**Noname manuscript No.**
(will be inserted by the editor)

# AnyNovel: Detection of Novel Concepts in Evolving Data Streams

## An application for activity recognition

Received: date / Accepted: date

**Abstract** A data stream is a flow of unbounded data that arrives continuously at high speed. In a dynamic streaming environment, the data changes over the time while stream evolves. The evolving nature of data causes essentially the appearance of new concepts. This novel concept could be abnormal such as fraud, network intrusion, or a sudden fall. It could also be a new normal concept that the system has not seen/trained on before. In this paper we propose, develop, and evaluate a technique for concept evolution in evolving data streams. The novel approach continuously monitors the movement of the streaming data to detect any emerging changes. The technique is capable of detecting the emergence of any novel concepts whether they are normal or abnormal. It also applies a continuous and active learning for assimilating the detected concepts in real time. We evaluate our approach on activity recognition domain as an application of evolving data streams. The study of the novel technique on benchmarked datasets showed its efficiency in detecting new concepts and continuous adaptation with low computational cost.

**Keywords** Stream mining · Concept evolution · Activity recognition · Continuous learning · Active learning · Novelty detection

### 1 Introduction

Data stream mining deals with high speed, unbounded, and fast changing flow of data. In order to discover knowledge from data streams, methods and approaches that are tailored to the nature of streaming data have been developed. The change in data stream distribution is known as *concept drift* that is first introduced in [Schlimmer and Granger Jr, 1986]. The essential issue of "concept evolution" in data streams refers to the appearance of novel classes while streams evolve. Traditional machine learning techniques are unable to detect

these novel classes and consequently misclassify all instances representing the novel concepts. In order to cope with concept evolution in data streams, the mining technique has to be able to detect a novel concept as soon as it arrives without being trained with labelled data. That also followed with assimilating the novel concept into the underlying concept for further detection of recurring novel class instances.

Learning from data streams with concept drift requires not only the detection of novel concepts but also forgetting outdated and abandoned concepts. Close monitoring of the evolution of data over the time facilitates the detection of concepts that became irrelevant by the time. Existing concepts could disappear either gradually or suddenly. Moreover, it is hard to distinguish between totally abandoned concepts or less frequently detected/occurred.

Few studies addressed the concept evolution problem with stream mining as in [Spinosa et al., 2007] and [Masud et al., 2011b]. However, most of the existing techniques in concept evolution target the detection of "abnormal" instances/class in data stream. Abnormality detection is applicable in various domains such as network intrusion detection, fall detection and credit card fraud detection. However, in our approach, we are not only concerned about the abnormal concepts but also provide a solution to the appearance of novel "normal" concepts that might emerge among existing normal concepts. This is very practical in data stream real life applications that suffer from the scarcity of labelled data resulting to poorly built learning models.

In this paper, we propose, develop, and evaluate our technique for detecting normal and abnormal novel concepts in evolving data streams termed **AnyNovel**. We mean by concept evolution the detection of both normal and abnormal novel concepts and also forgetting outdated ones. Our technique applies a continuous and active learning to handle the appearance/disappearance of concepts in data streams. AnyNovel has the merit of considering both the dependency and distribution of data in the stream of data. The idea of AnyNovel is to monitor the movement of data along the evolving stream over time. The sequence of data that moves away from all existing concepts and at the same time forms a stable and separate concept is expected to be novel. Moreover, AnyNovel dynamically adapts the learning model by incorporating the detected novel concepts or removing the outdated ones. Therefore, the adapted learning model is capable of recognising any recurring occurrences of the novel concepts without any manual intervention.

This approach is general for detecting novel concepts in data streams. We apply the proposed technique on activity recognition as a case study to demonstrate its efficiency in realistic conditions. Activity recognition (AR) aims to provide accurate and opportune information on people's activities and behaviour in real time. In AR domain, the novel approach aims to handle evolving activities that appear/disappear in the stream with no previous reference in the underlying model. The detection of novel concepts in AR includes both the detection of abnormal activities (outliers) such as a "Sudden fall" and also the detection of new normal activities such as "Driving" (when the "Driving"

activity did not exist in the training data). The term concept is often used interchangeably in this paper with the term activity.

AnyNovel provides the following major contributions:

- *Detecting any novel concepts:* The new technique is capable of detecting both new abnormal and normal concepts regardless of their spatial location from the underlying concepts. The detection of novel normal concept is intuitively more challenging as it appears amongst other normal concepts. In the experiments of this paper, we focus on the detection of the novel normal as the most challenging task for novelty detection. Also, a successful detection of novel normal concepts proves the capability of the technique to detect the abnormal ones too. However, the opposite is not correct in principle.
- *Distinguishing novel concept, drift, and outliers:* AnyNovel creates an elastic slack around each cluster to accommodate concept drift. The declaration of a novel concept requires data to reside outside all slacks and consistently move away from all existing concepts. This way, our technique can distinguish between a completely new concept and an extension or drift of an existing concept.
- *Forgetting abandoned concepts:* Equally important, AnyNovel can detect the disappearance of concepts in the recent stream. Therefore, AnyNovel also is capable of detecting concepts that are no longer relevant.
- *Dynamic adaptation of the learning model:* In AnyNovel, the underlying learning model is continuously enriched with novel detected concepts to accommodate recent changes in the stream and detect recurring data that belongs to novel classes.
- *Incorporating active learning with low cost:* The annotation process is very costly and impractical in a streaming environment. Thus, AnyNovel triggers active learning only for the small amount of data that is the most uncertain.

The rest of this paper is organised as follows. We first discuss the related work. Then, we begin with explaining assumptions and formalisation of the problem. Then, we discuss the implementation details of AnyNovel framework, components and methodology. The contribution of AnyNovel is discussed and highlighted. We evaluate and analyse AnyNovel on benchmarked datasets later in this paper. Finally, the paper is concluded with a summary.

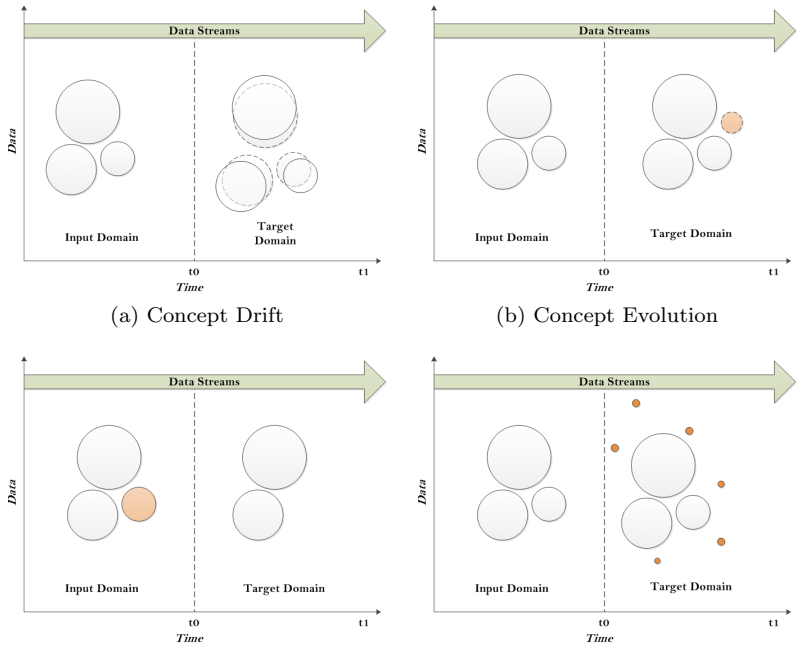
## 2 Background and Related Work

Our work is related to both data stream mining and the application of activity recognition. Thus, we briefly describe both areas of research.

Figure 1 illustrates different types of changes in data streams. Figure 1(a) represents the type of change known as *concept drift* which is first identified in [Schlimmer and Granger Jr, 1986]. It refers to the gradual change in distribution between input and target domains. Many techniques have been

developed to address concept drift in data stream such as an online information network (OLIN) [Last, 2002] and a fuzzy classifier in [Lughofer and Angelov, 2011]. The other type of change is *concept evolution*, which refers to the appearance of novel concept in the stream as in Figure 1(b). Detecting a novel concept is a challenging task in data streams, especially when dealing with data streams with concept drift. An efficient approach has to be able to distinguish between the drifting of an existing concept and the appearance of an entirely new concept. Novelty detection is defined in [Pimentel et al., 2014] as "the task of recognising that test data differ in some respect from the data that are available during training". Recent surveys have addressed concept evolution specifically for data streams as in [Gurjar and Chhabria, 2015] and [Faria et al., 2015a]. Different approaches have been developed to distinguish between existing and novel concepts. In terms of the underlying concept, some approaches identified the underlying existing concept as a single model that represents only a single class, while others considered a multi-class underlying concept. Spinosa, Carvalho, and Gama [Spinosa et al., 2007] represented the underlying model by a single concept with all incoming data either part of the underlying model or novel concept. This approach assumes that there is only one "normal" class and any other classes are novel. The cluster based system, named OLINDDA, is based on three models: normal profile model, concepts that extend the normal profile, and novel concepts. Learning phases in OLINDDA are offline and online. The normal model is built in the offline phase, while the extension in the online phase detects minor changes in the normal model. A novel concept is detected when incoming data is located away from the normal model and also satisfies a specific validation criterion. A single model approach limits the capabilities of a concept evolution algorithm to differentiate only between existing and novel concepts without considering the presence of multiple existing concepts. Thus, other approaches have developed to address the multi-class structure of an underlying concept.

The techniques developed in [Masud et al., 2011b, Faria et al., 2013, Hayat and Hashemi., 2010] are examples of stream learning approaches for novelty detection with multi-class underlying concept. ECSMiner [Masud et al., 2011b] applies an ensemble of classifiers to an incoming stream of equally sized data chunks for prediction. The global decision boundary is defined as the union of local decision boundaries for existing classes in the underlying model. The classifier model in the ensemble classifier is dynamically updated to detect instances that are outside the global boundary. If no classifier is able to predict the incoming data, then data is stored in short memory for further processing. The novel concepts are detected when data in a buffer maintains cohesion with other buffer data and separation from existing underlying concepts. This approach addresses the novelty detection in multi-class underlying concepts, yet it requires all data chunks to be labelled to define the new concept. Faria et al. [Faria et al., 2015b, Faria et al., 2013] proposed the MINAS system for concept evolution which applies unsupervised learning approaches. MINAS classifies new incoming instances as known or unknown. Unknown instances are the ones located outside the decision boundaries. The declared unknown




(a) Concept Drift (b) Concept Evolution

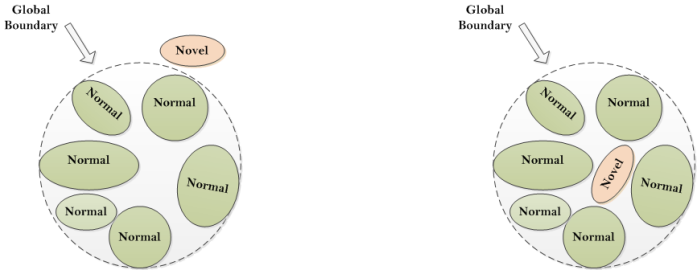
(c) Concept Forgetting (d) Outliers

Fig. 1: Changes in Data Streams

instances are stored in short time memory. Then, data in short memory is clustered in order to discover new concepts. Hayat and Hashemi [Hayat and Hashemi., 2010] proposed an approach that is based on the discrete cosine transform to build normal concepts of multi-classes with sub-clusters. The distance measure is also applied to distinguish between existing and novel concepts.

