Unsupervised learning approaches for non-stationary data streams

Kemilly Dearo Garcia

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

Kemilly Dearo Garcia

Unsupervised learning approaches for non-stationary data streams

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
Advisor at the Affiliated Institution: Prof. Dr. Joost N. Kok

USP – São Carlos
March 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

```
        Dearo Garcia, Kemilly
D285u      Unsupervised learning approaches for non-
        stationary data streams / Kemilly Dearo Garcia;
        orientador André Carlos Ponce de Leon Ferreira de
        Carvalho; coorientador Joost N. Kok. -- São Carlos,
        2021.
           130 p.


           Tese (Doutorado - Programa de Pós-Graduação em
        Ciências de Computação e Matemática Computacional) --
         Instituto de Ciências Matemáticas e de Computação,
        Universidade de São Paulo, 2021.


           1. Data Streams. 2. Unsupervised Learning. 3.
        Incremental Learning. I. Carlos Ponce de Leon
        Ferreira de Carvalho, André, orient. II. N. Kok,
        Joost, coorient. III. Título.
```

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Kemilly Dearo Garcia

Abordagens de aprendizagem não supervisionada para fluxos de dados não estacionários

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Supervisor: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
Orientador na Instituição Conveniada: Prof. Dr. Joost N. Kok

USP – São Carlos
Março de 2021

Dedicado à comunidade cientifica brasileira,
que em tempos sombrios comos de agora luta contra a ignorância.

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to have had the opportunity to work with Prof. André de Carvalho, my supervisor from the University of São Paulo. Besides being an exceptional professional, his optimism and kindness kept me motivated to move forward. I thank my friends from Biocom Davi Santos, Luis Paulo, Henrique Marques, Daniel Cestari, Everlândio Fernandes, Victor Barella, Adriano Rivolli, Victor Padilha, Rafael Mantovani, Daniel Tozadore, Adam Pinto and Jefferson Oliva for all the good and funny moments.

Despite all the adversities I encountered in Portugal, my passage through the lands of the poet Camões brought me good friends and professional achievements. I would like to thank my friends from LIAAD for the dinners at Baixa do Porto, escape room games, and all the good moments. I specially would like to thank Inês Meyer, Diogo Silva for being my friends and support me during the difficult moments.

I also would like to express my deep gratitude to Prof. Joost Kok, my supervisor of the University of Twente, for giving me the amazing opportunity to work at UT. To Mannes Poel, my daily supervisor, I would like to thank you for your patience and excellent guidance. I also would like to thank Raymond Veldhuis for welcoming me to the Data Science group of UT, for his guidance, and for his wise words. To Bertine, Geert Jan, Maurice, Muhammad, Ilias, Chris, Herson, Roeland, Thijs, Tim, Florian, Valeriu, Claudio, Nicola, Doina for all the good moments during coffee time, lunch, funny pub quizzes, and beers after work. Specially, I would like to thank Una, Meike, and Lu, sharing the office with you was a wonderful experience.

To my dear friends Chiara, Francis, Luca, Dayna, Taki for teaching me to believe in myself.

To my climbing friends, Irana, Michael, and Vera, our weekly meetings kept me strong physically and mentally.

I also would like to thank my parents Rone, Nina, and my brother Renan for all the love and support.

To Ricardo Cachucho for believing in me.

To my friends of the bachelor's degree, Mariana Bernardes, Iasmini Lima, Leidiane Patricia, and Pricila Rodrigues. I specially thank Pricila Rodrigues for the development of the cover of this thesis.

I also would like to thank Lise Stork and Una Kelly for being my good friends and my paranymphs.

My gratitude also goes to my best friend Débora Medeiros, you are my hero and my greatest inspiration.

Finally, I would like to thank CAPES for the financial support which made possible the development of part of this work.

“Se a estrutura não permite um diálogo,
a estrutura deve ser mudada.”
(Paulo Freire)

RESUMO

GARCIA, K. D. **Abordagens de aprendizagem não supervisionada para fluxos de dados não estacionários**. 2021. 132 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

A sociedade moderna está cercada por diversos aplicativos que geram diariamente grandes volumes de dados. Atualmente, qualquer usuário pode monitorar suas atividades físicas, em tempo real, usando seus celulares ou dispositivos vestíveis. Além disso, empresas e governos podem aprender mais sobre seus clientes e cidadãos analisando dados disponíveis em mídias sociais, por exemplo. Esses dados são chamados de *fluxo contínuo de dados* quando são gerados em sequência e continuamente, geralmente em alta velocidade. Esses dados também são potencialmente ilimitados em tamanho e podem não ser estritamente estacionários.

Extrair conhecimento de fluxos de dados é desafiador devido a várias restrições. O fluxo contínuo de dados requer que um algoritmo de aprendizagem atue em ambientes dinâmicos. O que significa que o algoritmo de aprendizagem deve permitir o processamento em tempo real. Além disso, deve ser capaz de se adaptar às mudanças ao longo do tempo, considerando a natureza não estacionária do fluxo de dados.

Nas últimas décadas, muitas abordagens de aprendizado de máquina foram propostas para fluxo contínuo de dados. A maioria dessas abordagens é baseada na aprendizagem supervisionada. Essas abordagens dependem de dados rotulados para adaptar seus modelos às mudanças nos fluxos de dados. No entanto, o processo de rotular os dados costuma ser caro e pode exigir a utilização de especialistas no domínio em questão. Além disso, se os dados forem coletados em alta velocidade, pode não haver tempo suficiente para rotulá-los.

Nesta tese, propomos algoritmos de aprendizado de máquina incremental e não supervisionado para fluxo contínuo de dados. Esses algoritmos são capazes de atualizar seus modelos de classificação com pouco ou sem *feedback* externo. Começamos abordando o problema de mudança de conceito em fluxo contínuo de dados, com poucos dados rotulados. Para esse problema, propomos uma abordagem semi-supervisionada chamada *Sliding Window Clusters*. Este método aprende os padrões atuais do fluxo contínuo de dados selecionando e resumindo os dados mais relevantes. A segunda abordagem é um algoritmo de aprendizagem não supervisionada chamada *Higia* que é capaz de classificar os dados em *normal*, *novidade* ou *mudança de conceito*. Na terceira abordagem presente nesta tese, propomos um algoritmo para combinar diferentes abordagens não supervisionadas em um modelo de classificação. Testamos essa abordagem considerando dois cenários. O primeiro é denominado *Homogeneous Ensemble Clustering para Data Streams* e é baseado na combinação de diferentes execuções do mesmo algoritmo de agrupamento. Neste estudo, também consideramos o cenário denominado *Heterogeneous*

Ensemble Clustering para Data Streams, que se baseia na combinação de diferentes algoritmos de agrupamento de dados. Esses métodos permitem o uso de abordagens de agrupamento com um viés diferente para obter um modelo de classificação mais robusto. Além disso, avaliamos as abordagens do estado da arte, comumente citadas na literatura de detecção de novidades em fluxos de dados.

A maior parte desta tese enfoca abordagens de agrupamento. Porém, dada a popularidade das redes neurais, também propomos o *Ensemble of Auto-Encoders*. Essa abordagem é baseada na combinação de *auto-encoders* em um conjunto de modelos. Cada *auto-encoder* é especializado em reconhecer uma classe particular. O Conjunto de *auto-encoders* possui uma estrutura modular que tem a vantagem de tornar o modelo facilmente adaptado às mudanças dos dados. Além disso, permite modelos personalizados, pois o modelo pode se adaptar às classes mais frequentes. Esta contribuição se aplica ao problema do Reconhecimento da Atividade Humana. Os resultados experimentais mostram o potencial das abordagens mencionadas.

Palavras-chave: Fluxo Continuo de Dados, Aprendizado de Máquina Não-Supervisionado, Aprendizado incremental.

ABSTRACT

GARCIA, K. D. **Unsupervised learning approaches for non-stationary data streams**. 2021. 132 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Modern society is surrounded by several applications which are daily generating large volumes of data. Nowadays, anyone can monitor their physical activities in real-time by using smartphones or wearable devices. Also, business and governments can learn more about their clients and citizens by analysing information from social media, for example. This data is called *data streams* when it is a sequence of data generated continuously, usually at high speed. This data is also potentially unbounded in size and may not be strictly stationary.

Extracting useful knowledge from data streams is challenged due to several constraints. A data stream requires that a learning algorithm acts in dynamic environments. Meaning that the learning algorithm should allow for real-time processing. Moreover, it should be able to adapt to changes over time, considering the non-stationary nature of the data stream.

In the last few decades, many machine learning approaches have been proposed for data streams. Most of them are based on supervised learning. These approaches rely on labelled data to adapt their models to the changes in data streams. However, the process of labelling data is usually costly and can require domain expertise. Besides, if the data is collected at high speed, it may be the case that there will not be enough time to label it.

In this thesis, we aim to propose unsupervised and incremental machine learning algorithms for data streams. We focus on algorithms able to update their classification model with few or without external feedback. We start by addressing the problem of concept drift in data streams with few labelled data. For that problem, we propose a semi-supervised approach called *Sliding Window Clusters*. This method learns the current patterns from the data stream by selecting and summarising the most relevant data. We also study how to learn from data streams when novelties appear over time. So, we proposed an unsupervised learning method called *Higia* which is able to classify data as *normal*, *novelty* or *concept drift*. In this thesis, we propose an approach to combine different unsupervised approaches into a classification model. We test this approach considering two scenarios. The first is called *Homogeneous Ensemble Clustering for Data Streams* and it is based on the combination of different runs from the same clustering algorithm. In this study, we also consider the scenario called *Heterogeneous Ensemble Clustering for Data Streams*, which is based on the combination of different clustering algorithms. These methods allow for the use of clustering approaches with a different bias to obtain a more robust classification model. Furthermore, we evaluate the state-of-art approaches, commonly referred to in the literature of novelty detection in data streams.

Most of this thesis focus on clustering approaches. However, given the popularity of neural networks, we also propose *Ensemble of Auto-Encoders*. This approach is based on the combination of auto-encoders into an ensemble model. Each auto-encoder is specialised on recognising one particular class. The Ensemble of Auto-Encoders has a modular structure that has the advantage of making the model easily adapted to the changes from the data. Besides, it allows for personalised models because the model can adapt to the most request classes. This contribution is applied to the problem of Human Activity Recognition. Experimental results show the potential of the approaches mentioned.

Keywords: Data Streams, Unsupervised Learning, Incremental Learning.

SAMENVATTING

GARCIA, K. D. **Abordagens de aprendizagem não supervisionada para fluxos de dados não estacionários**. 2021. 132 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

In onze moderne samenleving zijn applicaties, die dagelijks grote hoeveelheden data genereren, overal aanwezig. Tegenwoordig kan iedereen zijn of haar fysieke activiteiten monitoren met behulp van smartphones of draagbare apparaten. Verder kunnen bedrijven en overheden meer leren over hun klanten en burgers, bijvoorbeeld door het analyseren van informatie afkomstig van social media. Dit soort data heten *data streams* als ze een reeks data zijn, die continu gegenereerd worden, meestal op hoge snelheid. Een data stream is potentieel onbegrensd in grootte en hoeft niet strikt stationair te zijn.

Het extraheren van nuttige kennis uit data streams wordt bemoeilijkt door meerdere beperkingen. Een data stream vereist dat een *learning algorithm*, een algoritme dat informatie uit een data stream haalt, in een dynamische omgeving ageert. Dit betekent dat dit learning algorithm verwerking in real-time mogelijk moet maken. Bovendien moet het zich kunnen aanpassen aan veranderingen in de loop van de tijd, gezien de niet-stationaire aard van de data stream.

In de afgelopen decennia zijn veel benaderingen die baseren op machine learning voorgesteld voor data streams. De meeste hiervan zijn gebaseerd op supervised learning. Deze aanpakken hebben gelabelde data nodig om hun modellen aan veranderingen in de data streams te kunnen aanpassen. Echter is het proces van het labelen van data duur en kan domeindeskundigheid vereisen. Bovendien kan het zijn dat, als de data met hoge snelheid worden verzameld, er niet genoeg tijd is om te labelen.

In dit proefschrift is ons doel om unsupervised en incrementele machine learning algoritmes voor data streams voor te stellen. We concentreren ons op algoritmes die hun classificatiemodel met weinig of zonder externe feedback kunnen updaten. We beginnen met het aanpakken van het probleem van *concept drift* in data streams met weinig gelabelde data. Voor dit probleem stellen we een semi-supervised benadering voor, die *Sliding Window Clusters* heet. Deze methode leert de actuele patronen uit de data streams door de meest relevante data te selecteren en samen te vatten. We bestuderen ook hoe van data streams geleerd kan worden als novelties (data met nieuwe kenmerken of patronen) na verloop van tijd verschijnen. We stellen een unsupervised learning methode *Higia* voor, die data kan classificeren als *normal*, *novelty* of *concept drift*. In dit proefschrift stellen we een benadering voor, die verschillende unsupervised aanpakken combineert in één classificatiemodel. We testen deze benadering, rekening houdend met twee scenario's. De eerste heet *Homogeneous Ensemble Clustering for Data Streams* en is gebaseerd op het combineren van meerdere runs van hetzelfde clustering algoritme. We kijken ook naar het scenario dat *Heterogeneous Ensemble Clustering for Data Streams* heet, dat gebaseerd is op het combineren van verschillende clustering algoritmes. Deze methodes maken het mogelijk om benaderingen voor clustering, die verschillende biases hebben, te combineren om een robuuster classificatiemodel te verkrijgen. Verder evalueren we state-of-the-art benaderingen, waar vaak naar wordt verwezen in de literatuur over novelty detection in data streams.

De belangrijkste focus van dit proefschrift ligt op aanpakken voor clustering. Echter gezien de populariteit van neurale netwerken stellen we ook een *Ensemble of Auto-Encoders* voor. Deze aanpak is gebaseerd op het combineren van auto-encoders in een ensemble model. Elke auto-encoder is gespecialiseerd in het herkennen van één specifieke klasse. De Ensemble of Auto-Encoders heeft een modulaire structuur, die het voordeel heeft dat het model zich gemakkelijk kan aanpassen aan veranderingen in de data. Bovendien maakt dit gepersonaliseerde modellen mogelijk, omdat het model zich kan aanpassen een de meeste *request classes*. Deze contributie passen wij toe op het probleem van Human Activity Recognition. Experimentele resultaten tonen het potentieel van deze genoemde benaderingen aan.

Trefwoorden: data streams, unsupervised learning, incremental learning.

