model ensembles". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 11, pp. 9666–9674.

Sobolewski, Piotr and Michal Wozniak (2013). "Concept Drift Detection and Model Selection with Simulated Recurrence and Ensembles of Statistical Detectors." In: *J. UCS* 19.4, pp. 462–483.

Srivastava, Santosh et al. (2016). "Variance change point detection". In: *Proceedings of Indian Workshop on Machine Learning*.

Steiner, Stefan H (1999). "EWMA control charts with time-varying control limits and fast initial response". In: *Journal of Quality Technology* 31.1, pp. 75–86.

Sun, Linjin et al. (2021). "A new predictive method supporting streaming data with hybrid recurring concept drifts in process industry". In: *Comput. Ind. Eng.* 161, p. 107625.

Tanaka, Gouhei et al. (2019). "Recent advances in physical reservoir computing: A review". In: *Neural Networks* 115, pp. 100–123.

Tao, Qing, Dejun Chu, and Jue Wang (2008). "Recursive support vector machines for dimensionality reduction". In: *IEEE Transactions on Neural Networks* 19.1, pp. 189–193.

Tsai, Du-Yih, Yongbum Lee, and Eri Matsuyama (2008). "Information entropy measure for evaluation of image quality". In: *Journal of digital imaging* 21.3, pp. 338–347.

Tuluptceva, Nina et al. (2020). "Anomaly detection with deep perceptual autoencoders". In: *arXiv preprint arXiv:2006.13265*.

Van Geert, Eline and Johan Wagemans (2017). "Objective and subjective complexity-related measures and preferences for neatly organized compositions". In: *40th European Conference on Visual Perception (ECVP), Berlin, Germany*.

Vapnik, Vladimir (1963). "Pattern recognition using generalized portrait method". In: *Automation and Remote Control* 24, pp. 774–780.

Vaze, Sagar, Kai Han, Andrea Vedaldi, and Andrew Zisserman (2021). "Open-set recognition: A good closed-set classifier is all you need". In: *arXiv preprint arXiv:2110.06207*.

Vernekar, Sachin et al. (2019). "Out-of-distribution detection in classifiers via generation". In: *arXiv preprint arXiv:1910.04241*.

Vyas, Apoorv et al. (2018). "Out-of-distribution detection using an ensemble of self supervised leave-out classifiers". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 550–564.

Wang, Haoliang, Chen Zhao, Xujiang Zhao, and Feng Chen (2022). "Layer Adaptive Deep Neural Networks for Out-of-Distribution Detection". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pp. 526–538.

Wang, Heng and Zubin Abraham (2015). "Concept drift detection for streaming data". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–9.

Wang, Senzhang, Jiannong Cao, and S Yu Philip (2020). "Deep learning for spatio-temporal data mining: A survey". In: *IEEE transactions on knowledge and data engineering* 34.8, pp. 3681–3700.

Wang, Shuo et al. (2013). "Concept drift detection for online class imbalance learning". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–10.

Wang, Shuo, Leandro L Minku, and Xin Yao (2017). "A systematic study of online class imbalance learning with concept drift". In: *IEEE Transactions on Neural Networks and Learning Systems*.

Wang, Shuo, Leandro L. Minku, and Xin Yao (2018). "A Systematic Study of Online Class Imbalance Learning With Concept Drift". In: *IEEE Trans. Neural Networks Learn. Syst.* 29.10, pp. 4802–4821.

Wares, Scott, John Isaacs, and Eyad Elyan (2019). "Data Stream Mining: Methods and Challenges for Handling Concept Drift". In: *SN Applied Sciences* 1.11, pp. 1–19.

Webb, Geoffrey I et al. (2016). "Characterizing concept drift". In: *Data Mining and Knowledge Discovery* 30.4, pp. 964–994.

Wei, Hongxin et al. (2022). "Mitigating neural network overconfidence with logit normalization". In: *International Conference on Machine Learning*.

Wilcoxon, Frank (1992). "Individual comparisons by ranking methods". In: *Breakthroughs in statistics*. Springer, pp. 196–202.

Wu, Chen, Bo Du, and Liangpei Zhang (2013). "A subspace-based change detection method for hyperspectral images". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 6.2, pp. 815–830.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747*.

Xiao, Zhisheng, Qing Yan, and Yali Amit (2020). "Likelihood regret: An out-of-distribution detection score for variational auto-encoder". In: *Advances in neural information processing systems* 33, pp. 20685–20696.

Xu, Dan et al. (2015). "Learning deep representations of appearance and motion for anomalous event detection". In: *arXiv preprint arXiv:1510.01553*.

Xu, Shuliang and Junhong Wang (2017). "Dynamic extreme learning machine for data stream classification". In: *Neurocomputing* 238, pp. 433–449.

Yamanishi, Kenji and Jun-ichi Takeuchi (2002). "A unifying framework for detecting outliers and change points from non-stationary time series data". In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 676–681.

Yang, Jingkang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu (2021). "Generalized out-of-Distribution detection: a survey". In: *arXiv preprint* 2110.11334.

Yeh, Arthur B, Richard N Mcgrath, Mark A Sembower, and Qi Shen (2008). "EWMA control charts for monitoring high-yield processes based on non-transformed observations". In: *International Journal of Production Research* 46.20, pp. 5679–5699.

Yu, Honghai and Stefan Winkler (2013). "Image complexity and spatial information". In: *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, pp. 12–17.

Yu, Qing and Kiyoharu Aizawa (2019). "Unsupervised out-of-distribution detection by maximum classifier discrepancy". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9518–9526.

Yu, Shujian, Xiaoyang Wang, and José C Príncipe (2018). "Request-and-reverify: hierarchical hypothesis testing for concept drift detection with expensive labels". In:

Proceedings of the 27th International Joint Conference on Artificial Intelligence. AAAI Press, pp. 3033–3039.

Yu, Shujian et al. (2019). "Concept drift detection and adaptation with hierarchical hypothesis testing". In: *Journal of the Franklin Institute* 356.5, pp. 3187–3215.

Yuan, Yuan, Dong Wang, and Qi Wang (2016). "Anomaly detection in traffic scenes via spatial-aware motion reconstruction". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5, pp. 1198–1209.

Zamprogno, Bartolomeu et al. (2020). "Principal component analysis with autocorrelated data". In: *Journal of Statistical Computation and Simulation* 90.12, pp. 2117–2135.

Zhang, Yuhong et al. (2017). "Three-layer concept drifting detection in text data streams". In: *Neurocomputing* 260, pp. 393–403.

Zhao, Yiru et al. (2017). "Spatio-temporal autoencoder for video anomaly detection". In: *Proceedings of the 25th ACM International Conference on Multimedia*, pp. 1933–1941.

Zhou, Chong and Randy C Paffenroth (2017). "Anomaly detection with robust deep autoencoders". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674.

Zliobaite, Indre (2013). "How good is the electricity benchmark for evaluating concept drift adaptation". In: *arXiv preprint arXiv:1301.3524*.

Zliobaite, Indre et al. (2015). "Evaluation methods and decision theory for classification of streaming data with temporal dependence". In: *Mach. Learn.* 98.3, pp. 455–482.

Zong, Bo et al. (2018). "Deep autoencoding gaussian mixture model for unsupervised anomaly detection". In: *International Conference on Learning Representations*.