The aforementioned approaches rely mainly on the distance measures that predict novel concept based its location from the decision boundaries. The new concept is declared as novel if it is outside the global decision boundary which is the union of local boundaries of clusters, as explained in Figure 2(a). Although the underlying concept contains multi classes, creating a global decision boundary results in a similarity between multi-class models and single class models. It combines all existing concepts in one concept and defines the global boundary for the combined concepts. Thus, these approaches also did not address the appearance of novel concepts that might exist outside the local boundaries, yet inside the global boundary as in Figure 2 (b). Moreover, most of theses approaches did not consider the labelling cost of data streams when identifying novel concepts.

Concept evolution approaches are also categorised based on the action taken upon novelty detection. Yeung and Chow [Yeung and Chow, 2002] and




(a) Outside (b) Inside

Fig. 2: Explanation of the Novel Concept Positions

Yang et al. [Yang et al., 2002] presented instance based algorithms that consider whether the incoming data is sufficiently close or far from the underlying concepts based on some appropriate metric. Novelty detection in these techniques detect "filtered out" instances without tracking data for the detection of normal concepts. While approaches in [Faria et al., 2013, Hayat and Hashemi., 2010, Masud et al., 2011b, Spinosa et al., 2007] extend the detection of "filtered out" instances by detecting the level of cohesion among these instances to form a novel concept. Studies in [Masud et al., 2011a, Al-Khateeb et al., 2012] further integrated the novel concept with underlying ones in order to detect recurring instances that belong to novel concept. Figure 3 summarises various approaches in concept evolution. According to this summary, our technique detects the evolution of concepts in multi-class model. Once the new concept is detected, we integrate it into the model for detecting recurring concepts.

Techniques in novelty detection consist typically of two phases: training on the normal concepts and detection of novel concepts. For example, in [Costa et al., 2015] and [Costa et al., 2014], the authors apply two stage density based method for fault detection. In the first stage, the method extracts a very small amount of information from the training data samples. The data representatives, including mean and momentum, are stored in the memory and continuously updated. This training approach is similar to AnyNovel in extracting only the characteristics (data representatives) and dismissing all raw data. However, our technique aims not only to detect fault (outliers) but also new normal concepts in the stream. The neural network developed in [Marsland et al., 2005] is using a novelty filter that decides on the novelty with respect to the training data that were seen by the model before. One problem that encounters any novelty detection technique is the absence of labelled data that represents the novel concept. Authors in [Haque et al., 2015] developed a semi-supervised approach that applies a dynamic sliding window using a limited number of labelled data to detect both concept drift and concept evolution in evolving data streams. AnyNovel presents a more efficient solution for this problem by incorporating active learning in a semi-supervised approach to detect any novel concepts.

The idea of concept evolution is also attached to concept forgetting. The system performance relies on its capabilities to learn changes and new concepts appear in the stream as well as forgetting outdated concepts that became a burden on the system [Kifer et al., 2004]. Whereas incremental learning manages the continuous learning of new concepts, decremental learning focuses on ***forgetting abandoned concepts*** [Cauwenberghs and Poggio, 2001]. Figure 1(c) explains the concept forgetting in data streams.

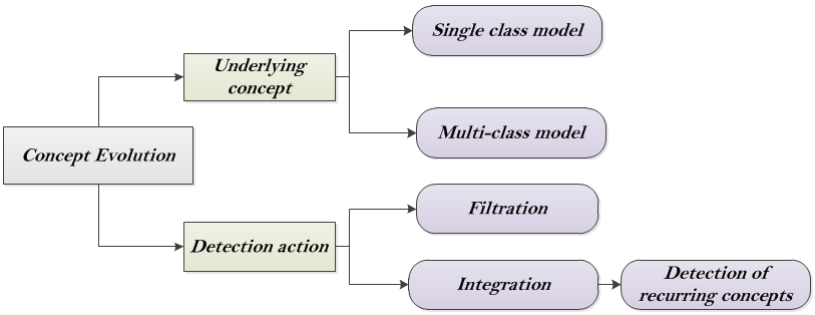



Fig. 3: Summary of Concept Evolution Approaches

The other category of change that is relevant to concept evolution is *outliers/anomalies* as in Figure 1(d). Many of the outliers are considered as noise, while others are of paramount importance such as credit card frauds. Outliers are defined as data instances which deviate from underlying concepts. However, this definition also applies for novel concepts. The main characteristic that differentiates between outliers and concept evolution is the cohesiveness among "filtered out" instances. The novel concept has to satisfy a validation criterion that concerns mainly the separation of short memory instances from underlying concepts as well as cohesion among these instances to form a novel concept. In streaming settings, outlier detection under realistic assumptions is an unsupervised learning approach because they are mostly rare to occur. Therefore it is not possible to train the model on them beforehand. Different metrics have been developed in literature to measure the deviation of incoming data from underlying concept that define anomalies. The deviation measure such as distance and distribution have been applied pervasively for outlier detection. There are many techniques that studied outlier detections in data streams such as [Angiulli and Fassetti, 2007, Assent et al., 2012, Niennattrakul et al., 2010, Pokrajac et al., 2007]. Few studies have combined the detection of outliers with concept evolution such as in [Masud et al., 2011a, Masud et al., 2013, Al-Khateeb et al., 2012]. A detailed survey of outlier detection approaches is presented in [Aggarwal, 2013]. Our technique is different from the aforementioned outlier techniques in two ways: i. integrating concept evolution and outliers; and ii. considering concept evolution for new normal concept that appears inside the global boundary.

The other research area that intersects with our research is AR. Methods commonly used for activity classification were reviewed in [Preece et al., 2009,Peterek et al., 2014]. In [Gomes et al., 2012,Abdallah et al., 2015,Andreu and Angelov, 2013], authors addressed the issue of stream mining with AR. Although this technique handled data stream with concept drift, it lacks the ability to detect the appearance of novel and recurring novel concept. Andreu et al [Andreu and Angelov, 2013] developed an evolving fuzzy rule classifier that adapts over the time. The classifier can cope with concept drift in data stream. This classifier is based on an early work in [Andreu et al., 2011] for real time classification. Concept evolution in AR is related also to detecting abnormal activity. The term of "sudden activity" or "fall detection" is pervasively discussed in activity recognition especially for elderly people aids [Luštrek and Kaluža, 2009,Mubashir et al., 2013]. However, these approaches are applied in non-streaming environment to detect only outlier/abnormal instances without integrating the emerging of new activities.

Few of the existing techniques combine the three aspects of activity recognition, novelty detection and stream mining. Table 1 summarises some of the recent techniques applied across the three disciplines.

Table 1: comparison among techniques for novelty detection in AR

| Technique | [Krishnan and Cook, 2014] | [Rashidi and Cook, 2010] | [Abdallah et al., 2015] | [Spinosa et al., 2007] | [Masud et al., 2011a] | [Nguyen et al., 2015] | [Lockhart and Weiss., 2014] |
|---|---|---|---|---|---|---|---|
| Activity recognition | ✓ | ✓ | ✓ | X | X | ✓ | ✓ |
| Stream mining | ✓ | ✓ | ✓ | ✓ | ✓ | X | ✓ |
| Novelty detection | partially | X | X | ✓ | ✓ | ✓ | X |
| Outliers | X | X | X | X | ✓ | X | X |
| Concept forgetting | X | X | X | X | X | X | X |

As shown in the Table 1, techniques for novelty detection in data streams are not applied for activity recognition. The approach in [Krishnan and Cook, 2014] is an exception that considers the discovery of 'other' activity. The recognition of other activity is different from recognising novel activities from an evolving stream. The technique has to build a static model offline that represents other transitional and unknown activities. It explicitly trains the model offline for transitional activities and other unknown activities. Then, it gathers them all in other activity class and incorporates it with the model. The technique developed by Nguyen et al. [Nguyen et al., 2015] requires limited labelled data representing the new concepts to train the model beforehand. The technique is also implemented in non-streaming settings. Therefore, the above

approaches focus either on novelty detection but not for activity recognition or on activity recognition without addressing the novelty detection. AnyNovel fills in this gap by addressing both directions as it combines novelty detection in streaming settings with activity recognition.

In the next section, we start formalising the problem for clear representation of AnyNovel framework and methodology.

## 3 Assumptions and Formalisations

Data streams arrive at high speed and require real or near real time processing. We assume that the $n^{th}$ chunk in the stream arrives at the $n^{th}$ time stamp. While a concept evolves, decisions may need to be delayed until enough data arrives. The collected data is kept in a *Buffer* till either reaching a decision or reaching the limit of the *Buffer* capacity.

Figure 4 illustrates different time snapshots along a data stream. A stream of data is segmented into $n$ chunks of equal size. AnyNovel starts at time $t_0$, when the *Buffer* is empty. As stream evolves, the *Buffer* is accumulated with instances of the new data chunks. The algorithm tries to make a decision about data in the *Buffer* whenever a new chunk arrives. A declared decision is one of three possibilities: existing, novel, or unknown. The *Buffer* data might have been declared as part of an existing cluster (not novel), while a novel concept is declared when the *Buffer* data satisfies all conditions of a novel concept. An unknown decision is declared when a high level of uncertainty occurs between the two decisions of existing and novel. $t_{dc}$ is the time when reaching any of the three decisions. At this time, we reset the *Buffer* and start processing new chunks of data.

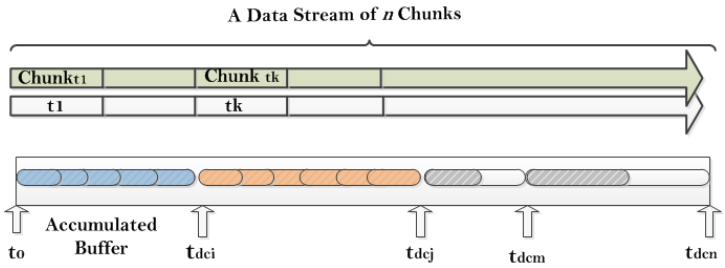



Fig. 4: Time Snapshots Along the Data Stream

AnyNovel handles three types of data repositories across various time stamps. For example, in Figure 4, assume the latest *Buffer* data is declared at time $t_{dc_i}$. After the time of declaration, the three data repositories are:

– *Data Chunk* ($Chunk_{t_k}$): Chunk of a data stream that arrives at time $t_k$. Where $t_{dc_i} < t_k$, which means the chunk arrives after the latest declared *Buffer* data.