LIST OF FIGURES

Figure 1 – Classification in different situations. Each geometric figure (circle, diamond and square) represents a concept and the dashed line represents the model. . 31
Figure 2 – Instances X_1 and X_2 are stored within the sliding window. The first instance, X_1, is closer to cluster C_1 and inside its radius. The second instance, X_2, is outside cluster area, but close to the uncertainty border. 47
Figure 3 – SWC accuracy performance of all datasets varying ρ value in 5% to 50%. . 49
Figure 4 – Higia Offline Phase. 57
Figure 5 – Higia Online Phase. 59
Figure 6 – Predictive accuracy over time. 61
Figure 7 – Higia accuracy over time in CDT dataset varying *k*. 62
Figure 8 – Offline Phase with the implementation of the algorithm Ensemble Clustering Generator 70
Figure 9 – Online Phase with the implementation of the algorithm Ensemble Clustering Combiner 72
Figure 10 – MINAS with 3 hyperparameters x F-Measure for Forest Cover dataset . . . 75
Figure 11 – Variability in F-Measures of different number of models of HoCluS for Forest Cover dataset 76
Figure 12 – F-measure of the unsupervised methods in the Gears dataset 77
Figure 13 – F-measure of the unsupervised methods in the SynD dataset 78
Figure 14 – F-measure of the unsupervised methods in the 1CDT dataset 80
Figure 15 – F-measure of the unsupervised methods in the Forest Cover dataset 81
Figure 16 – F-measure of the unsupervised methods in the KDD dataset 82
Figure 17 – F-measure of the unsupervised methods in the MOA dataset 83
Figure 18 – F-measure of the unsupervised methods in the UG_2C_2D dataset 84
Figure 19 – Overview of the ensemble model, E*k*VN, for HAR. 91
Figure 20 – An auto-encoder fully-connected structure with 3 hidden layers. 92
Figure 21 – Overview of the ensemble model, EAE, for HAR. 93
Figure 22 – The reconstruction of the signal by each AE. The color *red* represents the original signal and the color *blue* represents the reconstructed signal. 94
Figure 23 – Frequency per class of each dataset. 94
Figure 24 – Violin box-plot showing the dispersion of accuracy of the models in the dataset WISDM. 97

Figure 25 – Accuracy per user for the EAE and the E*k*VN models for the WISDM dataset considering the body location *Pocket*. 98
Figure 26 – Violin box-plot showing the dispersion of accuracy of the models in the dataset MHealth. 99
Figure 27 – Accuracy per user for the EAE and the E*k*VN models in the MHealth datasets, considering the body location *Hand*. 100
Figure 28 – Violin box-plot showing the dispersion of accuracy of the models in the dataset PAMAP2. 101
Figure 29 – Accuracy per user for the EAE and the E*k*VN models in the PAMAP2 datasets, considering the body location *Hand*. 102
Figure 30 – Overview of the semi-supervised ensemble model for HAR. 110
Figure 31 – Supervised vs Semi-Supervised Ensemble accuracy: Overlap factor influence. 112
Figure 32 – Supervised vs Semi-Supervised Ensemble accuracy: Overlap factor influence. 113
Figure 33 – Supervised vs Semi-Supervised Ensemble accuracy: Overlap factor influence. 113
Figure 34 – Total execution time required (left axis) to process the PAMAP2 dataset, per window size and overlap factor, divided in three parts: sampling (data extraction), features extraction and classification. The number of classifications per configuration (right axis) is shown as triangle marks. 115
Figure 35 – Energy consumed, in Joules, while processing the PAMAP2 dataset, per window size and overlap factor. Green values represent less energy consumed while red values represent higher energy consumed. The values in each configuration represent the accuracy, in percentage, of that configuration. . . 116

LIST OF ALGORITHMS

Algorithm 1 – SWC: Online Window Update 46
Algorithm 2 – Higia: Online Phase 58
Algorithm 3 – Ensemble Clustering Generator 71
Algorithm 4 – Ensemble Clustering Combiner 71

LIST OF TABLES

Table 1 – Characteristics of datasets evaluated. 48

Table 2 – Accuracy and time cost (in seconds) for each method. 50

Table 3 – P-values obtained for the multiple comparison post-hoc Nemenyi test. 50

Table 4 – Statistical information for each dataset 60

Table 5 – Performance metrics for each algorithm 62

Table 6 – Higia ($k = 1,3,5,10,20$) performance metrics for dataset CDT 63

Table 7 – Statistical information for each dataset 73

Table 8 – Example of a confusion matrix 74

Table 9 – Confusion matrix for Gear dataset 77

Table 10 – Confusion matrix for Synd dataset 78

Table 11 – Confusion matrix for 1CDT dataset 79

Table 12 – Confusion matrix for Forest Cover dataset 80

Table 13 – Confusion matrix for KDD dataset 82

Table 14 – Confusion matrix for MOA dataset 83

Table 15 – Confusion matrix for UG_2C_2D dataset 84

Table 16 – Details of the 3 HAR datasets used in this work (A=accelerometer, G=gyroscope, M=magnetometer, C=electrocardiograph). 95

Table 17 – Average Accuracy(Acc) and Time (Train/Test) for the models in the WISDM dataset. 98

Table 18 – Average confusion matrix for the EAE model in the WISDM dataset considering the body location *Pocket*. The columns represent the ground truth and rows represent the predicted. 99

Table 19 – Average Accuracy(Acc) and Time (Train/Test) for the models in the MHealth dataset. 99

Table 20 – Average confusion matrix for EAE model for MHealth dataset considering only the body location *Hand*. The columns represent the ground truth and rows represent the predicted. 100

Table 21 – Average Accuracy(Acc) and Time (Train/Test) for the models in the PAMAP2 dataset. 101

Table 22 – Average confusion matrix for EAE model for PAMAP2 dataset considering only the body location *Hand*. The columns represent the ground truth and rows represent the predicted. 103

Table 23 – Average accuracy for the EAE and the E*k*VN models for each dataset separated by body location (HCA=Hand,Chest,Ankle). 103
Table 24 – Average confusion matrix for EAE model for PAMAP2 dataset considering the aggregation of AEs. 104

LIST OF ABBREVIATIONS AND ACRONYMS

*k*NN	*k*-nearest neighbours
ADWIN	ADaptive sliding WINdowing
AE	Auto-Encoder
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
CF-Vector	Cluster Feature Vector
CNN	Convolutional Neural Networks
E*k*VN	*k*NN, Very Fast Decision Tree and Naive Bayes
EAE	Ensemble of Auto-Encoders
ECSMiner	Enhanced Classifier for Data Streams with novel class Miner
HAR	Human Activity Recognition
HeCluS	Heterogeneous ensemble Clustering for data Streams
HoCluS	Homogeneous ensemble Clustering for data Streams
MINAS	MultI-class learNing Algorithm for data Streams
ML	Machine Learning
MOA	Massive Online Analysis
OLINDDA	OnLIne Novelty and Drift Detection Algorithm
PAW	Probabilistic Approximate Window
SAE	Stacked Auto-Encoder
SAM	Self Adjusting Memory
SWC	Sliding Window Clusters
VFDT	Very Fast Decision Tree

CONTENTS

1 INTRODUCTION . . . 29
1.1 Learning From Data Streams . . . 30
1.2 Research Objective and Research Questions . . . 32
1.3 Contributions of this thesis . . . 33
1.4 Thesis outline . . . 34

2 BACKGROUND . . . 37
2.1 Data Stream . . . 37
2.2 Statistical Summary of Data Streams . . . 38
2.3 Concept Drift . . . 40
2.4 Novelty Detection . . . 40
2.5 Human Activity Recognition . . . 42

3 A CLUSTER-BASED PROTOTYPE REDUCTION FOR ONLINE CLASSIFICATION . . . 43
3.1 Introduction . . . 44
3.2 Related Work . . . 45
3.3 Problem Formalisation . . . 45
3.4 Methodology . . . 46
3.5 Experimental Evaluation . . . 47
3.5.1 Datasets . . . 48
3.5.2 Results and discussion . . . 49
3.6 Conclusion and Future Work . . . 50

4 ONLINE CLUSTERING FOR NOVELTY DETECTION AND CONCEPT DRIFT IN DATA STREAMS . . . 53
4.1 Introduction . . . 54
4.2 Related Work . . . 55
4.3 Problem Formalisation . . . 56
4.4 Methodology . . . 57
4.5 Experimental Evaluation . . . 58
4.5.1 Datasets . . . 59
4.5.2 Results and discussion . . . 60
4.6 Conclusions and Future Work . . . 63

5 AN ENSEMBLE OF UNSUPERVISED APPROACHES FOR NOVELTY DETECTION IN DATA STREAMS . . . 65
5.1 Introduction . . . 66
5.2 Related Work . . . 68
5.3 Problem Formalisation . . . 68
5.4 Ensemble Clustering for Data Streams . . . 69
5.4.1 The MINAS Algorithm . . . 69
5.4.2 Ensemble Clustering Applied To MINAS Algorithm . . . 70
5.4.2.1 Homogeneous Ensemble Clustering for Data Streams . . . 72
5.4.2.2 Heterogeneous Ensemble Clustering in Data Streams . . . 72
5.5 Experimental setup and results . . . 73
5.5.1 Datasets . . . 73
5.5.2 Evaluation . . . 73
5.5.3 Hyperparameter tuning . . . 74
5.5.4 Results . . . 76
5.5.4.1 Gear dataset . . . 76
5.5.4.2 SynD dataset . . . 77
5.5.4.3 1CDT dataset . . . 78
5.5.4.4 Forest Cover dataset . . . 79
5.5.4.5 KDD dataset . . . 80
5.5.4.6 MOA dataset . . . 81
5.5.4.7 UG_2C_2D dataset . . . 82
5.6 Conclusions . . . 84

6 AN ENSEMBLE OF AUTONOMOUS AUTO-ENCODERS FOR HUMAN ACTIVITY RECOGNITION . . . 87
6.1 Introduction . . . 88
6.2 Related Work . . . 89
6.3 Methodology . . . 91
6.3.1 Ensemble of kVN . . . 91
6.3.2 Ensemble Of Auto-Encoders . . . 92
6.4 Experiments . . . 94
6.4.1 Datasets . . . 94
6.4.2 Experimental setup . . . 95
6.4.3 Results and Discussion . . . 96
6.4.3.1 WISDM Dataset . . . 97
6.4.3.2 MHealth Dataset . . . 98
6.4.3.3 PAMAP2 dataset . . . 100
6.4.4 Accuracy per body location . . . 102
6.4.5 Aggregation of Classes . . . 103

6.5 Conclusion 104

7 A STUDY ON HYPERPARAMETER CONFIGURATION FOR HUMAN ACTIVITY RECOGNITION 107
7.1 Introduction 108
7.2 Related Work 109
7.3 Activity Recognition Overview 110
7.4 Experimental Results 111
7.4.1 The PAMAP2 Dataset 111
7.4.2 Experimental setup 112
7.4.3 HAR Accuracy Results 112
7.4.4 Execution Time and Energy Consumption 114
7.5 Conclusion 115

8 CONCLUSIONS AND FUTURE RESEARCH 117
8.1 Conclusions 118
8.2 Future Research 121

BIBLIOGRAPHY 123

CHAPTER

1

INTRODUCTION

It is a very sad thing that nowadays there is so little useless information.

Oscar Wilde

Recent technological advancements have led to significant changes in modern society. Nowadays, the world is surrounded by several digital applications, which are daily generating large volumes of data. These applications can be found in many different areas, such as healthcare, meteorological analysis, stock market analysis, network traffic monitoring, businesses, social networking, etc.

Many of these applications produce online data. In literature, this data is called *data stream*. A data stream is a sequence of instances continuously produced, usually at high speed. This data is potentially unbounded in size because its generation can occur without interruption. Additionally, the data generated may not be strictly stationary, meaning that its underlying probability distribution can change over time, sometimes presenting temporal correlation (AGGARWAL, 2007).

In the past few decades, extracting knowledge from data stream has been the core of much academic research and many business applications. For example, the knowledge from the data collected by smartphones, or wearable devices, can help healthcare professionals to monitor the daily routines of their patients (DOBBINS; RAWASSIZADEH; MOMENI, 2017). Another example is in businesses, where valuable knowledge can be used to predict users interests in advertisements, to recommend entertainment options and to make decisions regarding loan applications (FERREIRA *et al.*, 2019).

In this thesis, we focus on proposing novel incremental learning methods capable of learning from data stream. Each proposed algorithm was designed to achieve maximum predictive performance with minimum time and memory costs. This chapter is structured as follows: we briefly describe in Section 1.1 the main challenges involving data stream. In Section 1.2, we present our objectives and research questions. In Section 1.3, we present our main contributions. Finally, we present the thesis outline in Section 1.4.

1.1 Learning From Data Streams

The classical machine learning approaches are based on batch learning, usually coming from fixed-size datasets. In these approaches, ideally, the model is trained with instances that represent all classes that are part of the dataset application domain. After that, the model is tested on a new dataset. It is expected that the test dataset is from the same stationary probability distribution as the train dataset (GAMA, 2010).

Data streams impose a challenge to classical machine learning because they have characteristics that are limitations to these approaches. Since they are designed for static datasets, they are not capable of analysing continuous data, mainly due to memory constraints. Moreover, they do not allow for incremental learning, meaning that they are not able to detect and adapt to changes over time.

Data streams continuously generate new data and, because of their non-stationary nature, the underlying probability distribution of this new data can change over time. This means that algorithms that learn from data streams need to be able to adapt to a dynamic environment. Due to this and to memory constraints, learning from this type of data requires real-time processing (AGGARWAL, 2007). Depending on the changes in the probability distribution, three different phenomena can occur (GAMA *et al.*, 2014):

- *Concept drift* refers to changes in the statistical properties of a concept that was previously learned by a model;
- *Novel concepts* are patters that were not present during the training of a model, but appear in the stream;
- *Recurring concepts* are a special type of concept drift in which concepts forgotten by the model may reappear in the future.

Due to changes in the probability distribution, a learning algorithm needs to update its model with the incoming data. Otherwise, the model can become outdated and its predictive performance can decrease over time. Figure 1 illustrates what can happen to a model that does not update itself when the data distribution changes. In Figure 1a, the model correctly classifies the input data because the data is from the same probability distribution as the training set. In (b), the model misclassifies some of the data because of concept drifts in the data from the two classes (red diamond and blue circle). Finally, in (c), the model misclassified all data from the

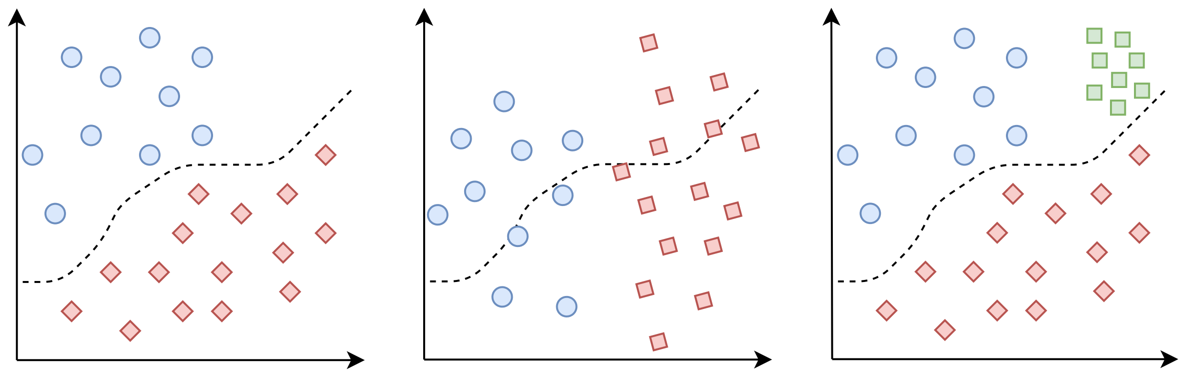

(a) Original data (b) Concept drift (c) Novel Concept

Figure 1 – Classification in different situations. Each geometric figure (circle, diamond and square) represents a concept and the dashed line represents the model.

new concept (green square) because the model only learned two concepts, the ones represented by the blue circle, and the red diamond. In this case, the data from the new concept is classified as the blue circle.

In order to act in data stream, one important property for a learning algorithm is to be incremental (GAMA, 2010). Incremental learning can, for example, deal with concept changes [1] by explicitly detecting changes in parts of the stream (BIFET *et al.*, 2013; BIFET; GAVALDÀ, 2007; LOSING; HAMMER; WERSING, 2016). For that, the model needs to assess whether the data from different periods of time follows the same probability distribution. Usually, the data from past concepts is compared with the data from the current stream.

Many of the machine learning approaches proposed for data stream are based on supervised learning. Most of them deal with concept changes by continuously calculating the predictive performance of the classification model. In order to do so, the accuracy of e.g. a classification model is monitored over time (GAMA *et al.*, 2004; ZLIOBAITE; KUNCHEVA, 2009; MASUD *et al.*, 2013). A concept change is detected when the accuracy falls below a given threshold. The essential assumption here is that label of this incoming data is available.

There are two main problems of assuming that the arriving data is labelled. First, the process of labelling usually has a cost, which increases with the complexity and the need of domain expertise. Second, if the data arrives at a high speed, there will not be enough time to label the data. Hence, for many applications, we can assume that the data arrives unlabelled.