– *Buffer*: Temporarily short memory to accumulate continuous undeclared chunks of data. It starts to accumulate from the last declared data at $t_{dc_i}$. The *Buffer* is accumulated until either a decision or capacity limit is reached at time $t_{dc_j}$, where $t_{dc_i} < t_{dc_j}$. For each chunk of data, we check first if there is sufficient information to declare data with no need to keep it in the *Buffer*. Otherwise, all undeclared data between $t_{dc_i}$ and $t_{dc_j}$ is kept in the *Buffer*. Thus, Buffer data is either the entire data or a subset of the data that is received between $t_{dc_i}$ and $t_{dc_j}$.
– *Just Predicted* ($JP$): Very short memory that keeps a reference of the last declared data at time $t_{dc_i}$. This short memory is released once checked at the following time stamp, when the next data chunk arrives.

Based on the defined concepts and assumptions, we then discuss the conceptual framework of AnyNovel and its components.

## 4 AnyNovel Conceptual Framework

Figure 6 illustrates the conceptual framework of AnyNovel with its phases and components. The modelling component in AnyNovel builds a fine-grained Baseline Learning Model (BLM) that not only represents activity but also represents various patterns inside each activity. The fine-grained model allows the separation of the cluster into subsets that have distinguishable characteristics. We refer to these subsets as sub-clusters of the cluster. Sub-clusters of a particular cluster in activity recognition application are analogous to different patterns within this particular activity. The methodology for building the BLM starts with applying a supervised learning technique to the annotated sensory data collected for training. Thus, the initial learning model that is composed of a number of clusters is created. Each cluster corresponds to one of the labelled activities that exist in the training data. After creating $m$ clusters, we further split each cluster into smaller sub-clusters. We apply EM clustering technique to each cluster to form sub-clusters representing different patterns inside the cluster. The process of building the BLM with the sub-clusters approach is illustrated in Figure 5.

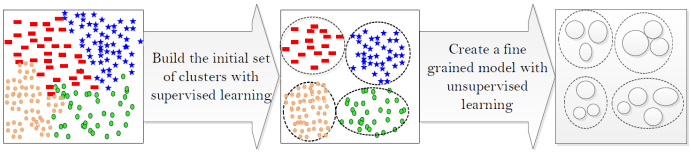



Fig. 5: Illustration of the Baseline Learning Model (BLM)

The BLM is represented by a set of informative characteristics that best describe activities in the training data. We aim in this component at building a

robust, flexible, and efficient BLM that is the basis for successful recognition. To achieve this, we extract a summary of the statistics of the each cluster, before we dismiss all raw data instances at the end of the modelling process. The Extracted characteristics span across three levels of description which are: sub-cluster characteristics, cluster characteristics, and holistic characteristics. Table 2 summarises the most commonly used characteristics for describing the baseline learning model. One of the important characteristics is $Dmax$ which defines the decision boundary of each cluster. $Dmax$ is a cluster level characteristic that defined in Table 2, as the maximum distance between any pair of sub-clusters within the cluster.

Table 2: Commonly Used BLM Characteristics

| Symbol | Description | Level |
|---|---|---|
| $Centroid_{sc_i}$ | The mean of $n$ dimensional points inside sc | Sub-cluster |
| $Weight_{sc_i}$ | Total number of data samples belong to sc | |
| $Radius_{sc_i}$ | The maximum distance within a sub-cluster between $Centroid_{sc}$ and any data sample belonging to sc | |
| $Nsub_{C_j}$ | Number of sub-clusters inside the cluster | Cluster |
| $Centroid_{C_j}$ | $C_j$ centroid which is the mean of all sub-clusters centroids belonging to this cluster | |
| $Weight_{C_j}$ | Total number of data instances inside the cluster | |
| $Radius_{C_j}$ | Max distance between $Centroid_{C_j}$ and all data instances inside the cluster | |
| $Dmax_{C_j}$ | Max distance between any pair of sub-clusters | |
| $Nclus$ | Number of clusters/activities in the data domain | Holistic |
| $Centroid_{glob}$ | Global centroid of all instances in the training data | |
| $Capacity$ | Total number of training data instances | |

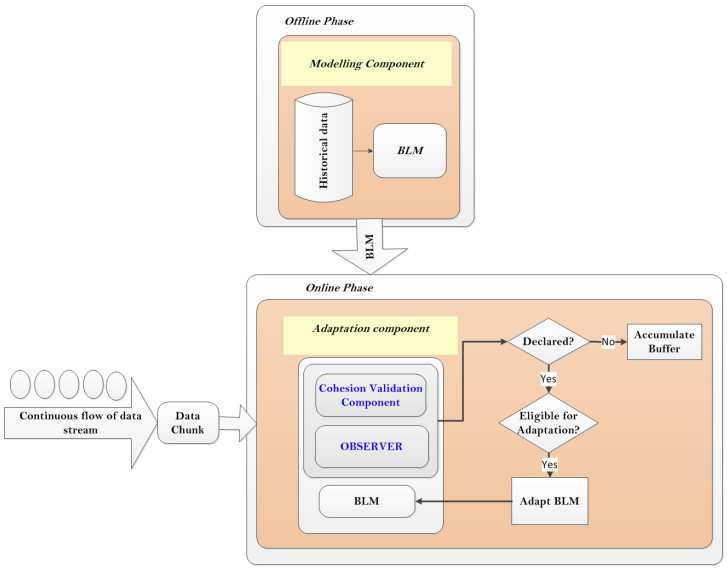



Fig. 6: AnyNovel Conceptual Framework

The adaptation component integrates the BLM and analyses chunks of data streams, in order to reach a decision. There are two key units in the adaptation component: the Cohesion Validation Component (CVC) and the OBSERVER as in Figure 6. The two units interact together to analyse data chunks for reaching a decision. The CVC checks and validates the dependency between the new incoming data chunk and the most recent declared data and/or *Buffer* data. The merit of the CVC is to aggregate dependent data that represents the same concept together, thus it enhances the OBSERVER capability to reach a decision about the aggregated data. The OBSERVER monitors the evolution of data along the stream. Thus, it checks the movement, separation, and stability of the continuously aggregated data that is accumulated in the *Buffer*.

Figure 7 shows the flow between the two key units of the CVC and the OBSERVER. The CVC detects and validates the dependency between *Buffer* and $JP$ data from one side and the new data chunk ($Chunk_{t_n}$) from the other side. It also aims to isolate suspected outliers. Therefore, the CVC finally generates a reformed *Buffer* that is fed to the OBSERVER along with the new data chunk for monitoring the movement of data stream. The OBSERVER is integrated with the latest updated BLM for declaring a decision based on the most recent information. In the following, we discuss both the Cohesion Validation Component and the OBSERVER in detail.

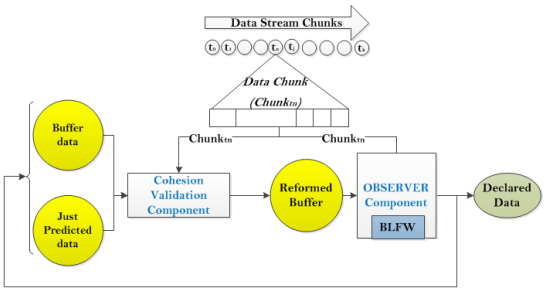



Fig. 7: AnyNovel Adaptation Component Overview

## 5 Cohesion Validation Component (CVC)

The goal of this component is to perform the essential reformation of the *Buffer* for monitoring the movement in the following OBSERVER component. The CVC performs the reformation by processing the three repositories of $JP$, *Buffer*, and the new data chunk $Chunk_{t_n}$. The flow chart of Figure 8 shows the flow of data for reforming the *Buffer* in the CVC. The principle function to check cohesiveness in the CVC is the Cohesion Validation Test (CVT). There are four operations performed on the *Buffer* based on this algorithm, namely seed, replace, keep, and reset. The *Buffer* is a decisive parameter in the algorithm.

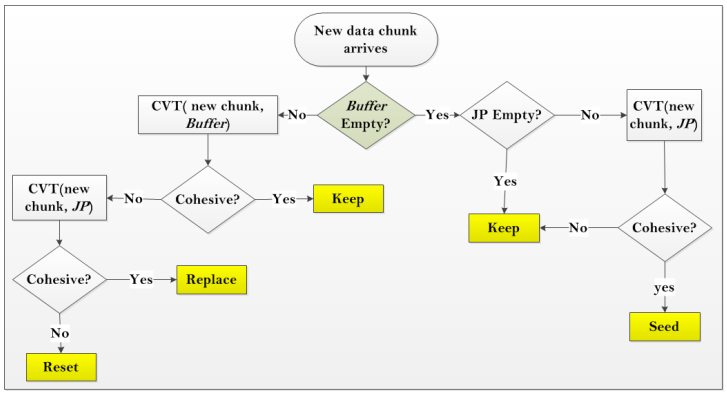



Fig. 8: Flow Chart of Cohesion Validation Component

The key function for testing the continuity and dependency among data is the Cohesion Validation Test. We discuss the definition of the cohesion validation and details of the test procedure.

### 5.1 Cohesion validation test

Any data repository consists of a set of data instances. We first define data repository as follows: $Rep_i$ = { $inst_{i0}$,$inst_{i1}$, $inst_{i2}$,...., $inst_{in}$ } where $n$ is the number of instances belonging to the repository. In CVT, the repository definition applies for the *Buffer*, $JP$ and $Chunk_{t_n}$ -stream data chunk at the $n^{th}$ time stamp. CVT checks a pair of repositories by the following procedure. It merges instances from both repositories and applies an online clustering technique to the merged data to generate exactly two clusters. If the generated clusters contain mixed up instances from both repositories, that means they are strongly related. Thus, the two repositories are cohesive as they are strongly attached. On the other hand, when the two clusters are well separated, each contains almost pure instances from one of the repositories, then CVT declares a separation between the two repositories. By the definition of the cohesion test, two repositories are separated if any of them is an empty set of instances. We apply EM clustering technique over a small chunk of data allowing real time clustering to take place in the online phase. EM clustering aims at maximising the likelihood among each individual repository instances. This is consistent with the purpose of the cohesion validation test. Based on the CVT, there are four operations to reform the *Buffer*. The four operations are seed, replace, reset, and keep. Description of the four operations is described in the following.

Using the aforementioned cohesion validation test, one of the following four operations, abbreviated SPRK, are Seed, rePlace, Reset, and Keep, is performed to reform the *Buffer*. Definition of each operation is defined as follows:

- **Seed Operation:** This operation is the one that declares the dependency between just declared data and the new data chunk.
  ***Preconditions***: the *Buffer* is empty and $JP$ is cohesive with $Chunk_{t_n}$.
  ***Actions***: Initiate the *Buffer* with $JP$ data.
- **Keep Operation:** This is the case when there is a dependency between the new incoming data and the data in the *Buffer*. Therefore, no reformation is required.
  ***Preconditions***: When the *Buffer* is in cohesion with $Chunk_{t_n}$ or when the *Buffer* is empty and $JP$ is not cohesive with $Chunk_{t_n}$.
  ***Actions***: No change.
- **Replace Operation** [suspected outlier case]: In this case, the CVC checks the dependency between $Chunk_{t_n}$ and $JP$ for the detection of possible outliers. When data that has been accumulated in the *Buffer* is not cohesive with the dependent data in both $JP$ and the new incoming data in $Chunk_{t_n}$, then it is suspected to be outlier or noisy data.
  ***Preconditions***: the *Buffer* is not empty and not cohesive with $Chunk_{t_n}$, however $JP$ is cohesive with $Chunk_{t_n}$.
  ***Actions***: the *Buffer* is declared, based on available information and $JP$ data replaces *Buffer* data.