Due to the lack of labelled data, the update of the model can rely only on the predictive attribute values. In this situation, to detect concept changes and update the model, it is possible to use clustering algorithms. These algorithms can extract patterns, clusters, from the current stream and compare them with previous patterns from other periods of the stream. The clusters can be used to summarise the relevant data by letting a set of clusters represent a concept.

[1] novelties and concept drift

Furthermore, they can be updated to incorporate concept changes and to detect changes in the stream (AGGARWAL, 2007).

1.2 Research Objective and Research Questions

In this thesis, we aim to design new unsupervised and incremental machine learning algorithms for learning concept changes in data streams. Our focus is on algorithms able to automatically choose the moment to update their models. In that sense, the update of a model should be done with little or no external feedback. To achieve this goal, we address the following objectives:

- To develop unsupervised learning algorithms that can detect and learn concept changes in data streams;
- To develop algorithms for multi-class problems; thus, models that can detect more than one concept change at the same time;
- To develop algorithms that can differentiate novelties from concept drift;
- To evaluate the algorithms' predictive performance, considering their recall over time.

For that, this research will be based on the following specific research questions:

- *RQ1: How to reduce the amount of data used to train a model and how to select the most representative data to update a model in data streams?*
- *RQ2: How to incrementally learn concept changes in data streams, considering an unsupervised approach, without storing data for future analysis?*
- *RQ3: How to combine clustering partitions from different clustering techniques and use them as a classification model in data streams?*
- *RQ4: In which data streams can an ensemble model of clusters from different clustering approaches achieve higher predictive performance than an ensemble model of clusters from the same clustering approach?*
- *RQ5: How to use a set of auto-encoders for classification of data streams?*

The answers to these research questions will enable us to develop algorithms that satisfy one or more of the objectives. Therefore, we address the research questions in the following chapters. In Chapter 3, *RQ1* is addressed in the context of concept drift. The *RQ2* is addressed in Chapter 4. In Chapter 5, we answer the questions *RQ3* and *RQ4*. We address *RQ5* in Chapter 6 in the context of human activity recognition, a real-world data stream application.

1.3 Contributions of this thesis

In this section, we give an overview of the contributions of this thesis and its motivations.

Data streams pose several challenges for machine learning applications. Among these challenges, this thesis focuses on proposing solutions to the topics of concept drift and novelty detection in data streams. We consider data streams with little labelled data, for these we proposed semi-supervise approaches. We also consider data streams without labelled data, where we proposed unsupervised approaches. The thesis conceptually consists of five parts.

The first part address to the problem of concept drift in data streams with little labelled data. In data streams applications, the classification algorithm *k*-nearest neighbours (*k*NN) is often implemented with a sliding window, also called temporary memory, that contains a certain amount of data, which is used as its training data. This training data is called prototypes and, ideally, it should be representative of the concepts presented in the data stream. In this first chapter (Chapter 3), we investigate how to select prototypes to incrementally update a model based on *k*NN. As a result of the investigation, we propose a method called Sliding Window Clusters (SWC). This method stores into a sliding window a set of clusters, summarising the concepts, and the representative data, according to a statistical test.

In the second part, we study unsupervised solutions to learn concept drift and novelty detection in data streams with unlabelled data. In Chapter 4, we propose a new method based on the *k*NN classifier that incrementally detects and learns the concept changes. This method, called *Higia*, uses micro-clusters as prototypes to model the current concepts in a stream. Each micro-cluster has a centroid, a radius and a threshold. These properties are used to define if the incoming data will be classified as a normal class, concept drift or novelty. Each micro-cluster can be incrementally updated when a new instance is close to its centre. Furthermore, new micro-clusters can be incorporated into the model when novelties are detected in the stream.

The two methods proposed in Chapter 3 and Chapter 4 are both based on a single model, which contains prototypes extracted by a single clustering partition. In the third part (Chapter 5), we study how to combine clustering partitions to build an ensemble model in data streams, since we assume that an ensemble can have a higher predictive performance than a single model. We propose a method based on an ensemble obtained by the combination of clustering partitions from one clustering algorithm, which we refer to as *HoCluS*. We also propose another method based on an ensemble of different clustering algorithms, referred to as *HeCluS*. Finally, in this chapter, we also compare the predictive performance of these methods considering different data streams. Both methods allow for the use of clustering techniques with different bias, in order to obtain more robust classification models.

In the fourth part of this thesis (Chapter 6), we propose a new method for Human Activity Recognition, a real-world data stream application. We propose an ensemble model to classify human physical activities based on auto-encoders, called Ensemble of Auto-Encoders (EAE).

In EAE, each auto-encoder is trained with data from one class. Thus, in the context of human activity recognition, each auto-encoder is associated with a label/activity. As new data arrives for classification, the reconstruction loss is calculated for each auto-encoder. The data is then classified with the label from the auto-encoder with the lowest reconstruction loss.

In Chapter 7, we investigate the impact of varying the hyperparameters associated with most methods proposed for human activity recognition applications. In this study, we also analyse how data from different users can impact on the accuracy of a predictive model. We measure the energy and time consumption to process and to classify new data. We conduct the experiments on a hardware system running an Android mobile operating system.

1.4 Thesis outline

This thesis is presented as a series of papers in the form of self-contained chapters. These are papers that have been published and peer-reviewed. Each chapter represents the progress of this research and the solutions proposed to the problems identified by this research. One can notice that there are similarities between the chapters, mainly because of the literature review. More concretely, the rest of this thesis is based on the following papers:

Chapter 3, *A Cluster-Based Prototype Reduction for Online Classification* (GARCIA; CARVALHO; MENDES-MOREIRA, 2018). This paper presents a semi-supervised method, called SWC, that incrementally updates a classification model when concept drift is detected. This paper was published in the proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, 2018.

In Chapter 4, *Online Clustering for Novelty Detection and Concept Drift in Data Streams* (GARCIA *et al.*, 2019c), presents an unsupervised approach for concept changes in data streams. This paper was published in the proceedings of the EPIA 2019 conference.

In Chapter 5, *An Ensemble of Unsupervised Approaches for Novelty Detection in Data Streams*, we propose two ensembles of clustering for data streams. This paper, which has been submitted to the Machine Learning Journal, is an extension of a previous work. The paper (GARCIA *et al.*, 2019a) was published in the proceedings of the Discovery Science 2019 conference.

Chapter 6, *An Ensemble of Autonomous Auto-Encoders for Human Activity Recognition* (GARCIA *et al.*, 2021), proposes an ensemble of auto-encoders for human activity recognition. This chapter was published in the Neurocomputing Journal 2021.

Chapter 7, *A Study on Hyperparameter Configuration for Human Activity Recognition* (GARCIA *et al.*, 2019b). This paper presents an empirical study on how the hyperparameters of an algorithm for human activity recognition can affect the model performance in terms of accuracy and computer resources. This chapter was published in the proceedings of the International Conference on Soft Computing Models in Industrial and Environmental Applications, 2019.

Finally, Chapter 8, gives an overview of the main contributions and findings in this PhD thesis.

CHAPTER 2

BACKGROUND

How hard it must be to live only with what one knows and what one remembers, cut off from what one hopes for!

Albert Camus, The Plague

Data stream is related to data constantly generated at a high rate and in a non-stationary way. Learning from data stream requires machine learning algorithms capable of dealing with large volumes of data, real-time processing, and online learning.

In this chapter, we formally describe data stream and its constraints to classical machine learning algorithms. We also discuss how to summarise data for future analysis. Additionally, we define concept drift and novelty detection. Finally, we present human activity recognition as an example of a real-world application of data stream.

2.1 Data Stream

In machine learning literature, a data stream D is represented as a potentially infinite sequence of data. This data is composed of instances, each one arriving at a timestamp t. Thus, each instance at time t is denoted by X_t. In supervised problems, X_t can be associated to a target class, y_t. According to this description, a data stream can be represented as (FARIA; GAMA; CARVALHO, 2013):

$$D_t = \{(X_1, y_1), (X_2, y_2), \ldots, (X_t, y_t)\}.$$

Data streams has a non-stationary nature; therefore, the probability distribution that generates the data can change over time (AGGARWAL, 2007). Depending on these changes, different phenomena can occur. In literature, these phenomena are named as concept drift, recurrent concept or novelty/new concepts. In these situations, a model must update itself; otherwise, its predictive performance can decrease over time. Thus, a learning algorithm has to take into account the following characteristics of data streams:

- It is a sequence of instances arriving online, usually at a high rate;
- It is potentially unbound in size;
- It is not possible to store all data into the main memory for future analysis;
- The probability distribution that generates the data is possible non-stationary.

In data streams, due to memory constraints, it is not possible to store all data into the main memory. However, there are some strategies to store parts of it. One of these strategies is the use of summarising approaches (AGGARWAL, 2007; GAMA, 2010; ZHANG; RAMAKRISHNAN; LIVNY, 1996). The idea is to maintain only the statistical information of past concepts and update it with the new data. The summary information of the stream should preserve the meaning of the original data, by representing its concepts, and allow for the efficient analysis of less data (AGGARWAL *et al.*, 2003).

2.2 Statistical Summary of Data Streams

Considering that the data stream is unbounded in size, it is not possible to store all data in the main memory for consultation. However, a compact representation of the data can be used for storing statistic summaries of the data stream (SILVA *et al.*, 2013).

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) (ZHANG; RAMAKRISHNAN; LIVNY, 1996) is a clustering algorithm that uses feature vectors for summarising data. Each vector Cluster Feature Vector (CF-Vector), $CF = (N, \vec{LS}, \vec{SS})$, is a condense representation of a cluster from the dataset. In that sense, a cluster is a group of similar data, according to a similarity measure like Euclidean distance (PANG-NING *et al.*, 2006). The CF-Vector has three components:

- N: the number of instances;
- $\vec{LS}$: the linear sum of the N instances, i.e., $\sum_{i=1}^{N} \vec{X_i}$;
- $\vec{SS}$: the sum of squared the N instances, i.e., $\sum_{i=1}^{N} \vec{X_i}^2$.

From these three components is possible to compute other statistical measures, such as: mean, standard deviation and correlation of features (GAMA, 2010). Besides, the CF-Vector has incremental and additive properties that allow for the online update of the clusters. This properties are as follows:

- Additivity: it is possible to join two or more *CF-Vector* with the Theorem of Additivity (ZHANG; RAMAKRISHNAN; LIVNY, 1996). As an example, considering the two CF-Vectors CF_1 e CF_2:

$$CF_1 + CF_2 = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, \vec{SS}_1 + \vec{SS}_2).$$

  The reverse operation is also valid, which means that it is possible to separate clusters.

- Incrementality: The Theorem of Additivity can also explain the property of incrementality. A new instance, X_t, can be inserted into a CF-Vector by updating its statistics properties as follows:

$$\vec{LS} \leftarrow \vec{LS} + X^N \tag{2.1}$$

$$\vec{SS} \leftarrow \vec{SS} + (X^N)^2 \tag{2.2}$$

$$N = N + 1 \tag{2.3}$$

The CF-Vector is a data structure that can be adapted to summarise data stream. It is efficient because it stores less data and contains information that is accurate enough to calculate the statistical measures used by learning algorithms. Note that the original CF-Vector was not originally designed for data streams, however many adaptations of it were proposed, such us: CluStream (AGGARWAL *et al.*, 2003), DenStream (CAO *et al.*, 2006), ClusTree (KRANEN *et al.*, 2011).

In this thesis, we use the CF-Vector properties in Chapter 3, Chapter 4 and Chapter 5. Thus, a CF-Vector is used to represent a concept, C_t, in a timestamp t. Each concept contains:

- a centroid c_t: a vector that contains the average of all data inside the cluster;
- a radius r_t: the distance between the centroid and the farthest instance from the cluster (MASUD *et al.*, 2010);
- a threshold T: a constant value multiplied by the radius of the concept.

2.3 Concept Drift

In data streams, the data distribution can unexpectedly change over time. In that sense, *concept drift* can be considered as the natural tendency of a data stream to evolve over time (AGGARWAL, 2007). In which, essentially, can be a change in the stochastic process that generates the data. These changes can occur, for example: due to changes in personal interesting about an online news (ŽLIOBAITĖ; PECHENIZKIY; GAMA, 2016); a medicine impacting on a patient blood pressure (MELLO *et al.*, 2019); changes in the patterns of physical activities on an elderly indicating immediate emergencies, such as falls (ZDRAVEVSKI *et al.*, 2017).

There are many definitions of concept drift in literature (GAMA *et al.*, 2014; ŽLIOBAITĖ; PECHENIZKIY; GAMA, 2016; AGGARWAL, 2007; MORENO-TORRES *et al.*, 2012). In this thesis, we follow the notation used in (GAMA *et al.*, 2014). In that sense, considering a distribution P, in a given time t, on the instance X and label y. The concept drift happens when P suffers changes affecting the conditional probability, $X : P_t(X,y) \neq P_{t+1}(X,y)$. As a result, a model built during time t could be outdated in time $t+1$.

The term concept drift is used as a generic term to describe many different changes. To simplify, two types of drifts can be distinguished, depending on the nature of the problem, as follows (GAMA *et al.*, 2014):

- The posterior probability $P(y|X)$ may change over time. These changes are independent of changes in $P(X)$;
- The distribution of $P(X)$ changes without affecting $P(y|X)$.

In this thesis, we focus our research on the drifts that change the data distribution without knowing the true labels; therefore, $P(X)$ changes. We are interested in problems related to concept drift in situations where labels are never available (unsupervised learning); or where some percentage of labels is available (semi-supervised learning).

2.4 Novelty Detection

In data streams, not only old concepts can change over time, but also new concepts can appear. The new concepts are patterns that were not present during the training phase of a model but appear later in the stream (FARIA; GAMA; CARVALHO, 2013).

There are several real-world applications in which new patterns can appear online: to perform intrusion detection in network systems (SPINOSA; CARVALHO; GAMA, 2008), to detect fraudulent credit (CHANDOLA; BANERJEE; KUMAR, 2009), to detect new physical activities performed by a person (GUO *et al.*, 2016). In healthcare applications, for example, monitoring sensing data from patients can help medical staff to receive a warning message immediately after an event happens, such as falling. That way, sick people can receive medical attention as soon as needed.

Novelty detection refers to the ability of models to learn new concepts (GAMA, 2012). In the presence of new concepts, a model needs to extend its representation by learning the new concepts. In that sense, this is different from batch methods because the data from the stream does not match its current model. Thus, for machine learning models, the challenge of detecting the new concepts relies on defining what is *abnormal* and what is *normal* to its model (GAMA, 2012).

The *abnormal* classes can also be named as *not normal* (SPINOSA; CARVALHO; GAMA, 2009), *anomaly* (MASUD *et al.*, 2013) or *novel/new* (FARIA; GAMA; CARVALHO, 2013) classes. We followed the notation from (FARIA; GAMA; CARVALHO, 2013). In the latter, normal concepts are a set of classes used to train the classification model and novelty concepts are the *new classes* that emerge over time (FARIA; GAMA; CARVALHO, 2013). This approach can be consider as a binary classification task, composed by *normal* and *abnormal* classes (SPINOSA; CARVALHO; GAMA, 2009). However, it can also be consider as a multi-class classification task (FARIA; GAMA; CARVALHO, 2013), in which more than one new class can emerge in the stream.

Several methods for novelty detection in data streams are divided in two phases: an *offline* and an *online* phase (SPINOSA; CARVALHO; GAMA, 2008; FARIA; GAMA; CARVALHO, 2013; MASUD *et al.*, 2010; ABDALLAH *et al.*, 2016; MELLO *et al.*, 2019). In the offline phase, a model (or an ensemble of models) is trained with a static and labelled dataset. Considering, for example, that this dataset has m classes, then $Y^{Nor} = \{y_1, y_2, ..., y_m\}$ represents the set of *normal classes*. The initial model trained with this dataset is used to classify the new data.

In the online phase, the new data arrives for classification as a stream of data. The model classifies the new data as *normal* or as *unknown*. The *unknown* data corresponds to patterns that significantly differentiate from the normal classes. Later, this *unknown* data can be identified as *outlier*, *concept drift* or *novelty*.

Novelty and outlier are correlated terms. Both are related to patterns that are different from the normal patters (FARIA *et al.*, 2016). However, we assume that the outliers are undesired patterns in non-dense areas, which means that they cannot form clusters. The outliers are candidates to aberrant data that could be resulted of human error, noise in the sensor reading or malicious activities (GOGOI *et al.*, 2011). They are not interesting to learn because there is no guarantee that they represent concepts.

On the other hand, a novelty is a group of similar data found in a dense area. Thus, the novelties are part of the natural evolution of the stream, and the model must learn them. Moreover, a dense group of data should be required as evidence of the appearance of a novel concept (GAMA, 2010).

When a novel class (or concept) with label y_{m+1} emerges, a novelty detection algorithm must be able to detect it and update the model. One strategy of novelty detection is storing the unknown data into a buffer, temporary memory. Considering that the buffer contains the potential novelties and concept drift, the buffer should be analysed periodically. This analysis can be done by clustering this data and compare the clusters found with the normal concepts.

In that sense, each new cluster should be labelled as a *concept drift* if the similarity between the new cluster and its most similar *normal* cluster is inside of a given boundary; or a *novelty* if the similarity between the cluster and all normal concepts is outside of each normal concept boundary. In this thesis, we define the boundary of a concept, called here as the threshold, as a constant value multiplied by the radius of the concept.

2.5 Human Activity Recognition

Data streams can be considered as stochastic processes, in which the instances are independent from each other (GAMA, 2010). However, when the data is from recording devices, this data likely has temporal dependence (ŽLIOBAITĖ *et al.*, 2015). This means that the data stream is consistent with consecutive instances that exhibit temporal dependence, also known as temporal correlation, between each other.

An example of an application of data streams with temporal dependence is in Human Activity Recognition. Human Activity Recognition is a research field focused on the use of sensing technology to classify human physical activities and to infer on human behaviour (DOBBINS; RAWASSIZADEH; MOMENI, 2017). The data collected from sensing devices is a potential infinity sequence of data that usually arrives at a high rate. This data can be unbounded in size if we consider that the data is continuously collected. The sensing data is collected from devices such as accelerometers, gyroscopes, and magnetometers; located on one or more body positions of an individual. Furthermore, the dataset contains the sensing data from a group of people performing a set of physical activities.

Most machine learning approaches for human activity recognition are based on a model trained with a dataset containing sensing data (LARA; LABRADOR, 2013; MANNINI *et al.*, 2013; GUO *et al.*, 2016). It is unlikely that this model will have the same predictive performance for all types of people. Due to many factors like age, physical conditions and health; each individual might perform the same activity differently (KRISHNAN; COOK, 2014). Besides, it is expected a natural change in the way an individual performs a physical activity. For example, the way a person runs can change as they get at older ages. Moreover, the preference for physical activities can be different from person to person. Thus, the model needs: to adapt to different individuals; to adapt itself over time; to learn new activities when they are detected.

In this thesis, we address the problem of human activity recognition in Chapter 6 and Chapter 7.

CHAPTER

3

A CLUSTER-BASED PROTOTYPE REDUCTION FOR ONLINE CLASSIFICATION

Collaborating authors

André C. P. L. F. de Carvalho
University of São Paulo, São Carlos, Brazil

João Mendes-Moreira
LIAAD-INESC TEC, University of Porto, Porto, Portugal

Abstract

Data stream is a challenging research topic in which data can continuously arrive with a probability distribution that may change over time. Depending on the changes in the data distribution, different phenomena can occur, for example, a concept drift. A concept drift occurs when the concepts associated with a dataset change when new data arrive. This paper proposes a new method based on *k*-Nearest Neighbors that implements a sliding window requiring less instances stored for training than existing methods. For such, a clustering approach is used to summarise data by placing labelled instances considered similar in the same cluster. Besides, instances close to the uncertainty border of existing classes are also stored, in a sliding window, to adapt the model to concept drift. The proposed method is experimentally compared with state-of-the-art classifiers from the data stream literature, regarding accuracy and processing time. According to the experimental results, the proposed method has better accuracy and less time consumption when less information about the concepts are stored in a single sliding window.

3.1 Introduction

In real-world data analysis, data can continuously arrive in streams, with a probability distribution that can change over time. These data are known as data streams. Depending on the changes in the data distribution, different phenomena can occur, like concept drift (GAMA *et al.*, 2014). In these situations, it is essential to adapt the classification model to the current stream; otherwise, its predictive performance can decrease over time.

Several algorithms proposed for data stream mining are based on online learning (FARIA; GAMA; CARVALHO, 2013; BIFET *et al.*, 2013; GAMA *et al.*, 2014; LOSING; HAMMER; WERSING, 2016). Some of them are based on the *k*NN algorithm. In the data stream mining, the *k*NN algorithm maintains a sliding window with a certain amount of labelled data, which is used as its training data.

Other algorithms from the literature deal with concept drift by explicitly detecting changes in parts of the stream, comparing the current concept with previous concepts from time to time (LOSING; HAMMER; WERSING, 2016). Some of them continuously calculate the model classification error. For such, they assume that the label of the data arriving in the stream is available.

However, there is a cost associated with the data labelling process that can become prohibitive or unfeasible when data arrive in high speed or volume. In online classification, the labelling of incoming instances can have a high cost (ZLIOBAITE *et al.*, 2014). The lack of the label makes the measure of classification error a problematic task.

Despite its simplicity, *k*NN has been largely used in literature, because it is nonparametric, favouring its use in scenarios with few available information and with known concepts changing over time (BIFET *et al.*, 2013). However, the use of a sliding window may ignore instances with relevant information about persistent concepts. Furthermore, the size of a sliding window affects its efficient use.

This article proposes Sliding Window Clusters (SWC), a method based on *k*NN that implements a sliding window whose number of instances stored can be reduced. SWC summarises data streams by creating a set of clusters, each one representing similar labelled instances. Instances close to decision border of each cluster are also stored, so they can be used to adapt the model to concept drift.

An experimental evaluation shows that SWC can increase the predictive performance and reduce both computational and time consumption than related methods based on *k*NN and sliding window.

This paper is structured as follows. Section 3.2, presents previous related works using *k*NN and sliding window. Data stream and concept drift are introduced in 3.3. The proposed method is described in Section 3.4. Sections 3.5 presents the experimental setup and analyses the results obtained. Finally, Section 3.6 has main conclusions and points out future work directions.

3.2 Related Work

This section briefly presents previous works using kNN for data stream classification with concept drift. These works use variations of sliding window to store the training instances.

One alternative of online learning is to randomly select instances to maintain or discard in the sliding window. This is the case of the method Probabilistic Approximate Window (PAW) method (BIFET *et al.*, 2013), a probabilistic measure used to decide which instance will be discarded from the sliding window when a new instance arrives. Thus, the size of the window is variant and represents a mix of outdated and recent relevant instances. The $k\text{NN}_W$ method combines the PAW method couplet with the kNN classifier.

Another related method, ADaptive sliding WINdowing (ADWIN) (BIFET; GAVALDÀ, 2007), is a concept drift tracker able to monitor changes in data streams. The algorithm automatically grows the sliding window when no change is detected in the stream. When a change is detected, the algorithm shrinks the sliding window and forgets the sub-window that is outdated. In the combination of kNN with PAW and ADWIN (BIFET *et al.*, 2013), $k\text{NN}_{WA}$, ADWIN is used to keep only the data related to the most recent concept from the stream, the rest of the instances are discarded.

A deficiency of updating instances using a sliding window is the possibility to forget old but relevant information. To avoid losing relevant information, another method named Self Adjusting Memory (SAM) (LOSING; HAMMER; WERSING, 2016), to adapt a model to concept drift by explicitly separating current and past information. SAM uses two memory structures to store information, one based on short-term-memory and the other on long-term-memory. The short-term-memory contains data associated with the current concept, and the long-term-memory maintains knowledge (old models) from past concepts. SAM is couplet with a kNN classifier.

The implementations and variations of kNN for data stream mining are available in the MOA framework . Due to memory and computational limitations, the implementations use a fixed-size window of 1000 labelled instances.

3.3 Problem Formalisation

A possible unbounded amount of data can sequentially arrive in a data stream. These data can often change their distribution over time, which may require the adaptation of the current model context (GAMA *et al.*, 2014).

Formally, a data stream is a sequence of instances, potentially infinity that can be represented by (FARIA; GAMA; CARVALHO, 2013):

$$D_{tr} = \{(X_1, y_1), (X_2, y_2), ..., (X_{tr}, y_{tr})\}$$

where X_{tr} is an instance arriving in time tr and y_{tr} is the target class. Each instance needs to be processed only once due to finite resources.

Concept drift is a change in the distribution probability of target classes (GAMA *et al.*, 2014). Formally, a distribution P in a given time tr conditioned by the instance X and label y can suffer changes affecting the conditional probability $P_{tr+1}(X, y)$. As a result, a model built during time tr could be outdated in time $tr + 1$.

$$X : P_{tr}(X, y) \neq P_{tr+1}(X, y)$$

3.4 Methodology

In data stream mining, an ideal classifier should be able to learn the current concept in feasible time without forgetting relevant past information (BIFET *et al.*, 2013).

The proposed method is described in Algorithm 1. Instead of storing all instances that fit in a sliding window (for representing both old and current concepts), SWC stores compressed information about concepts and instances close to uncertainty border of each class. As the previous methods, SWC is combined with the *k*NN classifier in the MOA framework (BIFET *et al.*, 2010a).

Algorithm 1 – SWC: Online Window Update

```
1: input: $X_{tr}$, $W$, $T$, $\rho$
2: output: $W$
3: $rand \leftarrow random(0, 1)$
4: if $rand \leq \rho$ then
5:     for all w in W do
6:         Let $X_{tr}$ be the nearest to $w$ $(w \in W)$
7:         $dist \leftarrow EuclidianDistance(X_{tr}, w)$
8:         if $dist < w_{radius}$ then
9:             $W \leftarrow UpdateCluster(X_{tr}, w)$
10:        else
11:            if $dist \leq T$ then
12:                $W \leftarrow W \cup X_{tr}$
13:            end if
14:        end if
15:    end for
16: end ifreturn $W$
```

A more detailed description of how SWC works is presented next. Initially, all instances arriving from the stream are stored in the form of clusters. The clusters are created using the

CluStream algorithm (AGGARWAL *et al.*, 2003). A constrain implying that each cluster must contain only instances from the same class was included.

As data arrives, a parameter based on probability, ρ, is used to decide if a new instance, X_{tr}, will be incorporated to the model W. If X_{tr} is inside a radius from an existing cluster, the instance is incorporated to this cluster. However, if X_{tr} inside the uncertainty border, X_{tr} is incorporated to the model alone, outside the existing clusters, for such, the uncertainty border is defined as the area outside the radius of a cluster, but inside a given threshold.

As is illustrated in Figure 2, if the instance, X_1, is inside the radius of the closest cluster, then it will be incorporated to the existing cluster, however, if the instance, X_2, is closer to an uncertainty border, it is stored alone.

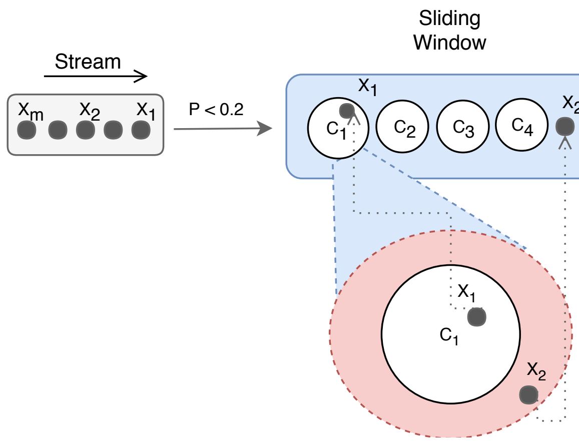


Figure 2 – Instances X_1 and X_2 are stored within the sliding window. The first instance, X_1, is closer to cluster C_1 and inside its radius. The second instance, X_2, is outside cluster area, but close to the uncertainty border.

It must be observed that not all instances in the stream are included into the sliding window. For each instance, arriving in the stream, SWC randomly decides if the instance will be learned or not. A similar procedure is used in (BIFET *et al.*, 2013), which uses a probability $\rho = 0.5$. SWC uses a lower probability, consider that the learning process can be done with a lower probability of $\rho = 0.2$, without significant predictive performance loss, but with a lower processing cost.

3.5 Experimental Evaluation

This section experimentally compares SWC with other methods implemented in the MOA framework that use kNN with sliding window, namely kNN, $k\text{NN}_W$, $k\text{NN}_{WA}$ and SAM. The experimental evaluated used was Interleaved Test-Train to incremental learning (BIFET *et al.*, 2013).

Table 1 – Characteristics of datasets evaluated.

Datasets	Samples	Features	Class
SEA	5.000 / 50.000 (total)	3	2
Mixed Drift	60.000 / 600.000 (total)	2	15
Rotating Hyperplane	20.000 / 200.000 (total)	10	2
Forest Cover Type	58.101 / 581.012 (total)	54	7
Airlines	53.938 / 539.383 (total)	4	2
Moving RBF	20.000 / 200.000 (total)	10	5

3.5.1 Datasets

Table 1 describes the datasets used in the experiments. Before the streaming, in an offline phase, all methods started with a batch of labelled data representing 10% of each dataset. The remaining data arrived in the stream. Real and artificial datasets were used.

Artificial Datasets

The SEA Concepts Dataset (STREET; KIM, 2001) has four concepts. A concept drift occurs at each 15.000 instances, with different thresholds for the concept.

The Rotating Hyperplane dataset is based on a hyperplane of d-dimensional space which is continuously changing in position and orientation. It is available in the MOA framework and was used in (BIFET *et al.*, 2013; LOSING; HAMMER; WERSING, 2016).

Moving RBF is a dataset, generated by MOA framework, based on Gaussian distributions with random initial positions, weights and standard deviations. Over time, these Gaussian distributions suffer changes. This dataset is used by (BIFET *et al.*, 2013; LOSING; HAMMER; WERSING, 2016).

Mixed Drift (BIFET *et al.*, 2013; LOSING; HAMMER; WERSING, 2016) is a mix of tree datasets: Interchanging RBF, Moving Squares and Transient Chessboard. Data from each dataset are alternatively presented in the stream.

Real World Datasets

The Forest Cover Type (DHEERU; TANISKIDOU, 2017) data set is a well-known benchmark for the evaluation of algorithms for data stream mining, being used continuously to validate proposed methods (LOSING; HAMMER; WERSING, 2016; BIFET *et al.*, 2013; ZLIOBAITE *et al.*, 2014).

The Airlines dataset has data from US flight control (ZLIOBAITE *et al.*, 2014). It has two classes, one indicating that a flight will be delayed, and the other that the flight will arrive on time.

3.5.2 Results and discussion

The proposed method, SWC, is compared with the methods kNN, $k\mathrm{NN}_W$, $k\mathrm{NN}_{WA}$ and SAM. For all methods, one nearest neighbour ($k = 1$) is adopted. The remaining parameters use default values, including a fixed window size ($w = 1000$).

The ρ parameter, chance of updating the model, in the SWC method is defined for an acceptable trade-off between accuracy and time cost. A parameter of threshold $T = 1.1$, uncertainty border, is also defined for each cluster. The threshold is multiplied by the radius of each cluster and indicates how much the cluster can expand. Both parameters were explained in Section 3.4.

Experiments were performed to decide the value of ρ and for SWC. Figure 3 shows that there is an increase of accuracy with $\rho = 0.5$, meaning that an instance has 50% of chance to be learned by the model. However, the selected value was $\rho = 0.2$, which results in a better balance between accuracy and time cost.