– **Reset Operation:** In this case, new incoming data represents a different and independent concept.
***Preconditions:*** When the *Buffer* is not empty and both *Buffer* and $JP$ are not cohesive with $Chunk_{t_n}$.
***Actions:*** Reset all and release data in the *Buffer* with available information.

The reformed *Buffer* along with the new data chunk are sent as an input for actual monitoring of data movement by the OBSERVER. Details of the OBSERVER procedure are depicted in the next section.

## 6 OBSERVER Component

The OBSERVER is the key component in the concept evolution technique that is responsible for monitoring different types of concepts and declaring decisions. It receives the new data chunk along with the reformed *Buffer* (output of the Cohesion Validation Component) in order to track the data movement and decide accordingly. There are two data repositories that are processed with the OBSERVER component, $Chunk_{t_n}$, and *Buffer*. In terms of time, $Chunk_{t_n}$ arrives at $t_n$ while *Buffer* data is collected along duration from $t_j$ to $t_n$, where $t_j < t_n$. The movement of data is monitored by checking the evolution in data from *Buffer* and $Chunk_{t_n}$ between $t_j$ and $t_n$.

In abstract, the OBSERVER takes one of three decisions which are *existing*, *novel*, or *unknown*. An *existing* decision reflects the recognition of a data repository as part of the underlying BLM existing clusters. This decision is declared when data falls inside the boundaries of any of the BLM clusters or consistently moves closer to existing clusters. On the contrary, a *novel* decision is declared when data resides outside all of the BLM clusters' boundaries and consistently moves away from all clusters. The novel concept must also satisfy a stability cohesion and separation criteria to be considered as an eligible novel concept.

The problem space in AnyNovel consists of the BLM with clusters and sub-clusters of each cluster. Each cluster in the BLM is a hypersphere in a feature space that has its own local decision boundary. The decision boundary of the cluster is identified based on $Dmax_c$ characteristic. Surrounding the decision boundary of each cluster, there is a slack that accommodates for drift of data that represents the cluster. We term this slack as DRAB that stands for **D**ynamic **R**el**A**xed **B**oundary. Thus, DRAB allows an extended space to accommodate concept drift for each cluster. Figure 9 shows the position of both decision boundary and DRAB around a single BLM cluster with sub-clusters. DRAB height is the threshold that defines the space between the decision boundary and DRAB.

To be able to check the movement along time, we need first to define the different possible positions of data repository with respect to the BLM. First, a BLM with $m$ clusters and $k$ sub-clusters across all $m$ clusters is defined as $BLM = \{C_1\{sc_{11}, sc_{12}....sc_{1P_1}\}, ....C_m\{sc_{m1}, sc_{m2}....sc_{mP_m}\}\}$. Where $k = \sum_{i=1}^{m} P_i$.

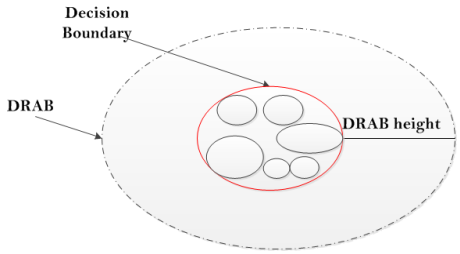



Fig. 9: Illustration of DRAB and Decision Boundary

$Rep_l$ is a general term that refers to any data repository. In the OBSERVER, term $Rep_l$ refers to either $Chunk_{t_n}$, the *Buffer*, or a union of both. We define the spatial position between repositories based on their centroids. The spatial position of data repository in respect to the BLM clusters is one of three: *Inside*, *InSlack* or *OutsideALL*.

- *Inside:* $Rep_l$ is Inside $C_j \in$ BLM, if the Euclidean distance between the centroid of $Rep_l$ and $Centroid_{C_j}$ is less than $Dmax_{C_j}$.
- *InSlack:* $Rep_l$ is InSlack of $C_j \in$ BLM if the Euclidean distance between the centroid of $Rep_l$ and $Centroid_{C_j}$ is above $Dmax_{C_j}$ yet less than $DRAB_{C_j}$.
- *OutsideALL:* $Rep_l$ is OutsideALL when the repository centroid is outside the $DRAB_{C_j}$ $\forall$ $C_j \in$ BLM. This definition applies also for a repository that resides inside the global boundary but outside the boundaries of all clusters in the BLM.

We also define a new term that explains the expected next position of merged instances of both the *Buffer* and $Chunk_{t_n}$. The term *VINPos*, which stands for VIrtual Next Position, indicates the virtual position when the *Buffer* is accumulated with new data chunk as well as the movement of data. The position of any repository is calculated based on the centroid location. *VINPos* is defined as: the prospective centroid position of the *Buffer* when accumulated with $Chunk_{t_n}$. *VINPos* has one of two values: either *IN* when the merged position resides Inside or InSlack of any BLM clusters or *OUT* otherwise.

Based on the definition of the problem space and formalisations, we explain the details of the OBSERVER methodology for detecting the evolution of concepts along the stream.

### 6.1 OBSERVER overview

The OBSERVER monitors the evolution of the reformed *Buffer* and $Chunk_{t_n}$ over time. Algorithm 1 illustrates steps of the OBSERVER procedure. The algorithm starts with checking data in the *Buffer* at time $t_n$ (line 1). The *Buffer* is empty in two scenarios. The first is at the beginning of the stream, when no data has been processed yet. The second scenario is when the algorithm resets all settings and empties the *Buffer*, when data is just declared or released at

$t_{n-1}$. In these two scenarios, the new chunk is processed with the **P**rocess **D**ata with **N**o **H**istory (PDNH) procedure (line 2). Otherwise, in case the *Buffer* is not empty, the algorithm calculates *VINPos* and checks its position in respect to the BLM clusters (line 4). The OUT value of *VINPos* (line 5) indicates its location outside the $C_j$ boundaries $\forall C_j \in$ BLM where $j \in 1...m$. When the OUT value of VINPos is validated, the data is suspected to be novel as its virtual position is spatially outside the boundaries of all existing clusters. The suspected novel data is tested with the **ObserveSuspNovel** procedure for monitoring and detecting novel concepts (line 6). Otherwise, when *VINPos* resides inside the boundaries of any cluster $C_j \in$ BLM, then the accumulated data is declared as an existing concept (line 8).

**Algorithm 1** OBSERVER Algorithm

**Input:** $Chunk_{t_n}$: data chunk at time $t_n$
*Buffer*: short time memory of previous undeclared data
BLM: most recent baseline learning model
**Output:** declared data if any
1: **if** Empty(*Buffer*) **then**
2: **PDNH**($Chunk_{t_n}$, BLM)
3: **else**
4: *VINPos* = CalculateVINPos($Chunk_{t_n}$, *Buffer*)
5: **if** *VINPos* is OUT **then**
6: **ObserveSuspNovel**($Chunk_{t_n}$,*Buffer*, BLM)
7: **else**
8: **DeclareExisting** ($Chunk_{t_n}$, *Buffer*, BLM)
9: **end if**
10: **end if**

The OBSERVER implements one of three procedures of PDNH, ObserveSuspNovel, or DeclareExisting. Each of the three key procedures may declare a decision whenever adequate information is available. The PDNH procedure deals directly with $Chunk_{t_n}$ and checks its position from the BLM clusters when there is no previous history of its movement. The position is either *Inside*, *InSlack*, or *OutsideALL*. A $Chunk_{t_n}$ that falls *Inside* any clusters' decision boundaries is classified and declared as existing. The declared data is moved to the $JP$ short memory for checking the dependency with the following data chunks. In all other cases, PDNH initiates the *Buffer* with data in the $Chunk_{t_n}$ that resides *InSlack* of any of BLM clusters or *OutsideALL*. At this stage, there is not enough information to check the movement of data. Therefore, the data is stored in the *Buffer*, for further processing when more stream chunks arrive.

PDNH is the basic procedure that is implemented at the beginning of the stream or when all settings are reset. When the *Buffer* is not empty, i.e., undeclared data exists in the *Buffer*, the virtual position decides on the movement according to new data whether closer or away from the BLM underlying concepts. Moving closer declares an existing concept; moving away suggests the appearance of a novel concept. The two main processes in the OBSERVER are for declaring existing concepts and observing suspected novel concepts.

### 6.2 Filtration process

The OUT value of *VINPos* denotes the position of the merged data centroid OutsideALL clusters. However, the OUT value does not always indicate the appearance of a novel concept. In order to find a genuine OUT, we have to understand the genuine reason that causes the merged data to reside outside. Therefore, we can decide whether the accumulated data is suspected novel concept or existing one with concept drift. We defined genuine out as follows:

**Definition 1** *(genuine out (genOUT)): VINPos for observing $Rep_n$ and $Rep_k$ has a genuine out value if the position of both repositories in addition to the merged position are OutsideALL.*

Table 3 states the possible scenarios that cause OUT and genOUT values for VINPos.

Table 3: Cases When *VINPos* is OUT and genOUT

| $Rep_n$ Pos | $Rep_k$ Pos | Merged Pos | VINPos |
|---|---|---|---|
| $Inside_{ci}$ | $OutsideALL$ | $OutsideALL$ | OUT |
| $InSlack_{ci}$ | $OutsideALL$ | $OutsideALL$ | OUT |
| $OutsideALL$ | $OutsideALL$ | $OutsideALL$ | genOUT |

According to Definition 1, *VINPos* may reside outside with no genuine reason, because of one or more of the following. (i) one of the data repositories represents noise, (ii) one of the repositories is an outlier, (iii) repositories refer to interleaved classes (in an AR context, interleaved classes commonly occur especially with strongly related activities such as "Walk" and "Stand"), and/or (iv) there is an error in positioning repositories because of the shifting in BLM clusters boundaries due to concept drift. The filtering process of non-genuine OUT cases resolves the first three issues. Whereas, the adaptation of the BLM resolves the last case.

### 6.3 Detecting a novel concept

Once genuine OUT (genOUT) data has been detected, we perform further analysis to ensure the arrival of a novel concept. Although genOUT indicates the virtual position of OUT and for a genuine reason, it does not indicate the actual movement from all existing clusters. The novel concept is the one that is genOUT and "moves away" from all existing BLM clusters. The suspected novel concept has also to satisfy a set of stability criteria, to be declared as novel.

To materialise the notation of moving away, the novelty detection technique considers data in $Chunk_{t_n}$ as more recent along the time line than data in the *Buffer*. Thus, the algorithm computes the distances from the $Chunk_{t_n}$ centroid to BLM cluster centroid and compare them to the distance between the

*Buffer* and cluster centroid for all clusters existing in the BLM. The increase of distance depicts the away movement of data while time evolves. Figure 10 explains an example of the detection of away movement from a BLM containing four clusters. $Di_t$ is the distance between the *Buffer* centroid (up to time t) and the $C_i$ centroid, while $Di_{t+1}$ is the distance between the centroid of new data chunk arrives at time (t+1) and the $C_i$ centroid. As shown in Figure 10, the moving away criterion is satisfied only when $Di_{t+1} > Di_t \forall C_i \in BLM$.