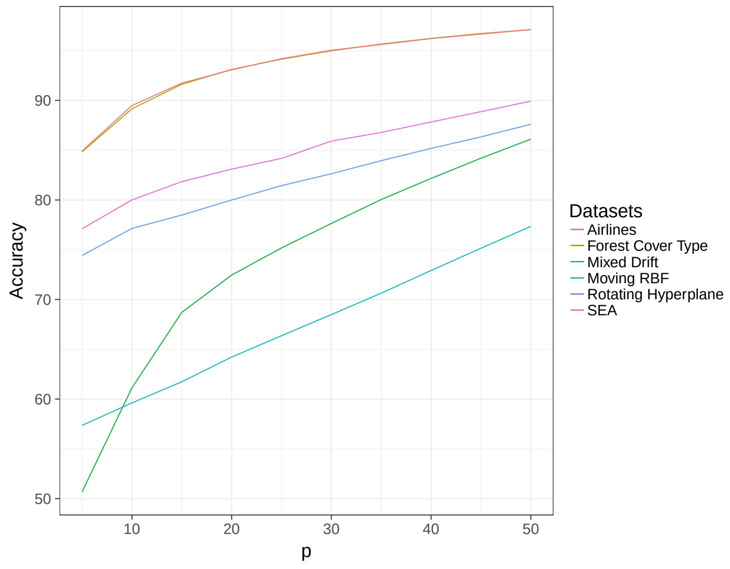


Figure 3 – SWC accuracy performance of all datasets varying ρ value in 5% to 50%.

Table 2 shows the average accuracy and total time cost. It must be observed that accuracy is the measure of instances correctly classified over test/train interleaved evaluation (BIFET *et al.*, 2013).

The results show that SWC is competitive with state-of-the-art SAM and is considerably faster. The method baseline kNN presented the worst performance, which was expected, once it does not learn over time. However, it is a good baseline to measure how much time each other method take to learn new instances.

Both methods $k\mathrm{NN}_W$ and $k\mathrm{NN}_{WA}$ present similar accuracy rates. However, $k\mathrm{NN}_{WA}$ has a higher cost due to the use of ADWIN.

Table 2 – Accuracy and time cost (in seconds) for each method.

Dataset	kNN	$k\text{NN}_W$	$k\text{NN}_{WA}$	SAM	SWC
SEA	77.24 (6)	77.25 (9)	77.25 (10)	80.53 (18)	83.49 (8)
Mixed Drift	16.82 (68)	53.62 (94)	53.62 (102)	80.53 (2724)	72.46 (73)
Rotating Hyperplane	50.00 (63)	66.42 (88)	68.42 (91)	70.27 (318)	80.17 (70)
Forest Cover Type	23.46 (614)	54.56 (898)	55.56 (1024)	89.84 (3422)	93.12 (394)
Airlines	54.37 (91)	52.53 (163)	52.53 (146)	88.37 (1530)	93.07 (120)
Moving RBF	26.07 (62)	59.98 (89)	59.97 (100)	69.92 (1788)	64.22 (71)

Table 3 – P-values obtained for the multiple comparison post-hoc Nemenyi test.

	kNN	$k\text{NN}_W$	$k\text{NN}_{WA}$	SAM
$k\text{NN}_W$	0.8536	-	-	-
$k\text{NN}_{WA}$	0.7591	0.9998	-	-
SAM	0.0090	0.1506	0.2201	-
SWC	0.0024	0.0621	0.0987	0.9962

Finally, although SAM and SWC obtained predictive accuracy similar to SWC, for some cases, SWC was better. Besides, SWC is faster due to the use of only one sliding window with compressed concepts and relevant instances.

To assess their statistical significance, a Friedman rank sum test combined with Nemenyi post-hoc test (DEMSAR, 2006), both with a significance level of 5%, was applied to the experimental results. A $p-value = 0.000441$ was obtained in the Friedman test, showing a significant difference between the five methods. Additionally, the Nemenyi post-hoc test, Table 3, showed meaningful statistical differences between the following pair of methods: SWC$\succ k$NN. There is no significant difference between all remaining pairs. However we emphasise that SWC$\succ k\text{NN}_W$ and SWC$\succ k\text{NN}_{WA}$ have relatively low p-values (less than 10%).

3.6 Conclusion and Future Work

This paper presented a new method, SWC, based on k-Nearest Neighbors that implements a sliding window that stores less training instances than related methods. SWC stores in a sliding window clusters and instances close to uncertainty border of each class. The clusters are compressed stable concepts, and the instances are possible drifts of these concepts.

Considering accuracy performance, time and storage cost, SWC was experimentally compared with state-of-the-art related methods. According to the experimental results, SWC presented higher predictive performance, with lower processing and memory cost than the compared methods.

As future work, the authors want to distinguish concept drift from novelty detection and study an efficient alternative to discard outdated information. Besides, they intend to include an unsupervised concept drift tracker.

CHAPTER

4

ONLINE CLUSTERING FOR NOVELTY DETECTION AND CONCEPT DRIFT IN DATA STREAMS

Collaborating authors

Mannes Poel

University of Twente, Enschede, The Netherlands

Joost N. Kok

University of Twente, Enschede, The Netherlands

André C. P. L. F. de Carvalho

University of São Paulo, São Carlos, Brazil

Abstract

Data streams are related to large amounts of data that can continuously arrive with a probability distribution that may change over time. Depending on the changes, different phenomena can occur, like new classes can appear, or concept drift can occur in existing classes. New classes are patterns that are not seen during the training of the current classification model but appear after some time. Concept drift occurs when the concepts associated with a dataset change as new data arrive. This paper proposes a new algorithm based on *k*NN that uses micro-clusters as prototypes and incrementally updates the micro-clusters or creates new micro-clusters when novelties are detected. The proposed algorithm is experimentally compared with a state-of-the-art classifier from the data stream literature and one baseline. According to the experimental results, the proposed algorithm increases the predictive performance over time by incrementally

learning changes in the data distribution.

4.1 Introduction

Data streams are known as data that can continuously arrive in streams, with a probability distribution that can change over time (GAMA, 2010). As new data arrives, models previously induced can become outdated (SILVA *et al.*, 2013). In addition, due to the great amount of data generated, it is not feasible to store all incoming data in the main memory, requiring the removal of previous outdated data and online processing of incoming data (FARIA; GAMA; CARVALHO, 2013; GARCIA; CARVALHO; MENDES-MOREIRA, 2018).

Depending on the changes in the data distribution, different phenomena can occur, like concept drifts (GAMA, 2010; ZLIOBAITE *et al.*, 2014) and novelties (FARIA; GAMA; CARVALHO, 2013; MARKOU; SINGH, 2003). Concept drift refers to changes in the concept definitions of a normal class (GAMA *et al.*, 2014). Novelties concepts are patterns that are not present during the training of the classification model but appear later on in the data stream (FARIA; GAMA; CARVALHO, 2013). In these situations, it is important to adapt the classification model to the current data distribution; otherwise, its predictive performance can decrease along the time.

In this work, normal concepts are a set of *normal classes* used to train the classification model, and novelties concepts are the *new classes* that emerge in a data stream along the time (FARIA; GAMA; CARVALHO, 2013).

Novelty detection is a Machine Learning (ML) task based on the identification of novelties in the data (DING *et al.*, 2014). In data streams, the novelty detection can be divided into two phases: *offline* and *online phase*. In the offline phase, a classification model is trained using an initial, static, labelled dataset. In the online phase, the model is updated using unlabelled data arriving in streams. The update occurs when the predictive performance of the model decreases, usually because of a change in the data distribution. Thus, the model can be continuously updated (GAMA *et al.*, 2014).

One of the strategies to deal with novelty detection and concept drift is by explicitly detecting changes in parts of the stream, comparing the current concept with previous concepts from time to time (LOSING; HAMMER; WERSING, 2016). An example of this strategy is to continuously calculate the model classification error. This strategy assumes that the data arriving in the stream are labelled.

Another strategy is to store in a buffer the potential novelty classes instances. However, the use of a buffer with a fixed size may ignore instances with relevant information about persistent concepts. Furthermore, the size of the buffer affects its efficient use when the degree and speed of changes vary in the data stream. Another deficiency of updating the model using a buffer with a fixed size is the possibility to forget old, but relevant information.

There are two problems in assuming that the arriving data is labelled. First, the process of labelling an instance usually has a cost, which increases with the complexity and the need for domain expertise. Second, if the data arrives in high speed, there will not be enough time to label them. Thus, in this paper, we assume that the instances in a data stream come unlabelled.

Due to the lack of labelled data in data streams, the update of the model can rely only on the predictive attribute values. Clustering algorithms can be used to deal with this limitation. Clusters can summarise the main data profiles present in a data stream and be updated to incorporate changes in class profiles and detect the appearance of novelties (AGGARWAL, 2007). When clustering algorithms are applied to data streams, micro-clusters can be used as a strategy to summarise data present in different periods of time (AGGARWAL *et al.*, 2003). Each micro-cluster can be structured as a temporal extension of a CF (Cluster Feature Vector) (ZHANG; RAMAKRISHNAN; LIVNY, 1996), which is a compact statistical representation of a set of instances.

In this paper we propose *Higia*, a novelty detection algorithm based on *k*-Nearest Neighbor (*k*NN) that uses *micro-clusters* (AGGARWAL *et al.*, 2003) as prototypes and incrementally updates the micro-clusters or creates new micro-clusters when a novelty is detected. *Higia* training is divided into offline learning and online learning. During the offline learning phase, we assume that there is data from one or more normal classes. The instances from each normal class are summarised into a set of micro-clusters. Each micro-cluster has instances from the same normal class label. In the online learning phase, each instance close to a micro-cluster is considered an extension of the micro-cluster, a concept drift. This instance is then used to adapt the predictive model to this concept drift. However, if a set of new instances are close together in a dense region, they are considered representative of new classes, named novelties.

This paper is structured as follows. Section 4.2, presents previous related works for novelty and concept drift detection in data streams. The concepts data stream, novelty, concept drift and micro-clusters are introduced in Section4.3. The proposed algorithm, Higia, is described in Section 4.4. Section 4.5 presents the experimental setup and analyses the results obtained. Finally, Section 4.6 has the main conclusions from this study and points out future work directions.

4.2 Related Work

This section briefly presents previous works using ML-based approaches for novelty and concept drift detection in data streams. Most of these studies use supervised algorithms to induce classifiers.

Most of the classification algorithms proposed for data stream mining are based on online learning (FARIA; GAMA; CARVALHO, 2013; BIFET *et al.*, 2013; GAMA *et al.*, 2014; LOSING; HAMMER; WERSING, 2016). Some of them continuously update the classification model using true labelled data (MASUD *et al.*, 2011; ABDALLAH *et al.*, 2016; AL-KHATEEB *et al.*, 2012). However, as previously mentioned, true labels are not always available at feasible time, delaying the updating of the classification model. Others classification algorithm apply clustering algorithms in the arriving data when the data is unlabelled. Thus, the clusters are representatives of normal and new classes (FARIA; GAMA; CARVALHO, 2013; SPINOSA; CARVALHO; GAMA, 2009; HAYAT; HASHEMI, 2010; AMINI; TEH; SABOOHI, 2014).

One of the first algorithms to use clusters for novelty detection in data streams is OnLIne Novelty and Drift Detection Algorithm (OLINDDA) (SPINOSA; CARVALHO; GAMA, 2009), (SPINOSA; CARVALHO; GAMA, 2008). During the offline phase, a single model is built by a set of clusters with data from the normal classes. After, in the online phase, whenever a new instance arrives, it is calculated the distance between it and the closest cluster from the normal model. When the distance is large, according to a threshold value, the instance is stored in a buffer, where it can later be defined as a novelty after a clustering step.

Enhanced Classifier for Data Streams with novel class Miner (ECSMiner) (MASUD *et al.*, 2011) is an ensemble of models. Each model is represented by a set of clusters created using the clustering algorithm k-means. ECSMiner also stores in a buffer the instances that are distant from the normal clusters. The ensemble is updated when the instances stored in the buffer receive their true label. Afterwards, the ensemble predictive accuracy is calculated. The model with the lowest accuracy is updated with the novelties found in the buffer. While waiting for labelled data, the model can wait for a long period of time to be updated, which could reduce the accuracy of the ensemble. Besides, it is not always guaranteed that all data will be labelled, since it may be application dependent.

Another novelty detection algorithm, MultI-class learNing Algorithm for data Streams (MINAS) (FARIA; GAMA; CARVALHO, 2013), also uses an offline phase followed by an online phase. In its offline phase, the data is separated by labels in subsets. From each subset, it is generated a set of micro-clusters representing each class. In the online phase, the incoming data is stored in a buffer if it is not identified by the model. When the buffer reaches a certain size, a clustering algorithm is applied in the data stored in the buffer. Valid micro-clusters are classified as extensions of normal classes or as novelties and are incorporated into the model.

4.3 Problem Formalisation

A data stream is a sequence of instances, potentially of infinity size, that can be formally represented by (FARIA; GAMA; CARVALHO, 2013; GAMA *et al.*, 2014; GARCIA; CARVALHO; MENDES-MOREIRA, 2018): $D_{tr} = \{(X_1, y_1), (X_2, y_2), ..., (X_{tr}, y_{tr})\}$, where X_{tr} is an instance arriving in time tr and y_{tr} is the target class of this instance. Due to finite resources,

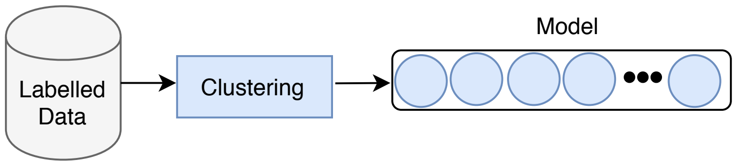


Figure 4 – Higia Offline Phase.

each instance must be processed only once.

Concept drift is a change in the distribution probability of a problem target classes (GAMA *et al.*, 2014). Formally, the joint probability distribution $P_{tr+1}(X,y)$ over instances X and label y can change over time, defined by tr, so that $P_{tr}(X,y) \neq P_{tr+1}(X,y)$.

Assuming that in the offline phase a dataset has m classes, the set $Y^{Nor} = \{y_1, y_2, \ldots, y_m\}$ represents the set of Normal Classes. These classes are used to build the initial classification model. Afterwards, during the online phase, a novel class, y_{m+1}, has the following property: $y_{m+1}, \notin Y^{Nor}$ i.e., y_{m+1} was not used in the training of the classification model, but emerges during the online phase. Therefore, for any given new set of novel classes, $Y^{New} = \{y_{m+1}, y_{m+2}, \ldots, y_n\}$ any novelty detection approach must be able to fast detect novelties as they appear (FARIA; GAMA; CARVALHO, 2013; GARCIA; CARVALHO; MENDES-MOREIRA, 2018). Considering the sets of normal classes and new classes, the total set of classes is simply defined as $Y = Y^{Nor} \cup Y^{New}$.

Micro-clustering (AGGARWAL *et al.*, 2003) is a strategy commonly used to summarise data coming from a stream in different periods of time. Each micro-cluster $C_j = (n, \vec{LS}, \vec{SS}, t)$ stores four components: the number of its instances n, the linear sum of its instances $\vec{LS}$, the square sum of its instances $\vec{SS}$ and the timestamp, t, of when the last instance was incorporated in the micro-cluster. By using these values, it is possible to calculate the centroid ($c_j = \vec{LS}/n$) and the radius ($r_{C_j} = 2 \times (\vec{SS} \times n/n^2 + \vec{SS}/n^2)$) of a micro-cluster, which can be used to classify new instances.

4.4 Methodology

The proposed algorithm, Higia, is based on the assumption that in a data stream, an ideal classifier should be able to learn the current concept in feasible time without forgetting relevant past information (BIFET *et al.*, 2013).

Higia induces a classification model using unsupervised online learning. The training also occurs by an offline phase followed by an online phase. In the offline phase, a predictive model, illustrated by Figure 4, is induced from a batch containing labelled data. As in (FARIA; GAMA; CARVALHO, 2013), the model is composed of micro-clusters created using the CluStream algorithm (AGGARWAL *et al.*, 2003).

In the online phase, illustrated by Figure5, as new data arrives, the model classify each new instance as *normal*, *extension* or *unknown*. For the classification, the model calculates the euclidean distance between each instance and the centroids of the micro-clusters from the *normal* classes.

Algorithm 2 – Higia: Online Phase

```
 1: input: $X_{tr}$, $T$, $k$
 2: Let $\psi_k$ be a list of the $k$ nearest micro-clusters to $X_{tr}$
 3: if majority of $\psi_k$ have the same label then
 4:     Let $C_j$ be the nearest micro-cluster to $X_{tr}$
 5:     Let $c_j$ be the centroid of $C_j$
 6:     Let $radius\left(C_j\right)$ be the radius of $C_j$
 7:     $dist \leftarrow EuclidianDistance(X_{tr}, C_j)$
 8:     if dist $\leq radius\left(C_j\right)$ then
 9:         update $C_j$ with $X_{tr}$
10:         classify $X_{tr}$ with the same label of $C_j$
11:     else if $dist \leq \left(radius\left(C_j\right) \times T\right)$ then
12:         create extension of $C_j$ with centroid $X_{tr}$ and radius 0.5
13:         classify $X_{tr}$ with the same label of $C_j$
14:     end if
15: else
16:     add $X_{tr}$ to buffer
17:     classify $X_{tr}$ as unknwon
18: end if
```

Algorithm 2 describes how Higia works in the online learning phase when a new instance, X_{tr}, arrives. First, Higia finds the k micro-clusters closest to X_{tr}. If the majority of these micro-clusters have the same label and if the smallest distance is less than the radius of the nearest micro-cluster C_j, the instance is classified with the label of C_j. Besides, C_j is updated with X_{tr}. If the distance is larger than the radius of C_j but is smaller than a given threshold of T, the instance is added to the model as an extension. The threshold is multiplied by the radius of C_j and indicates the maximum drift of C_j.

If the distance is larger than the radius of C_j and T, then X_{tr} is labelled as *unknown* and stored in a buffer. The instances stored in the buffer are incrementally forming micro-clusters. When a micro-cluster has a given amount of instances, defined by a hyperparameter, a novelty is found, and the new micro-cluster is added to the model as a new class.

4.5 Experimental Evaluation

In this section, we present the experiments carried out to assess the predictive performance of Higia. In these experiments, Higia was compared with two other algorithms, MINAS and the *k*NN algorithm. From the algorithms described in Section 4.2, MINAS is the only unsupervised algorithm for multiclass novelty detection. The *k*NN algorithm from the MOA framework (BIFET

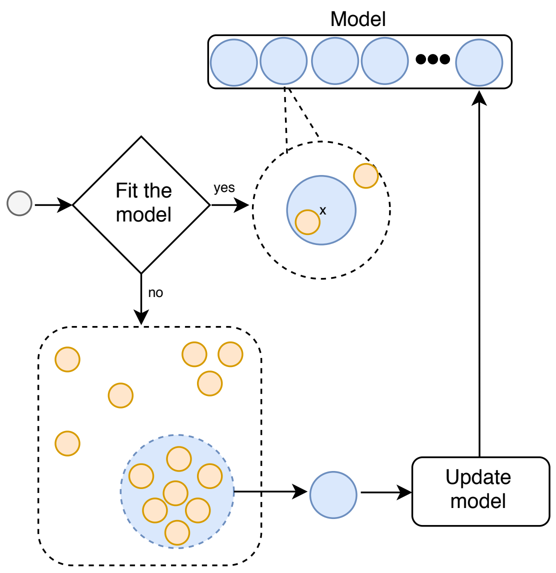


Figure 5 – Higia Online Phase.

et al., 2010b) was used as a baseline. In the offline phase, all three algorithms are initialised with the same batch of labelled data representing 10% of the dataset. We assume that the instances in the training data are from the normal classes and that new classes can continuously appear during the online phase.

We adopted the accuracy measure to evaluate the predictive performance of the models over time. The accuracy is the amount of correctly classified data divided by the amount of total data in a window of 1000 instances. We also use the total accuracy measure to evaluate the performance for the whole dataset.

For more detailed analyses, we used the evaluation approach proposed in (MASUD *et al.*, 2011). This evaluation approach has 3 measures: M_{New} = the percentage of instances from the new classes misclassified as existing class, F_{New} = the percentage of instances from the normal classes classified as novelties, and Err = average misclassification error.

In the evaluation metric accuracy, the instances classified as *unknown* are counted as misclassification. To analyse the algorithms in terms of misclassification of the total classes, (Y), without the influence of *unknown*, we used the Err measure.

4.5.1 Datasets

The experiments were performed on both synthetic and real datasets, commonly used in novelty detection studies (MASUD *et al.*, 2011; IENCO; ZLIOBAITE; PFAHRINGER, 2014; AL-KHATEEB *et al.*, 2012; FARIA *et al.*, 2016; AGGARWAL *et al.*, 2003; CAO *et al.*, 2006). The synthetic datasets are MOA, SynD, CDT, UG and Gear. The real dataset used was the Forest Cover dataset.

The *MOA* dataset (FARIA; GAMA; CARVALHO, 2013) has concept drift, the appearance

Table 4 – Statistical information for each dataset

Statistics	1CDT	MOA	Gear	UG	SynD	Forest Cover
Attributes	2	4	2	2	10	54
Classes	2	4	2	2	2	7
Normal Classes	1	2	2	1	2	3
New Classes	1	2	0	1	0	4
Instances MinCla	7199	9987	99935	44999	124660	587
Instances MajCla	7200	18180	100065	45000	125340	18350

of new classes, recurrence and disappearance of existing classes. The real shape of the clusters in this dataset is normally distributed hyper-spheres. The *SynD* (MASUD *et al.*, 2011; AL-KHATEEB *et al.*, 2012; FARIA *et al.*, 2016) has no novelty, but the concept associated with known classes changes over time. Finally, *CDT, UG* and *Gear* are non stationary datasets [1]. In these datasets, a single novelty occurs, and concept drifts happen at every interval of 400 instances in CDT, 1000 instances in UG and 2000 instances in Gear.

Regarding the only real dataset, *Forest Cover* (ASUNCION; NEWMAN, 2007), it contains observations from 7 types of forest in the United States. It has 7 classes and 54 attributes. In the experiments, the training set is formed by observations from 3 normal classes.

Table 4 presents some basic statistics collected from these datasets: the number of normal and new classes, number of attributes, number of instances in the minority and majority classes.

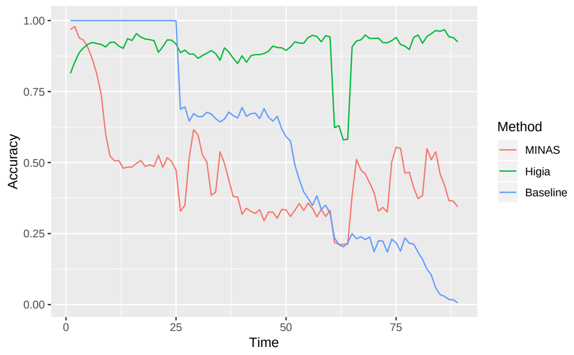
4.5.2 Results and discussion

In these experiments, we used the default parameters of MINAS. Because MINAS uses the label of the nearest micro-cluster to classify a new instance, we set $k = 1$ in the Higia algorithm. As for the other parameter of Higia, the $threshold$, we set it to $threshold = 1.1$ as in MINAS (FARIA; GAMA; CARVALHO, 2013). Also for Higia, we experimentally defined the parameter $radius = 0.5$. The kNN has the default parameters, $k = 1$ and window size ($w = 1000$), and there is no online training. It is used to understand how the model loses predictive accuracy over time without its update to the changes in the data stream.

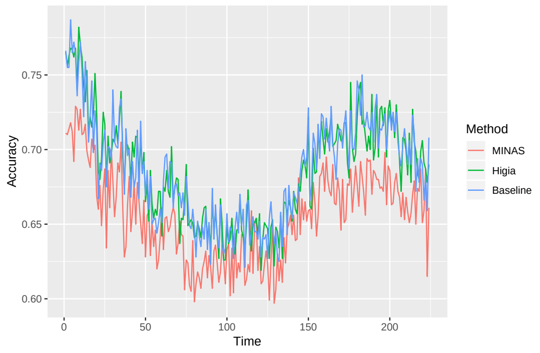
Next, we present the predictive accuracy overtime of the 3 algorithms. As can be seen in Figure 6, the predictive performance of Higia was better than those obtained by MINAS and the kNN baseline. Higia incrementally updates the normal and the new micro-clusters in the classification model. As a consequence, the model represents the current probability distribution of the data streams better than MINAS. The model induced and updated by the MINAS algorithm is sensitive to the buffer size, i.e., the model needs to wait until the buffer is full of *unknown*

[1] https://www.sites.google.com/site/nonstationaryarchive/

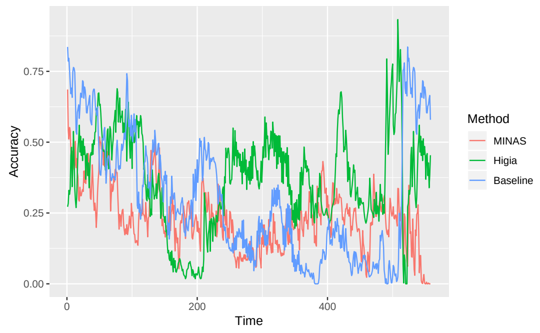
instances to be updated. Thus, by not being able to quickly adapt to the changes in the data stream, the model loses accuracy along the time.



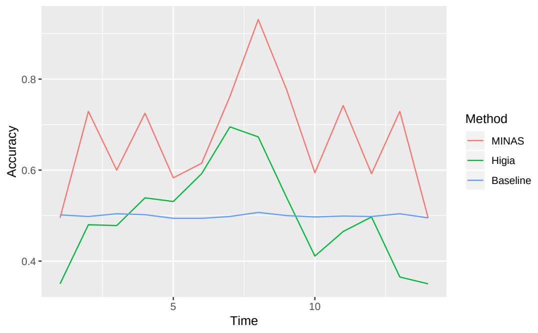
(a) MOA



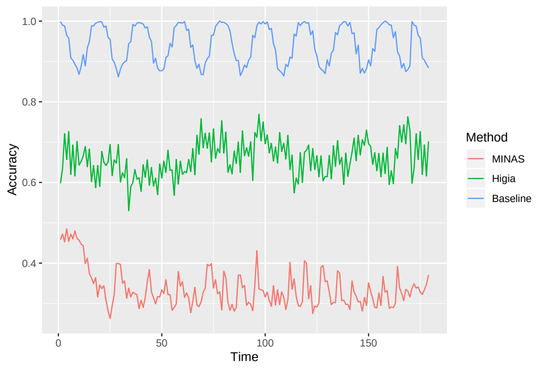
(b) SynD



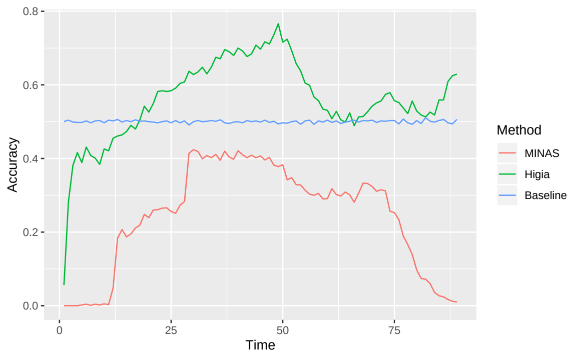
(c) Covertype



(d) CDT



(e) GEARS



(f) UG

Figure 6 – Predictive accuracy over time.

The baseline, *k*NN, presented the worst predictive performance in most of the datasets. This is somewhat expected since it does not learn along the time. However, we can see from Figure 6e that, for one of the datasets, the baseline had the best predictive accuracy. A possible reason is that the training set of this dataset has data from all classes. Meanwhile, for this dataset, different from the baseline, the generalisation of the models induced by MINAS and Higia was not able to capture the behaviour of the data. This indicates the importance of the initial training even in data stream scenarios with concept drift.

Table 5 summarises the predictive performance obtained by the Higia and MINAS

Table 5 – Performance metrics for each algorithm

	M_{New}		F_{New}		**Err**		**Acc**	
	Higia	MINAS	Higia	MINAS	Higia	MINAS	Higia	MINAS
MOA	0.11	0.00	0.00	0.46	**0.08**	0.48	**0.62**	0.46
SynD	0.00	0.00	0.00	0.00	**0.30**	0.34	**0.70**	0.66
CoverType	0.18	0.46	0.49	0.24	**0.49**	0.54	**0.37**	0.23
CDT	0.43	0.00	0.01	0.00	0.46	**0.03**	0.47	**0.68**
GEARS	0.00	0.00	0.00	0.00	**0.34**	0.66	**0.66**	0.34
UG	0.78	0.41	0.03	0.36	**0.60**	0.70	**0.68**	0.24

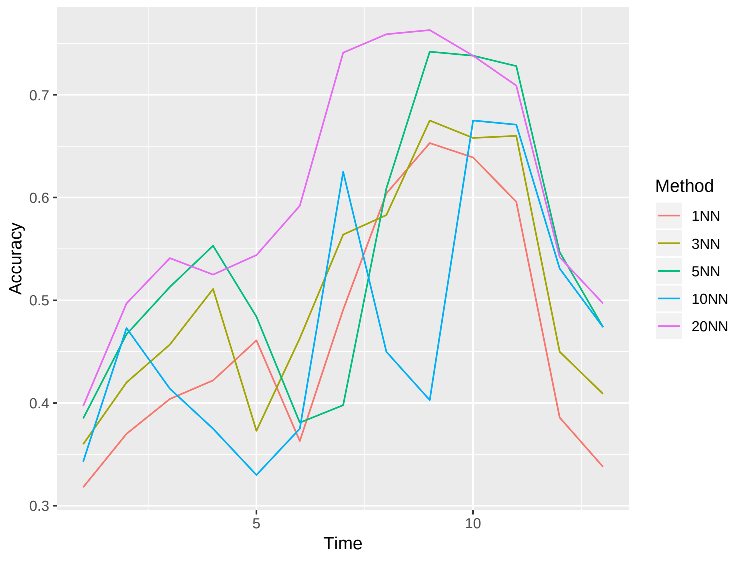


Figure 7 – Higia accuracy over time in CDT dataset varying k.

algorithms in all datasets. According to these results, for the datasets *MOA, CDT and UG*, Higia had the lowest F_{New}, but the highest M_{New}. This shows that Higia models gave more relevance to the normal classes than to the novelties. The only exception is the *CoverType* dataset. However, these results did not seem to affect the accuracy, Acc, of Higia in most datasets. Higia presented the lowest Err values in almost all datasets, meaning that, for these datasets, the model made fewer misclassifications and labelled less instances as *unknown* than MINAS.

An exception occurred for the dataset *CDT*, which has a seasonal overlap between the normal class and the new class. For this dataset, Higia incremental learning mixed instances from different classes into the same micro-clusters, reducing Acc and increasing Err.

Additional experiments were performed to analyse the impact of the parameter k ($k = 1, 3, 5, 10, 20$) in Higia predictive accuracy. We used the *CDT* dataset because Higia had a lower performance with this dataset. Figure 7 shows that, for this dataset, the accuracy is more stable over time for higher k. We also see a reduction of Err, Table 6, with the increase of the parameter k.

Table 6 – Higia ($k = 1, 3, 5, 10, 20$) performance metrics for dataset CDT

	MNew	FNew	Err	ACC
1NN	0.43	0.01	0.46	0.47
3NN	0.42	0.01	0.41	0.51
5NN	0.38	0.01	0.32	0.54
10NN	0.32	0.02	0.32	0.47
20NN	0.45	0.02	0.35	0.60

4.6 Conclusions and Future Work

Data stream mining has gained a great deal of attention in the last decades. Novelty detection algorithms have been successfully applied to many applications. However, most of the proposed algorithms assume that instances arriving in a stream are labelled, which is often not the case.