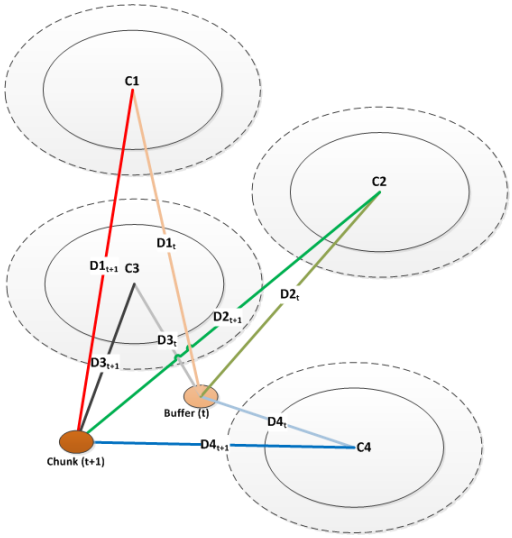



Fig. 10: Away Movement of Novel Concept

A moving away concept is a suspected novel class that must satisfy a set of stability criteria, to be declared as a newborn novel concept. The stability criteria that concern both separation and cohesion of the novel concept are weight, density and minimal centroid movement. Suspected data that satisfies all stability conditions is declared as a newborn novel concept. A newborn novel concept adapts the BLM to facilitate the recognition of future recurring occurrences of the newborn concept. The OBSERVER can also detect more than one novel cluster, as long as it follows the same procedure and satisfies the same criteria.

## 7 Model Adaptation

Upon a decision declaration with OBSERVER, we adapt the BLM with the most recent changes detected in the streams. Both decisions of novel and unknown are eligible for model adaptation, while the decision of existing is not. Declaring a novel concept includes the first appearance of the newborn concept

and also recurring occurrences that belong to the same novel concept. When we detect the appearance of the novel concept at the first place, we adapt the BLM by adding the newborn novel concept to the BLM. We do that by simply forming a cluster of the data that is declared as a newborn concept and generate sub-clusters of the newborn cluster. We then extract all characteristics of the newborn cluster and its sub-clusters and incorporate them into the BLM characteristics. Our algorithm assimilates a newborn concept into the BLM. Therefore, it enables classification of recurring occurrences of the novel concept and distinguishes it from existing concepts.

The novel concept, being the newborn class in AnyNovel, by definition is the one that is not well presented so far in the stream. To address problems with unbalanced classes, AnyNovel captures the novel concepts using small-sized chunks that can closely monitor the movement and detect the arrival of new concepts as soon as they arrive. Once the novel concept is detected, in order to strengthen and boost its presentation, we incrementally accumulate the newborn cluster with the automatically recognised recurring data instances. The detection of recurring instances is part of the learning algorithm as the newborn concept is integrated into the BLM after adaptation. We apply incremental learning only for the newborn cluster, in order to enrich its presentation with more data. The incremental learning approach updates the characteristics of the newborn cluster to reflect the accumulation of new data until the cluster grows heavier. We assume that the cluster is heavy weighted if its weight exceeded the minimum weight of any of BLM clusters. A heavy weighted cluster is considered as a normal cluster instead of being newborn; therefore no more incremental learning is applied.

The other decision that triggers BLM adaptation is the declared unknown decision. The most common cause of the unknown decision is reaching the limit of the *Buffer* size with insufficient information to make a decision. Therefore, the data in the *Buffer* in this case is uncertain and thus it requires model adaptation to accommodate for it. Other reasons for unknown decision are noise and outliers. We incorporate an active learning approach in AnyNovel to ask the user to label the unknown data. When uncertain data exits, active learning is required. Active learning triggers inquiries in the form of unannotated instances to be annotated by user. When the inquiries have been processed, the BLM is adapted in batches accordingly with the true labels of the data.

Active learning in AnyNovel is triggered in two cases. The first case occurs with the most uncertain data which refers to an unknown decision. Also, active learning is provoked when a novel concept/outdated concepts are first detected. The detection of recurring concepts does not require active learning as it is part of the learning process.

### 7.1 Forgetting outdated concepts

The set of concepts is not static along the stream. As novel concept may emerge, outdated concepts might disappear. In order to keep the BLM up to

date with the most recent changes in the stream, we aim also to detect the abandoned concepts and exclude it from the BLM automatically. Therefore, AnyNovel automatically assigns a time stamp for each concept that appears in the stream. Then, the algorithm scans the BLM periodically for outdated concepts and removes them from the BLM.

The process of concept forgetting has two stages: detection and validation. The detection stage is implemented as follows. While AnyNovel continuously monitors the evolution of concepts, three decisions are declared: existing, novel, and unknown. In case of an existing decision, AnyNovel automatically assigns a time stamp for each activity that appears in the stream and is classified as existing. The algorithm scans the BLM periodically for existing clusters that have not been stamped for a long time. When the outdated activity is detected, AnyNovel triggers an inquiry in an active learning approach to get user confirmation for removing the abandoned activity from the BLM (the validation stage). Deletion of the activity is simple as it only removes the characteristics that represent the abandoned activity from the BLM.

## 8 Experimental Study

### 8.1 Experimental setup

We conducted our experiments on benchmarked datasets for the use case of AR: OPPORTUNITY [Roggen et al., 2009] and WISDM [Kwapisz et al., 2011]. The setup for analysing and evaluating AnyNovel aims essentially to test its efficiency in detecting the appearance of new concept(s) and its concurring instances in data streams with concept drift. In the context of AR, concept drift is crystallised when comparing sensory data from different users. Therefore, in all our experiments, part of the data is used to build the BLM while other data from a different user is applied for testing. The underlying BLM is built from training data of multi classes. The default chunk size is 20 data instances unless otherwise stated. We handle the data stream with a continuous sliding window. We also apply an online clustering for each chunk in order to separate concurrent concepts appearing in a single chunk. we first define the base metrics applied for measuring the performance. $TP$, the number of novel instances correctly detected as novel; $TN$, the number of existing instances correctly classified as existing; $FP$, the number of existing instances falsely classified as novel; $FN$, the number of novel instances incorrectly classified as existing; and *nInstances*, total number of instances in the stream. *nInstances* include $TP$, $TN$, $FP$, $FN$, and *unknown* instances. We refer to total number of unknown instances as *NumUnk*. Performance measures are described in Table 4.

***AnyNovel parameters:*** Our developed technique can be applied to any data domain when its data represents sequential data that can be also non identically distributed. AR is one of these domains. In the context of AR, AnyNovel is a learning technique for concept evolution that is independent

Table 4: Performance Measures for COSTAR

| Name | Description | Formula |
|---|---|---|
| Acc | Accuracy as the percentage of correct classification along the stream | $\frac{TP+TN}{nInstances}$ |
| ERR | Error rate in the data stream | $\frac{FP+FN}{nInstances}$ |
| UnkPerc | Percentage of instances that are declared as unknown | $\frac{NumUnkn}{nInstances}$ |
| Recall | Ratio of instances detected as novel among all true novel instances | $\frac{TP}{TP+FN}$ |
| Precision | Ratio of correctness in the examples classified as novel | $\frac{TP}{TP+FP}$ |
| Specificity | Ratio of instances classified as existing among all true existing instances | $\frac{TN}{TN+FP}$ |
| Fall-out Rate (FOR) | False alarm rate | $\frac{FP}{FP+TN}$ |
| False-discovery Rate (FDR) | Ratio of false positives among all instances classified as novel | $\frac{FP}{FP+TP}$ |
| ALRate | Number of triggered active learning inquiries per 10 thousand data instances | - |
| ALPoints | Percentage of total number of instances that are batch labelled with active learning inquiries | - |
| CPur | Percentage of the cluster purity, which is the percentage of instances with the major label within the cluster | - |

on the sensing scheme. Thus, AnyNovel can be applied to a wide range of datasets. Nevertheless, AnyNovel parameters have to be tuned to suit some of these characteristics for better performance. The parameters are the chunk size, initial DRAB height, and stability weight of the novel concept. One of these parameters is the chunk size. The *Buffer* size is much associated to the chunk size in AnyNovel. Information about dataset sampling rate and minimum duration of any of the activities/class are essential to set a good chunk size. A small *Buffer* capacity might increase the percentage of unknown data. On the other hand, a large *Buffer* capacity will increase the complexity. Therefore, to allow a reasonable time for monitoring the data movement, we set the capacity of the *Buffer* to accommodate the maximum of double the size of a data chunk. In the following experiments, we manually tune the

parameters for each dataset in order to adapt to the dataset characteristics and thus improve the performance of AnyNovel. We aim in the future research to automate the tuning for all parameters.

### 8.2 Complexity analysis

AnyNovel online flow is distributed between two threads of processing: i) capturing the data stream and performing online clustering for each chunk (for separating concurrent concepts); and ii) monitoring the movement and the adaptation processes . In terms of time complexity, the two processes are implemented in parallel. That means, while the stream chunks are being captured by the sliding window, the chunks from the previous window are being processed by the adaptation component.

The complexity in the first process is from the online clustering. We apply EM clustering for each chunk with a complexity of O($nki$), where $n$ is the number of instances in each chunk, $k$ is the number of clusters and $i$ is the number of iterations. As the chunk size is set to be small (represents 1-4 seconds of the stream depending on the sampling rate), we limit the number of interleaved, concurrent, or transitional activities for each chunk to only 2, which is very reasonable based on the small window size. Thus, the complexity for clustering data in each window is O($2ni$), with $n$ is a very small number of instances in each chunk, around 20 instances.

The second process of AnyNovel consists of the two components of Cohesion Validation Test (CVT) and OBSERVER. The CVT is another pass of online clustering that runs on a maximum of $m$ data instances, where $m$, in the worst-case scenario, is the maximum *Buffer* capacity. Therefore, the complexity of CVT is O($2mi$). The second component in the online phase, OBSERVER, does not require an extra processing overhead as it only performs a set of comparisons to decide on the data movement. The complexity of the OBSERVER component is O(1) as it is independent on the number of instances.

We have analysed the results based on the above discussion. AnyNovel is deployed on a desktop machine with the following specifications: Core i7 with speed of 2.7 GHZ and 4 GB RAM. The data dimensions influence the processing time in AnyNovel. Therefore, We start with analysing AnyNovel processing time with 3D data collected from a mobile accelerometer sensor in WISDM. The sampling rate of WISDM is 20 Hz. We also vary the chunk size to monitor its impact on the processing time. Table 5 shows the average processing time with different chunk sizes on WISDM dataset. The processing time of a chunk with 20 instances, which is equal to an observation time of 1 second, is 67 msec. As the chunk size increased, the observation time extends as well as the processing time. Yet, the processing time is always less than the observation time. As the sampling rate in the dataset is longer than the processing time across small and big chunks of data, AnyNovel successfully fulfils the real time constraint in this dataset.

Table 5: AnyNovel time analysis with WISDM dataset

| Chunk size | Window (ms) | Chunk processing time (ms) | Processing Rate (Instance/second) |
|---|---|---|---|
| 20 | 1000.0 | 67 | 298.51 |
| 50 | 2500.0 | 70 | 714.29 |
| 100 | 5000.0 | 126 | 793.65 |
| 500 | 25000.0 | 268 1 | 865.67 |

Unlike, the 3D datasets, the number of features in the OPPORTUNITY dataset is 110. The sampling rate for the OPPORTUNITY dataset is 30 Hz (approximately 1 sample every 33 msec). Thus, the observation time for a chunk of 20 instances is about 667 msec. Although the processing time increases in the OPPORTUNITY compared to the 3D data in WISDM, AnyNovel still performs in real time despite the high data dimensionality. The processing time as short as 0.05 to 0.3 of the observation time for all chunk sizes. The processing rate shows the capacity for AnyNovel to process more data instances once they are available. It can successfully process more than 670 instance/second when the window size is 16 seconds (with the 500 chunk size).

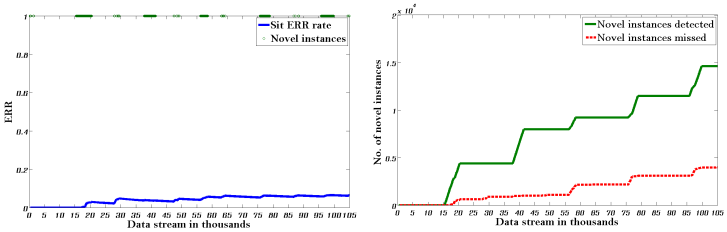
Table 6: AnyNovel time analysis with the OPPORTUNITY dataset

| Chunk size | Window (ms) | Chunk processing time (ms) | Processing Rate (Instance/second) |
|---|---|---|---|
| 20 | 666.7 | 227.5 | 87.91 |
| 50 | 1666.7 | 348 | 143.68 |
| 100 | 3333.3 | 523.5 | 191.02 |
| 500 | 16666.7 | 745.5 | 670.69 |

### 8.3 OPPORTUNITY dataset

The OPPORTUNITY dataset consists of data for the four atomic activities of “Sitting”, “Walking”, “Standing”, and “Lying” represented with 110 features. Data is collected for 5 unsegmented sessions for four subjects. Subset of the data contains rational and additive noise to evaluate the robustness of AR techniques to noise. To be able to evaluate the AnyNovel performance on data streams with concept drift, in the following experiments, the data used for building BLM and testing is from different users. Evaluation of the detection of a novel concept is performed for each activity. The sampling rate for the OPPORTUNITY dataset is 20 Hz. We set the chunk size for all experiments to 50 instances (2.5 seconds) and the weight of stable novel concepts to 300 instances. DRAB height is tuned based on $Dmax$ values of BLM exiting clusters in the interval (1800–3000). Figures 11, 12, 13, and 14 show various

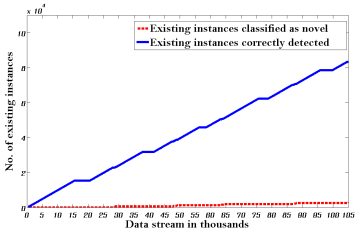
performance metrics for detecting each activity as novel. Sub-figure (a) shows the ERR rate of AnyNovel up to a certain point in the stream for each activity type. ERR is the total percentage of misclassified instances (%) in the stream that includes FP, FN, and unknown. The arrival of the novel activity instances is marked at the top border in Sub-figure (a) in all figures. The number of correctly detected novel instances ($TP$) and missed novel instances ($FN$) are displayed in Sub-figure (b) for each activity. Whereas Sub-figure (c) shows the correlation between the classification of existing classes correctly ($TN$) and false discovery of existing class as novel ($FP$). In general, the activities of "Sitting" and "Lying" achieved higher performance over "Walking" and "Standing". In Figures 11(b) and 12(b), the number of novel instances detected ($TP$) increases steadily over time. Whereas, the novel instances missed ($FN$) are at the lower bound along the stream with a slow increase of a small constant rate. The error graph for the "Lying" activity, Figure 12(a), starts with a high error rate and continues until the first correct detection of the novel activity. Once the novel concept is declared, recurring instances belonging to the novel concept are correctly classified and thus the error rate drops over time. The specificity relationship in Figure 11(c) and 12(c) indicates the AnyNovel ability to identify existing instances correctly. The rate of $FP$ is slightly increasing by a very small amount, along the stream which indicates a high percentage of specificity.




(a) ERR along the Data Stream



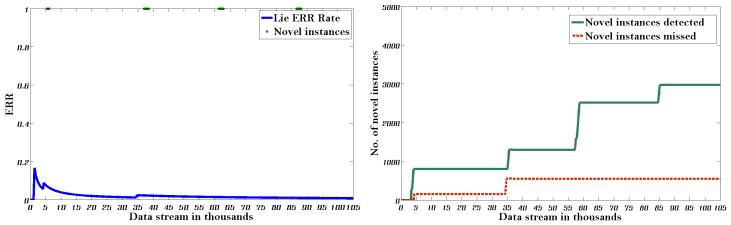

(b) Detection of Novel Instances



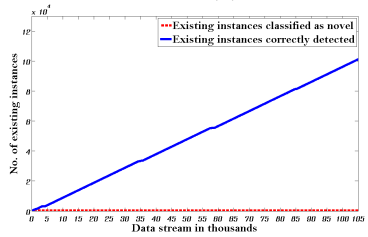

(c) Detection of Existing Instances

Fig. 11: AnyNovel Performance in Detecting "Sitting" Activity

"Walking" and "Standing" activities are more challenging. Many reasons are behind the confusion between theses two activities. Both activities are strongly correlated and interleaving. That means combining and switching of the two activities are frequently occurring. Also, novel concepts that represent



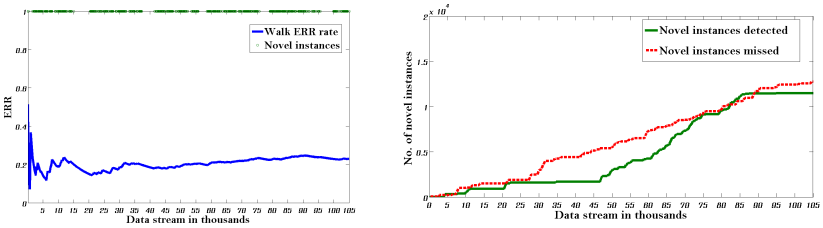

(a) ERR along the Data Stream (b) Detection of Novel Instances



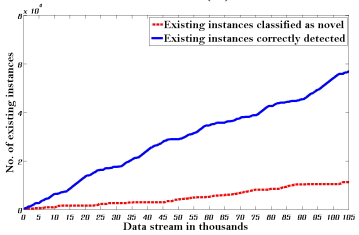

(c) Detection of Existing Instances

Fig. 12: AnyNovel Performance in Detecting "Lying" Activity

these activities are relatively big with diverse patterns embedded inside. For instance, the "Walking" activity might contain patterns of strolling, jogging, and normal pace walking. The strolling pattern of "Walking" is commonly interleaved with "Standing" (as pauses) while "Walking".

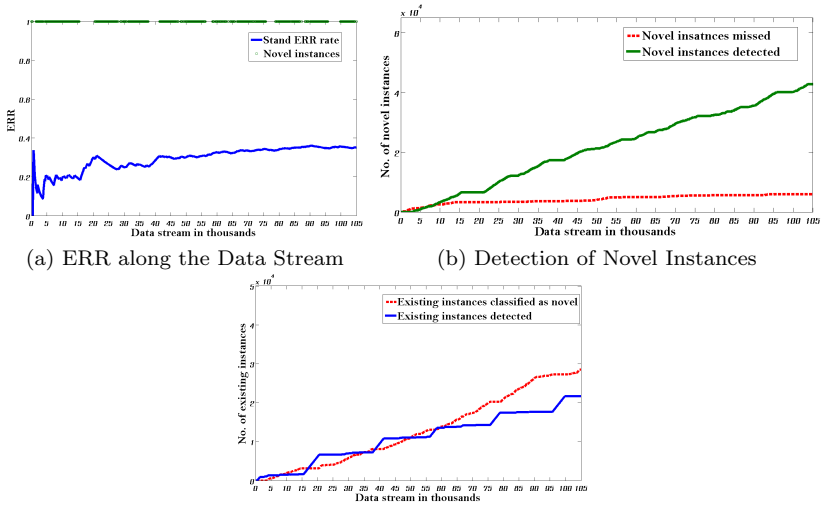



(a) ERR along the Data Stream (b) Detection of Novel Instances




(c) Detection of Existing Instances

Fig. 13: AnyNovel Performance in Detecting "Walking" Activity




(a) ERR along the Data Stream

(b) Detection of Novel Instances

(c) Detection of Existing Instances

Fig. 14: AnyNovel Performance in Detecting "Standing" Activity

In order to visualise the OPPORTUNITY dataset, we have applied Principle components analysis and transformation of the data. Dimensionality reduction is accomplished by WEKA Principle Component Analysis (PCA) [Witten and Frank, 2005] by choosing enough eigenvectors to account for 95% of the variance in the original data. Therefore, the reduced dimension OPPORTUNITY sample data can be visualised with a 3D graph as displayed in Figure 15. The graph shows the clear overlapping between "Standing" and "Walking" activities that would justify the confusion in decision between the two activities. It also shows the location of "Sitting" activity inside global decision boundary between "Lying" from one side and "Standing" and "Walking" from the other side.

In Table 7, we summarise the performance of AnyNovel for recognising each activity in the OPPORTUNITY dataset. The BLM for all experiments consists of all activities except the tested novel activity. Each activity has its own weight in the test data. The activity weight is the percentage of instances that represent the activity in the stream.

The accuracy of distinguishing between existing and novel when "Lying" is the novel activity attains the best performance with more than a 99% accuracy rate. Despite the small weight of the "Lying" class (only 5%), AnyNovel can effectively detect both existing and novel activities with recurring occurrences. AnyNovel also attains an accurate recognition, of 93.1%, in the case of a bigger class of "Sitting", with weight 24%. The precision in detecting the novel activity of "Sitting" is 84%, mainly because of the false detection rate. The accuracy in distinguishing between existing and novel is lower with "Standing" and "Walking" activities.