This paper presented Higia, a new unsupervised learning algorithm based on micro-clusters for novelty and concept drift detection in data streams. The micro-clusters are incrementally updated every time a new instance arrives from a data stream. When a novelty is detected, Higia creates new micro-clusters to represent the new class.

Considering several performance metrics, Higia was compared with MINAS, a state-of-the-art unsupervised novelty detection algorithm, and a k-NN-based baseline. According to the experimental results, Higia presented a better predictive performance than these other algorithms. Besides, Higia labelled less instances as *unknown* because it can faster adapt the model to the current concept than the compared algorithms.

As future work, the authors want to study alternatives to discard outdated information and the inclusion of an unsupervised concept drift tracker.

CHAPTER

5

AN ENSEMBLE OF UNSUPERVISED APPROACHES FOR NOVELTY DETECTION IN DATA STREAMS

Collaborating authors

Elaine R. Faria
University of Uberlandia, Uberlandia, Brazil

Cláudio Rebelo de Sá
University of Twente, Enschede, The Netherlands

João Mendes-Moreira
LIAAD-INESC TEC, University of Porto, Porto, Portugal

Charu C. Aggarwal
IBM T. J. Watson Research Center, Yorktown Heights, USA

André C. P. L. F. de Carvalho
University of São Paulo, São Carlos, Brazil

Joost N. Kok
University of Twente, Enschede, The Netherlands

Abstract

In data streams, new classes can appear over time due to changes in the statistical data distribution. Consequently, models can become outdated, which requires the use of incremental learning algorithms capable of detecting and learning the changes over time. However, when a single classification model is used for novelty detection, there is a risk that its bias may not be suitable for new data distributions. A solution could be the combination of several models into an ensemble. We propose two unsupervised ensemble approaches for novelty detection in data streams: one combining clustering partitions using the same clustering technique; and other using different clustering techniques. We compare the performance of the proposed methods with well-known novelty detection algorithms. The methods were tested on datasets commonly used in the novelty detection literature. The experimental results show that proposed ensembles have competitive performance for novelty detection in data streams.

5.1 Introduction

In many real-world scenarios, data continuously arrives at a high rate and in a non-stationary way. This type of data is commonly referred to as *data streams*. As new data arrives, previously trained models can become outdated (SILVA *et al.*, 2013). Thus, a model needs to be incrementally updated to prevent predictive loss. In addition, due to the great amount of data generated, it is impossible to store it all in the main memory, requiring the elimination of outdated data and the online processing of the incoming data (FARIA; GAMA; CARVALHO, 2013).

Three types of changes can be found in the literature of data streams: concept drift (GAMA *et al.*, 2014), recurring concepts (AL-KHATEEB *et al.*, 2012) and novel concepts (FARIA; GAMA; CARVALHO, 2013). Concept drift refers to changes in the statistical properties of a known concept; for example, it can be a change in the stochastic process that generates the data (GAMA *et al.*, 2014). Recurring concepts are a special type of concept drift in which concepts that appeared in the past may recur in the future (AL-KHATEEB *et al.*, 2012). Novelty concepts are patterns that are not present during the training of a classification model (FARIA; GAMA; CARVALHO, 2013) but appear in the data stream.

Novelty detection is a machine learning task based on the identification of new concepts (DING *et al.*, 2014). This approach can be consider as a binary classification task, composed by *normal* and *abnormal* classes (SPINOSA; CARVALHO; GAMA, 2009). However, it can also be consider as a multi-class classification task (FARIA; GAMA; CARVALHO, 2013), in which more than one new class can emerge in the stream. The *abnormal* classes can also be named as *not normal* (SPINOSA; CARVALHO; GAMA, 2009), *anomaly* (MASUD *et al.*, 2013) or *novel/new* (FARIA; GAMA; CARVALHO, 2013) classes. In this paper, we followed the notation from (FARIA; GAMA; CARVALHO, 2013). In the latter, normal concepts are a set of classes used to train the classification model and novelty concepts are the *new classes* that emerge over time.

Most of the existing approaches for novelty detection consider that a model is updated only with labelled data. This implies that a model needs to wait for the data‘s label before update itself. However, this delay can affect the predictive performance of the model over time. In the absence of labels, an alternative is to use unsupervised approaches. In this case, the clusters represent normal and new classes (AMINI; TEH; SABOOHI, 2014), where a class can be composed of one or more clusters (SPINOSA; CARVALHO; GAMA, 2009).

A consistent and representative clustering partition should be able to address scalability; moreover, should detect concept changes over time in the data streams (AGGARWAL *et al.*, 2003). However, a unique partition could not well represent the statistical distribution of a dataset (VEGA-PONS; RUIZ-SHULCLOPER, 2011). For example, the k-means algorithm (BISHOP, 2007) can build different partitions, even with the same hyper-parameters, due to the randomness in the algorithm initialisation.

Instead of using one clustering partition, we propose an ensemble of clustering partitions for novelty detection in data streams. We consider one ensemble obtained by the combination of the CluStream algorithm (AGGARWAL *et al.*, 2003), referred here as Homogeneous ensemble Clustering for data Streams (HoCluS). We also consider another ensemble with different clustering techniques, referred here as Heterogeneous ensemble Clustering for data Streams (HeCluS). This approach allows the use of clustering techniques with different bias, in order to obtain more robust classification models. Hence, as new data s, each clustering technique can independently create and update a predefined number of partitions.

In order to compare the performance of these two different approaches, we implemented them in MINAS (MultI-class learNing Algorithm for data Streams) (FARIA; GAMA; CARVALHO, 2013), a single classifier novelty detection algorithm for data streams. We conducted a set of experiments using datasets, commonly referred to in the novelty detection literature.

This work is an extension of (GARCIA *et al.*, 2019a). In this study, our main objectives are:

- To investigate if the use of an ensemble of clustering can improve the predictive performance of a single model classifier, in this case, the algorithm MINAS.
- To empirically study the behaviour of an ensemble of the same clustering approach and different approaches of clustering.
- To investigate if the proposed ensembles of clustering achieve competitive predictive performance in comparison to ensembles of supervised algorithms.

This paper is organised as follows. In Section 5.2 we present related work on novelty detection in data streams. In Section 5.4, we describe the proposed approaches and how they are incorporated into the MINAS algorithm. Section 5.5 presents the experiments performed. Finally, we conclude and discuss future research in Section 5.6.

5.2 Related Work

Several machine learning approaches have addressed novelty detection in data streams. Following we describe the principal approaches according to two aspects: i) number of classification models and ii) strategy to update the classification model.

Considering the first aspect, we can divide the existing approaches in the single classification model or the ensemble of models. Most of the single classification approaches use a *k*NN classification model based on a clustering approach (SPINOSA; CARVALHO; GAMA, 2009; HAYAT; HASHEMI, 2010; FARIA; GAMA; CARVALHO, 2013; ABDALLAH *et al.*, 2016). These approaches can forget old clusters, insert new clusters and update the existing ones. However, even though a single model is computationally less costly to train and update, it may not be the most suitable to all periods of a stream.

In contrast, other works focus on ensemble models. Ensemble classification for novelty detection in data streams are usually formed by combining classification models from the same strategy (MASUD *et al.*, 2010; MASUD *et al.*, 2011; Farid; Rahman, 2012; FARID *et al.*, 2013; AL-KHATEEB *et al.*, 2012). In most approaches, the update of an ensemble consists of replacing the worst accurate model by a new one. The new model is trained with the last labelled data chunk. However, the disadvantage is that the ensemble can wait for an extended period before the labels are known. During this waiting period, the predictive performance of the ensemble could decrease drastically.

Considering the second aspect, the possible update strategies are: the *supervised* or the *unsupervised*; which will depend on the presence/absence of labelled instances. Supervised approaches assume that the true label of all instances will eventually be available to update the model. Some examples of models using supervised learning are decision trees (MASUD *et al.*, 2010; MASUD *et al.*, 2011; Farid; Rahman, 2012; FARID *et al.*, 2013) and *k*NN (AL-KHATEEB *et al.*, 2012; MASUD *et al.*, 2011; HAQUE; KHAN; BARON, 2015).

On the other hand, unsupervised learning approaches assume that the true label will not be available. Therefore, they need to update the classification model without external feedback (SPINOSA; CARVALHO; GAMA, 2009; FARIA; GAMA; CARVALHO, 2013; HAYAT; HASHEMI, 2010; TAN; TING; LIU, 2011). In general, they use the *k*-means algorithm to extract clusters to represent the current classes. As a result, they have some limitations: find only hyperspherical clusters, have a fixed number of clusters and are sensitive to outliers.

5.3 Problem Formalisation

Formally a data stream D_t is a potentially infinite sequence of instances arriving in a time $t, t \in \{1, ..., \infty\}$. Where, each instance, X, contains d dimensions denoted by $X_t = (X^1, ..., X^d)$ and a target class y_t. A data stream can be represented as (FARIA; GAMA; CARVALHO, 2013;

GARCIA; CARVALHO; MENDES-MOREIRA, 2018): $D_t = \{(X_1, y_1), (X_2, y_2), ..., (X_t, y_t)\}$.

A concept is a cluster $\mathscr{C}_t$ that contains a centroid c_t, a radius r_t, a threshold T_t and a class label y. The radius of a cluster is the Euclidean distance between its centroid and the farthest data point in that cluster (MASUD *et al.*, 2011). The threshold is the area outside a cluster where the cluster could extend. Thus, this area is $T_t = h * r_t$, where h is a given value.

Concept drift is a change in the distribution probability of target classes (GAMA *et al.*, 2014). Formally, a distribution P in a given time t conditioned by the instance X and label y can suffer changes affecting the conditional probability, $X : P_t(X, y) \neq P_{t+1}(X, y)$. As a result, a model built during time t could be outdated in time $t + 1$.

In data streams the novelty detection can be divided into two phases: the *offline* and the *online phase*. Assuming that in the offline phase, a dataset has m classes. Then, $Y^{Nor} = \{y_1, y_2, ..., y_m\}$ represents the set of normal classes. These class labels and the corresponding data samples are used to build the initial classification model. When during the online phase, a novel class with label y_{m+1} emerges, a novelty detection approach needs to detect this new class (concept) as quickly as possible and update the classification model accordingly.

5.4 Ensemble Clustering for Data Streams

The idea of Ensemble Clustering for Data Streams is similar to the general concept of combining classification models to construct an ensemble (VEGA-PONS; RUIZ-SHULCLOPER, 2011). For that reason, it can be used to find a set of partitions to represent a dataset. The most common ensemble learning algorithms are AdaBoost (FREUND; SCHAPIRE, 1997) and Bagging (BREIMAN, 1996).

In this work, we created an ensemble of clustering. This process is divided in: the generation of a set of partitions and the combination of partitions into an ensemble of clustering.

5.4.1 The MINAS Algorithm

To test the proposed approaches, we implemented HoCluS and HeCluS into the algorithm MINAS (FARIA; GAMA; CARVALHO, 2013). The algorithm MINAS, in the offline phase, has a single model trained with labelled data from the *normal* classes. This phase happens only once at the initial stage. The dataset with the labelled instances is split into subsets of data, each one containing data from one class in Y^{Nor}. Then, a clustering algorithm is applied to each subset to create a partition for each class. These partitions will form a single model that contains the clusters found in the offline phase.

In the online phase, it is calculated the Euclidean distance between each new instance and all the centroids of the clusters from the model. If the smallest distance is less than the radius of the closest cluster, then the instance is labelled with the same label of the cluster. Otherwise, the instance is labelled as *unknown* and stored in a fixed size buffer for future analysis. When the buffer reaches its limit, a clustering algorithm is applied to obtain new clusters.

A new cluster is incorporated into the model as *concept drift* if the distance between its centroid and the centroid from the nearest cluster, from a *normal* class, is below the threshold from the nearest cluster. Otherwise, the cluster is considered a *novelty*. At the end of this process, the buffer is empty and the process is repeated every time the buffer is full with *unknown* instances.

5.4.2 Ensemble Clustering Applied To MINAS Algorithm

In this section, we will explain how the two HoCluS and HeCluS were embedded in the MINAS algorithm [1]. Considering the MINAS implementation, in both offline and online phase, the ensemble of partitions are created following the two steps: generation and combination. In the generation step, a predefined number of P partitions from N clustering techniques is generated, from the dataset D_t. The output is an ensemble, L_N, containing $P \times N$ clustering partitions.

In the offline phase, the labelled instances are separated by their labels in subsets. For each subset, it is applied an Ensemble Clustering Generator, Algorithm 3, and P partitions from N clustering techniques are generated. This process is illustrated in Figure 8. The figures in grey represent the original MINAS algorithm, and the coloured figures with dashed lines indicate our proposed method.

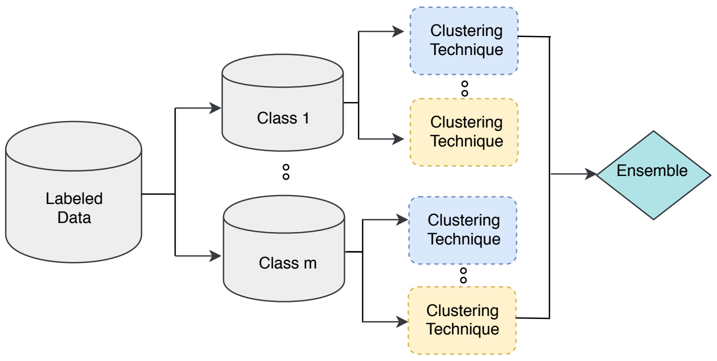


Figure 8 – Offline Phase with the implementation of the algorithm Ensemble Clustering Generator

The online phase is represented in Figure 9. To classify an incoming instance each partition from the ensemble, blue diamond figure, gives a vote about the label of the instance. The class is determined by the majority vote of the partitions from the ensemble. Thus, an instance is classified as *normal*, if the majority agrees on the label; or as *unknown*, if there is no agreement about the instance label. The instances classified as *unknown* are stored in a buffer for future analysis.

[1] The codes of HoCluS and HeCluS are available at: <https://github.com/Keh/Ensemble-Clustering-For-Data-Streams.git>

Algorithm 3 – Ensemble Clustering Generator

```
 1: input: $P$, $N$, $D_t$
 2: output: $L_N$
 3: $L_N \leftarrow NewList(\{\})$                          ▷ Start an empty list
 4: for $i = 1$ to $N$ do
 5:     $L_i \leftarrow$ NewList({})                        ▷ Start an empty list
 6:     for $j = 1$ to $P$ do
 7:         $P_j \leftarrow Clustering(D_t)$    ▷ A generic Clustering function receives as input the data
 8:         $L_i$.insert($P_j$)
 9:     end for
10:     $L_N$.insert($L_i$)
11: end for
12: return $L_N$
```

When the buffer reaches a given size of W, new partitions are generated. In Figure 9, the circles/ellipses figures represent clusters and each colour is a label. The Ensemble Clustering Combiner (Algorithm 4), blue rectangle, is applied on these new partitions. In this step, we verify which clusters, from different partitions, are similar to each other, so, they can be labelled as the same cluster. For that, the consensus function computes the Euclidean distance between the centroids of each cluster from the different partitions.