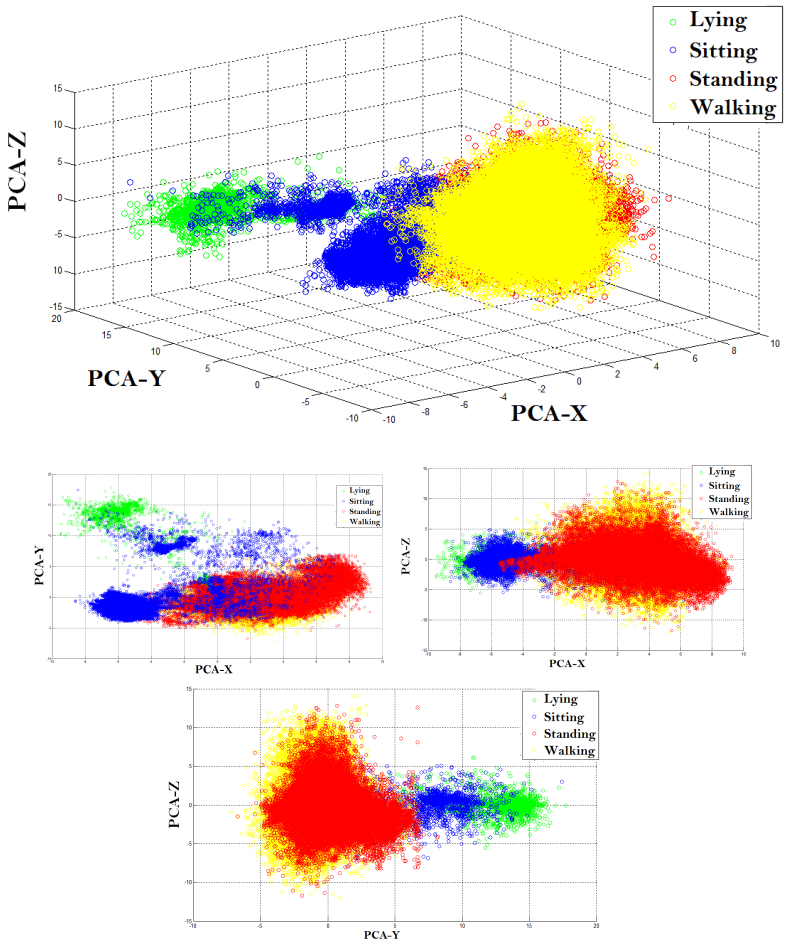



Fig. 15: Visualisation of the OPPORTUNITY Dataset

Although the "Walking" activity has a similar weight of 26% as "Sitting", its detection accuracy of 70.4% is lesser. The false detection rate of the "Walking" activity is high due to the confusion with the "Standing" activity. The low recall percentage indicates the misclassification of many novel instances as existing. Thus, the recall percentage that measures AnyNovel sensitivity drops to $\simeq$ 17 % for "Walking", while the false-discovery rate (FDR) reaches 40%. The "Standing" activity in this dataset is diverse and heavy with a weight of 46%. The accuracy in detecting novel and existing activities in the case of "Standing" is 61.3%. Although, the recall percentage reaches more than 85%, opposite to "Walking" performance, specificity is dropped to $\simeq$ 40% due to the high fall-out rate of 56.8%.

Cluster purity, which measures the percentage of major labels inside declared clusters, achieves more 90% in all runs. Cluster purity indicates the average purity of the cluster when declared. The high purity percentage demonstrates the effective selection of parameters, specifically the chunk size and stable size. The purity in the"Walking" and "Standing" clusters is lower than other activities, which indicates that the overlapping between these two activities results in less pure clusters.

Table 7: AnyNovel Performance with the OPPORTUNITY Dataset

| Novel activity | Acc | Recall | Specf | Prec | FOR | FDR | CPur |
|---|---|---|---|---|---|---|---|
| Sitting (%) | 93.1 | 78.5 | 96.8 | 84.0 | 3.2 | 16.0 | 98.0 |
| Lying (%) | 99.2 | 86.5 | 99.8 | 94.0 | 0.2 | 6.0 | 97.3 |
| Walking (%) | 70.4 | 17.2 | 95.7 | 61.9 | 4.3 | 38.1 | 93.2 |
| Standing (%) | 61.3 | 87.8 | 43.2 | 60.0 | 56.8 | 40.0 | 93.4 |

The analysis and discussion on the OPPORTUNITY dataset show the AnyNovel ability to distinguish between existing and novel activities in data streams with concept drift. The error rate might be high at the beginning of the stream until the first appearance of the novel activity is successfully detected. The error rate decreases gradually afterwards, which indicates an efficient adaptation of the model to recognise recurring instances of the detected novel concept. The evaluation demonstrates AnyNovel ability to recognise activities that are not well-presented in the stream, i.e., small sized activities, such as "Lying" in OPPORTUNITY. The experiments also show the trade-off between recall and precision, especially in the "Walking" and "Standing" activities.

### 8.4 WISDM dataset

In this section, we discuss AnyNovel performance with another AR dataset, but collected from mobile accelerometer sensor. The WISDM dataset contains five annotated activities with only three features: x, y, and z accelerometer components. Data is collected from many users and contains more than a million instances. For each row in Table 8, we aim to detect a novel activity with a combination of users while ensuring the novelty of activities in test data. Performance metrics are reported in Table 8 for each novel activity across the different users. The set of parameters used for WISDM evaluation is as follows: chunk size=10, stable size=20, while DRAB height is in the interval (0.5–3).

AnyNovel achieves its highest accuracy in detecting the "Sitting" activity with an accuracy of 99.8%. The remaining 0.2% of instances are declared as unknown. Although the weight of the "Sitting" class in the incoming stream is only 7%, AnyNovel can detect the appearance of the novel concept and its recurring instances."Sitting" is a static activity that does not require body

Table 8: AnyNovel Performance on Different Users of the WISDM Dataset

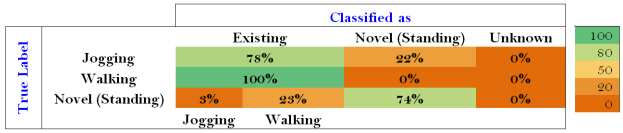
| Novel | Acc | Recall | Specf | Prec | FOR | FDR |
|---|---|---|---|---|---|---|
| Sitting (%) | 99.8 | 100 | 100 | 100 | 0 | 0 |
| Standing (%) | 81.1 | 74.2 | 85.4 | 76.3 | 14.5 | 23.7 |
| Walking (%) | 83.5 | 81.4 | 84.5 | 70.5 | 15.5 | 29.5 |
| Stairs (%) | 71.3 | 73.9 | 96.9 | 96.7 | 3.1 | 3.2 |
| Jogging (%) | 84.5 | 69.6 | 93.0 | 84.3 | 7.0 | 15.6 |

movement. Therefore, the detection of the "Sitting" activity is attained with high accuracy despite its small size.

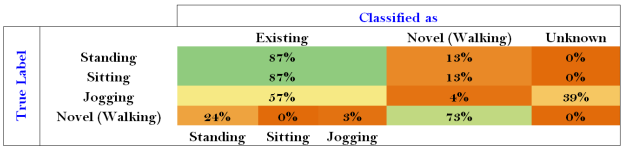
The distinction between existing activities and novel activity of "Standing" is more challenging, especially when training the BLM with data containing "Walking" and "Jogging" activities, as in the user 2 data. Both "Walking" and "Jogging" are tightly related to "Standing", which makes it hard to detect the appearance of the "Standing" activity. Periods of "Walking" or "Jogging" would be overlapped with pauses of "Standing". AnyNovel can distinguish between "Standing" as a novel activity and existing "Jogging" and "Walking" activities with an accuracy of 80%. However, both the false detection rate (FDR) and fall out rate (FOR) are at high percentages (23.7 and 14.5 respectively). That shows the confusion occurs in detecting "Standing" activity as existing and also the detection of existing instances as novel. The confusion matrix of detecting "Standing" activity is illustrated in Figure 16(a). Novel, in all matrices, shows both FA and recurring concepts. The results show that 74% of novel instances of "Standing" are correctly detected as novel. The first reason for confusion is because of the misclassification of the "Standing" activity as existing. In this experiment, the ambiguity between "Standing" and "Walking" results in more than 23% of the novel instances misclassified as "Walking". On the other hand, the existing "Walking" instances are correctly classified as existing. Another reason of confusion is in detecting the "Jogging" activity as novel ("Standing") for more than 22% of "Jogging" data.

We face the same challenges when detecting the novel activity of "Walking". Both the false detection rate and fall out rate are high (similar to "Standing"). The confusion between "Walking" and "Standing" is the main reason for the fall out rate as shown in Figure 16(b). The confusion matrix also shows that 39% of the "Jogging" activity is declared unknown because of uncertainty in recognition. Also, instances of "Standing" and "Sitting" are falsely detected as novel ("Walking").

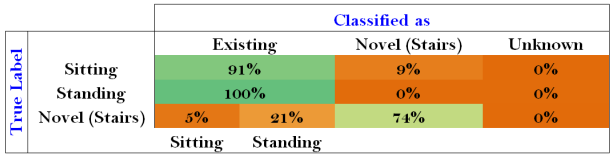
We examine AnyNovel performance in detecting the "Stairs" activity. The results show an overlapping between the large class of "Stairs" and other classes. AnyNovel sensitivity of detecting the novel activity is 74%, indicating a confusion with existing. In the confusion matrix of the "Stairs" activity, shown in Figure 16(c), it is clear that the main ambiguity occurs between "Standing" and "Stairs". This can be justified by the natural pauses of "Stand" that occur while climbing up and down the stairs.



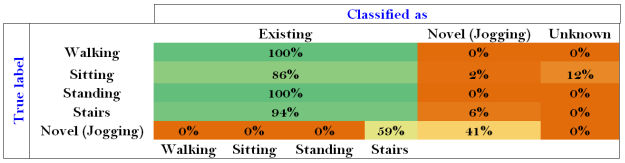

(a) "Standing" Activity




(b) "Walking" Activity




(c) "Stairs" Activity




(d) "Jogging" Activity

Fig. 16: Confusion Matrices on the WISDM Dataset

The two activities of "Jogging" and "Stairs" are naturally similar. As illustrated in the confusion matrix in Figure 16(d), the reason for the lower recall rate in "jogging" is the confusion with "Stairs". Despite the efficiency of AnyNovel to detect existing activities, the correct detection of a novel "Stairs" activity is only 41% of the instances representing the novel activity.

## 8.5 Active learning

Active learning in AnyNovel is provoked in two cases, *unknown* data and the *first appearance* of a novel activity (FA). In Figure 17, the results show the two metrics that evaluate active learning performance. ALRate represents the average number of inquiries triggered for every 10K instances in the stream. The smaller the ALRate, the more efficient is our technique. ALPoints refers to the total number of instances that require active learning and thus feed back to adapt the BLM. The two metrics include the two cases of active learning.

In the OPPORTUNITY dataset, the rate of active learning needed is very low in the "Lying" and "Sitting" activities. There is a small percentage of *unknown* instances noted in these two activities. Thus, the low active learning

rate and small percentage of ALPoints correspond mostly to the FA case. The rate is slightly higher in the "Walking" and "Standing" activities as they are more challenging activities to be recognised. Yet, the ALRate is very small at 3.8 and 2.6 inquiries per 10k instances in "Walking" and "Standing" respectively.

| Activity | ALRate | ALPoints |
|---|---|---|
| Sitting | 1.7% | 3.4% |
| Lying | 0.3% | 0.4% |
| Walking | 3.8% | 7% |
| Standing | 2.6% | 7% |

(a) OPOORTUNITY dataset

| Activity | ALRate | ALPoints |
|---|---|---|
| Stairs | 6.5% | 6.7% |
| Jogging | 0.3% | 9.8 % |
| Walking | 3.1% | 4.4% |
| Standing | 22.8% | 9% |
| Sitting | 0.3% | 0.1% |

(b) WISDM dataset

Fig. 17: Active Learning with Different Datasets

The WISDM dataset results show a low ALRate in "Jogging", "Sitting", and "Walking" activities. The high percentage of the ALPoints rate in "Standing" is because of the high percentage of $FP$ as per the aforementioned discussion on the confusion matrix. Despite the low rate of active learning in "Jogging", the size of batches that require active learning is big and therefore the ALPoints are as high as 9.8%. We can conclude that, most results show an efficient performance of the batch active learning technique with a low ALRate in most of the cases.

## 9 Conclusion and Future Work

The number of concepts in some data stream applications changes over time. New concepts might emerge; irrelevant concepts might disappear. Training the learning model with a static number of concepts is impractical and inefficient especially in a streaming environment. The capability of the learning model to adapt for extension and prune continuously to reflect the changes of concepts in data streams is essential for accurate and efficient performance.

In this paper, we have proposed, developed, and evaluated our AnyNovel technique for detecting concept evolution in evolving data streams. AnyNovel applies a continuous learning approach for monitoring the evolution in the stream and thus detects the expected changes. The detected changes include novel normal and abnormal concepts. AnyNovel is capable of adaptation with the changes by detecting recurring novel concepts and abandoned concepts too. It comprises mainly of two components: the Cohesion Validation Component (CVC) and OBSERVER. While the CVC checks the dependency in the stream, OBSERVER monitors the movement of data over time. The learning model is adapted incrementally and continuously to reflect the detected changes. AnyNovel also addresses the scarcity of labeled data by incorporating an active learning approach for labelling only the most uncertain data in the stream.

The evaluation of AnyNovel on a case study of AR with benchmarked datasets demonstrated its ability to detect novel concepts and its recurring instances with learning model that contains multi-concepts. The recognition performance showed its efficiency in detecting the small sized classes that are mostly overlooked and thus misclassified. The results also showed the efficiency of the model adaptation technique to refine the model with the expected changes. AnyNovel can distinguish with good accuracy between concept drift and novel concepts. We discussed also in the results the active learning percentage for different activities. The rate of active learning inquiry is noted to be small with different experiments. That indicates efficient performance of the batch active learning approach.

AnyNovel parameter tuning is crucial for obtaining a good performance. We aim in our future work to automate the selection of AnyNovel parameters for more efficient recognition. Applying continuous and active learning for activity recognition is still new and promising as it aims at less supervision in the recognition process and addresses the known issue of scarcity of labelled data. Thus, work in this area represents a new direction for research

## References

[Abdallah et al., 2015] Abdallah, Z. S., Gaber, M. M., Srinivasan, B., and Krishnaswamy, S. (2015). Adaptive mobile activity recognition system with evolving data streams. *Neurocomputing*, 150, Part A:304 – 317.

[Aggarwal, 2013] Aggarwal, C. C. (2013). *Outlier Analysis*. Springer New York.

[Al-Khateeb et al., 2012] Al-Khateeb, T., Masud, M. M., Khan, L., Aggarwal, C. C., Han, J., and Thuraisingham, B. M. (2012). Stream classification with recurring and novel class detection using class-based ensemble. In *ICDM*, pages 31–40.

[Andreu and Angelov, 2013] Andreu, J. and Angelov, P. (2013). An evolving machine learning method for human activity recognition systems. *Journal of Ambient Intelligence and Humanized Computing*, 4(2):195–206.

[Andreu et al., 2011] Andreu, J., Baruah, R. D., and Angelov, P. (2011). Real time recognition of human activities from wearable sensors by evolving classifiers. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 2786–2793. IEEE.

[Angiulli and Fassetti, 2007] Angiulli, F. and Fassetti, F. (2007). Detecting distance-based outliers in streams of data. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 811–820, New York, NY, USA. ACM.

[Assent et al., 2012] Assent, I., Kranen, P., Baldauf, C., and Seidl, T. (2012). Anyout: Anytime outlier detection on streaming data. In *Proceedings of the 17th International Conference on Database Systems for Advanced Applications - Volume Part I*, DASFAA'12, pages 228–242, Berlin, Heidelberg. Springer-Verlag.

[Cauwenberghs and Poggio, 2001] Cauwenberghs, G. and Poggio, T. (2001). Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, pages 409–415.

[Costa et al., 2014] Costa, B. S. J., Angelov, P. P., and Guedes, L. A. (2014). Real-time fault detection using recursive density estimation. *Journal of Control, Automation and Electrical Systems*, 25(4):428–437.

[Costa et al., 2015] Costa, B. S. J., Angelov, P. P., and Guedes, L. A. (2015). Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. *Neurocomputing*, 150:289–303.

[Faria et al., 2013] Faria, E. R., Gama, J. a., and Carvalho, A. C. P. L. F. (2013). Novelty detection algorithm for data streams multi-class problems. In *Proceedings of the 28th*

*Annual ACM Symposium on Applied Computing*, SAC '13, pages 795–800, New York, NY, USA. ACM.

[Faria et al., 2015a] Faria, E. R., Gonçalves, I. J., de Carvalho, A. C., and Gama, J. (2015a). Novelty detection in data streams. *Artificial Intelligence Review*, pages 1–35.

[Faria et al., 2015b] Faria, E. R., Ponce de Leon Ferreira Carvalho, A., and Gama, J. (2015b). Minas: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, pages 1–41.

[Gomes et al., 2012] Gomes, J. B., Krishnaswamy, S., Gaber, M. M., Sousa, P. A. C., and Menasalvas, E. (2012). Mars: A personalised mobile activity recognition system. In *Proceedings of the 2012 IEEE 13th International Conference on Mobile Data Management (Mdm 2012)*, MDM '12, pages 316–319, Washington, DC, USA. IEEE Computer Society.

[Gurjar and Chhabria, 2015] Gurjar, G. and Chhabria, S. (2015). A review on concept evolution technique on data stream. In *Pervasive Computing (ICPC), 2015 International Conference on*, pages 1–3.

[Haque et al., 2015] Haque, A., Khan, L., and Baron, M. (2015). Semi supervised adaptive framework for classifying evolving data stream. In *Advances in Knowledge Discovery and Data Mining*, pages 383–394. Springer.

[Hayat and Hashemi., 2010] Hayat, M. Z. and Hashemi., M. R. (2010). A dct based approach for detecting novelty and concept drift in data streams. In *Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of*, pages 373–378.

[Kifer et al., 2004] Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment.

[Krishnan and Cook, 2014] Krishnan, N. C. and Cook, D. J. (2014). Activity recognition on streaming sensor data. *Pervasive Mobile Computing*, 10.

[Kwapisz et al., 2011] Kwapisz, J. R., Weiss, G. M., and Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82.

[Last, 2002] Last, M. (2002). Online classification of nonstationary data streams. *Intelligent Data Analysis*, 6(2):129–147.

[Lockhart and Weiss., 2014] Lockhart, J. W. and Weiss., G. M. (2014). The benefits of personalized smartphone-based activity recognition models. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 614–622.

[Lughofer and Angelov, 2011] Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2057–2068.

[Luštrek and Kaluža, 2009] Luštrek, M. and Kaluža, B. (2009). Fall detection and activity recognition with machine learning. *Informatica (Slovenia)*, 33(2):197–204.

[Marsland et al., 2005] Marsland, S., Nehmzow, U., and Shapiro, J. (2005). On-line novelty detection for autonomous mobile robots. *Robotics and Autonomous Systems*, 51(2):191–206.

[Masud et al., 2011a] Masud, M. M., Al-Khateeb, T. M., Khan, L., Aggarwal, C., Gao, J., Han, J., and Thuraisingham, B. (2011a). Detecting recurring and novel classes in concept-drifting data streams. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 1176–1181, Washington, DC, USA. IEEE Computer Society.

[Masud et al., 2013] Masud, M. M., Chen, Q., Khan, L., Aggarwal, C. C., Gao, J., Han, J., Srivastava, A., and Oza, N. C. (2013). Classification and adaptive novel class detection of feature-evolving data streams. *Knowledge and Data Engineering, IEEE Transactions on*, 25(7):1484–1497.

[Masud et al., 2011b] Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. (2011b). Classification and novel class detection in concept-drifting data streams under time constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 23(6):859–874.

[Mubashir et al., 2013] Mubashir, M., Shao, L., and Seed, L. (2013). A survey on fall detection: Principles and approaches. *Neurocomputing*, 100:144–152.

[Nguyen et al., 2015] Nguyen, L. T., Zeng, M., Tague, P., and Zhang, J. (2015). Recognizing new activities with limited training data. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 67–74. ACM.

[Niennattrakul et al., 2010] Niennattrakul, V., Keogh, E., and Ratanamahatana, C. A. (2010). Data editing techniques to allow the application of distance-based outlier detection to streams. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 947–952, Washington, DC, USA. IEEE Computer Society.

[Peterek et al., 2014] Peterek, T., Penhaker, M., Gajdo, P., and Dohnlek, P. (2014). Comparison of classification algorithms for physical activity recognition. In *Innovations in Bio-inspired Computing and Applications*, volume 237 of *Advances in Intelligent Systems and Computing*, pages 123–131. Springer International Publishing.

[Pimentel et al., 2014] Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99:215–249.

[Pokrajac et al., 2007] Pokrajac, D., Lazarevic, A., and Latecki, L. J. (2007). Incremental local outlier detection for data streams. In *Computational Intelligence and Data Mining, CIDM. IEEE Symposium on*, pages 504–515. IEEE.

[Preece et al., 2009] Preece, S. J., Goulermas, J. Y., Kenney, L. P., Howard, D., Meijer, K., and Crompton, R. (2009). Activity identification using body-mounted sensorsa review of classification techniques. *Physiological measurement*, 30(4):1–33.

[Rashidi and Cook, 2010] Rashidi, P. and Cook, D. J. (2010). Mining sensor streams for discovering human activity patterns over time. In *IEEE 10th International Conference on Data Mining (ICDM)*, pages 431–440.

[Roggen et al., 2009] Roggen, D., Förster, K., Calatroni, A., Holleczek, T., Fang, Y., Tröster, G., Lukowicz, P., Pirkl, G., Bannach, D., Kunze, K., Ferscha, A., Holzmann, C., Riener, A., Chavarriaga, R., and del R. Millán, J. (2009). Opportunity: Towards opportunistic activity and context recognition systems. In *World of Wireless, Mobile and Multimedia Networks Workshops. WoWMoM. IEEE International Symposium on a*, pages 1–6.

[Schlimmer and Granger Jr, 1986] Schlimmer, J. C. and Granger Jr, R. H. (1986). Incremental learning from noisy data. *Machine learning*, 1(3):317–354.

[Spinosa et al., 2007] Spinosa, E. J., Carvalho, A. C. P. L. F., and Gama, J. a. (2007). Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 448–452, New York, NY, USA. ACM.

[Witten and Frank, 2005] Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2. edition.

[Yang et al., 2002] Yang, Y., Zhang, J., Carbonell, J., and Jin, C. (2002). Topic-conditioned novelty detection. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 688–693, New York, NY, USA. ACM.

[Yeung and Chow, 2002] Yeung, D.-Y. and Chow, C. (2002). Parzen-window network intrusion detectors. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 385–388.