Algorithm 4 – Ensemble Clustering Combiner

```
 1: input: $L_N$
 2: output: $L_N$
 3: $label \leftarrow null$
 4: for each $p_i$ in $L_N$ do
 5:     for each $c_j$ in $p_i$ do
 6:         $label \leftarrow c_j.label$
 7:         for all $c_k$ in $L_N \setminus \{pi\}$ do
 8:             $dist \leftarrow EuclideanDistance(c_j, c_k)$
 9:             if $dist < (radius_{c_j} + radius_{c_k})$ then
10:                 $c_k.label \leftarrow label$
11:             end if
12:         end for
13:     end for
14: end for
15: return $L_N$
```

The consensus function is based on the rule: a cluster $\mathscr{C}_j$ with centroid c_j is similar to a cluster $\mathscr{C}_k$ if the Euclidean distance between them is less than the sum of their radius: $D(c_j, c_k) < (r_j + r_k)$. The latter, it will be maintained only the clusters that are similar to other clusters from different partitions. The ones that are not similar are discarded. These remaining clusters, Consensus Clustering, can be classified as novelties or concept drifts.

A new cluster will be considered a *novelty* if the distance of its centroid is bigger than the threshold of the closest cluster from the model. Otherwise, the cluster is a *concept drift* if the distance of its centroid is bigger than the radius of the closest cluster from the model, but is smaller than the threshold of the closet cluster.

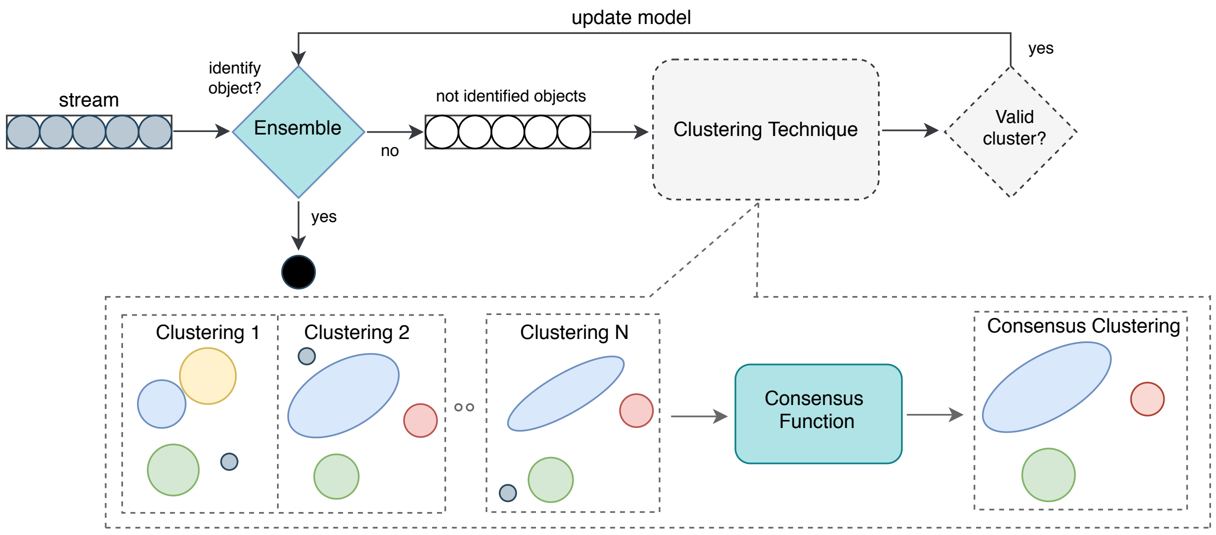


Figure 9 – Online Phase with the implementation of the algorithm Ensemble Clustering Combiner

5.4.2.1 *Homogeneous Ensemble Clustering for Data Streams*

We define Homogeneous ensemble Clustering for data Streams (HoCluS) as an ensemble clustering obtained by the combination of P partitions from the same clustering algorithm. In this work, we will use the algorithm for clustering in data streams CluStream (AGGARWAL *et al.*, 2003). CluStream is based on the k-means algorithm. Because of the random initialisation phase of k-means, different partitions can be obtained. Because of that, an ensemble of CluStream partitions can be more robust for novelty detection than a single partition of CluStream.

5.4.2.2 *Heterogeneous Ensemble Clustering in Data Streams*

We define a Heterogeneous Ensemble Clustering for data Streams (HeCluS) as the combination of P partitions from N different clustering techniques. In this work, the HeCluS has one ensemble model built from each one of the following clustering algorithms for data streams: CluStream (AGGARWAL *et al.*, 2003), DenStream (CAO *et al.*, 2006) and ClusTree (KRANEN *et al.*, 2011). The main motivation to use DenStream is because it is a stream clustering algorithm that is able to find clusters with arbitrary shape. Besides, it can also handle outliers (CAO *et al.*, 2006). We also use the ClusTree algorithm because it has a different bias from the other two algorithms. ClusTree builds clusters in a hierarchical data structure and can automatically set clusters with arbitrary shape.

5.5 Experimental setup and results

We conducted several experiments to compare the predictive performance of our methods and other well-known novelty detection methods for data streams.

5.5.1 Datasets

The experiments were performed with synthetic and real datasets, both commonly used in novelty detection studies (MASUD *et al.*, 2011; FARIA *et al.*, 2016; AGGARWAL *et al.*, 2003; CAO *et al.*, 2006). Table 7 presents for each dataset: the number of normal and new classes, number of attributes, number of instances in the minority and majority classes.

The synthetic datasets used are: MOA, SynD, 1CDT, UG_2C_2D and Gear. The *MOA* dataset (FARIA; GAMA; CARVALHO, 2013) has concept drift, the appearance of new classes, recurrence and disappearance of existing classes. The clusters in this dataset are shaped as normally distributed hyperspheres. The *SynD* (MASUD *et al.*, 2011) does not contain new classes, but does include concept drift. Finally, *1CDT, UG_2C_2D* and *Gear* are non stationary datasets [2]. In these datasets a single novelty occurs and concept drifts happen every 400 instances in 1CDT, 1000 instances in UG_2C_2D and 2000 instances in Gear. We also used in the experiments two real datasets: *Forest Cover* and *KDD-CUP'99 NetWork Intrusion*[3]. The *KDD* dataset was used by (AGGARWAL *et al.*, 2003) and (CAO *et al.*, 2006).

Table 7 – Statistical information for each dataset

Statistics	1CDT	MOA	Gear	UG_2C_2D	SynD	Forest Cover	KDD99
Attributes	2	4	2	2	10	54	38
Classes	2	4	2	2	2	7	23
Normal Classes	1	2	2	1	2	3	18
New Classes	1	2	0	1	0	4	5
Instances MinCla	7199	9987	99935	44999	124660	587	2
Instances MajCla	7200	18180	100065	45000	125340	18350	254058

5.5.2 Evaluation

We assume that the instances in the training data are the *normal* classes and the *new* classes/concept drifts can appear during the online phase. In the offline phase, all methods are initialised with a batch of labelled data representing 10% of the data. The rest of the data comes as a stream during the online phase.

We present the results of each experiment with a confusion matrix (e.g. Table 8). For each method, the table contains the percentage of: the instances from class C1 correctly classified

[2] <https://www.sites.google.com/site/nonstationaryarchive>
[3] <http://archive.ics.uci.edu/ml/index.php>

Table 8 – Example of a confusion matrix

	C1	C2
C1	TP	FP
C2	FN	TN
Novelty		
Unknown		
F-Measure		

(TP), the instances from C2 correctly classified (TN), the instances from C2 misclassified as C1 (FP) and instances from C1 misclassified as C2, *unknown* or *novelty* (FN). The *unknown* is the percentage of instances that the ensemble did not agree on a label. The novelty is the percentage of instances classified as *novelty*. The sum of each column should be 1. However, since we represent the average of 30 runs, the sum might not precisely be 1.

Whenever an instance is labelled as *unknown*, it is considered as a classification error, and it counts as a *false negative* (FN), which is a different approach adopted by (FARIA *et al.*, 2016). For this reason, a high percentage of instances labelled as *unknown* will force the recall to be lower. This will make the comparison of the models fairer.

The *F-Measure* is the weighted harmonic mean of the precision, *pi*, and the recall, *ro* (FARIA *et al.*, 2016):

$$FMeasure = \frac{2 * pi * ro}{pi + ro}$$

where, $pi = \frac{TP}{TP+FP}$ and $ro = \frac{TP}{TP+FN}$.

We also studied the predictive performance of the MINAS, HoCluS and HeCluS over time. We computed the F-Measure of these methods every 10.000 instances. The CLAM, MINER and SAND were not used in this analysis due to difficulties to obtain the necessary information to calculate the F-Measure over time.

5.5.3 Hyperparameter tuning

We study how the predictive performance, F-Measure, of MINAS can be affected by varying its hyperparameters. We hope to see if an ensemble can outperform the most suitable hyperparameter configuration. In this experiment, MINAS was trained with the dataset *Forest Cover*. The hyperparameters of MINAS are: k (number of clusters), W (window size) and T (cluster threshold). We conducted this experiment by varying the number of clusters $k \in \{10, 20, 40, 80, 100\}$, window size $W \in \{500, 1000, \ldots, 3500, 4000\}$ and cluster threshold $T \in \{1, 1.25, 1.5, 1.75, 2.0\}$.

The results obtained are presented in Figure 10, where each subplot represents a different number of clusters, k. In each subplot, the x-axis represents the window size, W; and the grade of the blue colour of the dots represents the cluster threshold, T. From this plot, we observe that k seems to affect the performance more than the other hyperparameters. In terms of the window size W, the models tend to obtain higher F-Measure with small windows. For example, when $k = 100$, the F-Measure is higher for $W = 500$ and $W = 1000$, than when the $W = 3000$ and $W = 4000$. From this experiment, it is also possible to conclude that the performance can be affected by the randomisation of the algorithm. For example, when $k = 80$ or $k = 100$ and $W = 500$ the F-Measure has high variance, regardless of the threshold, T, value.

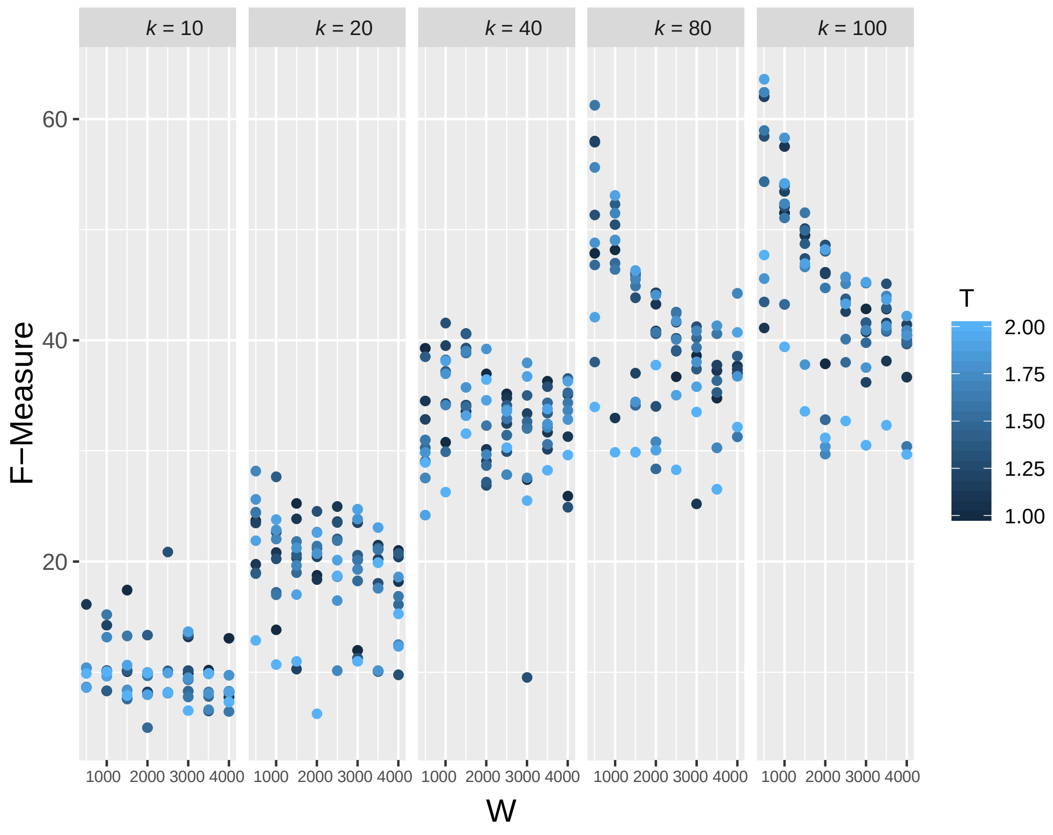


Figure 10 – MINAS with 3 hyperparameters x F-Measure for Forest Cover dataset

We also studied the performance of HoCluS with different numbers of partitions. We used this study to select the number of models in the remaining experiments. In Figure 11 the boxplots represent the variance of F-Measure of the HoCluS with different number of partitions when trained with the Forest Cover dataset. We tested HoCluS with 3, 10, 20 and 30 partitions, for each configuration we repeat the experiment 30 times. We observe that the variance of F-Measure reduces with the number of models. For example, we see that the variance is smaller with 30 models than with 3 or 10 models. However, to avoid the high computational cost, we decided to use HoCluS with 10 models.

In terms of the HeCluS, since the DenStream and the ClusTree have high computational cost, we used only 1 partition of each of the 3 different clustering approaches.

We compared the predictive performance of our methods with the original MINAS and

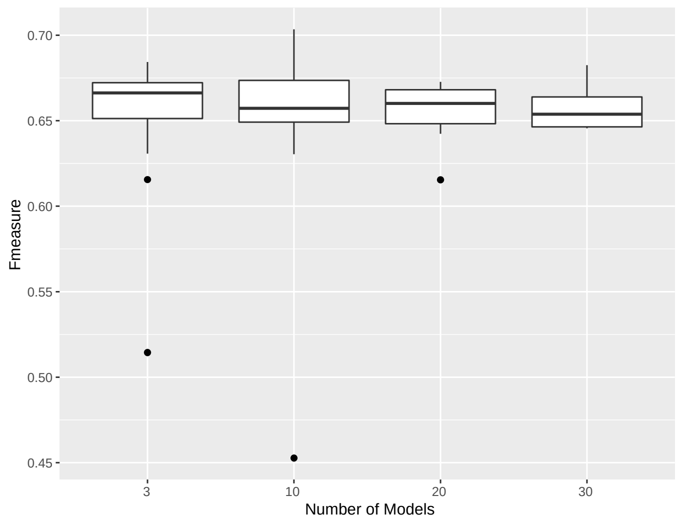


Figure 11 – Variability in F-Measures of different number of models of HoCluS for Forest Cover dataset

with three other supervised novelty detection methods: Miner (MASUD *et al.*, 2011), CLAM (AL-KHATEEB *et al.*, 2012) and SAND (HAQUE; KHAN; BARON, 2015). For simplicity, we use the default hyperparameters of the existing algorithms.

5.5.4 Results

In this section, we present and discuss the main results of the experiments. We note that the comparison between the performance of supervised and unsupervised approaches is not trivial. One might think that it is possible that the supervised approaches will always have a better performance because they have access to the true labels. However, in real scenarios, it is not always guaranteed that the incoming data is labelled. Thus, a supervised model can wait for longs periods of time to be updated.

5.5.4.1 Gear dataset

The Gear dataset has two *normal* classes, C1 and C2, both with concept drift in the online phase. In this experiment, we hope to understand how the predictive performance of the methods is affected by concept drift.

We can observe in Table 9 that the HeCluS has the highest *F-Measure*, compared with the other unsupervised methods, MINAS and HoCluS. Moreover, HeCluS does not misclassified C2 as a novelty; therefore, correctly detecting the concept drift in both classes. HeCluS is able to build models with non-spherical clusters, which can better represent the classes of this dataset. In terms of the supervised methods, MINER has the best *F-Measure*, with few misclassifications in both classes. In the case of SAND, it correctly classified C1; however, misclassified C2 as *novelties* or as *unknown*. Finally, CLAM correctly classified C1 but misclassified some data from C2 as C1 and the rest were classified as *unknown*. In other words, this means that CLAM was not able to learn the patterns presented in the window containing the *unknown* data.

Table 9 – Confusion matrix for Gear dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C 1	1.00	0.00	1.00	0.25	0.97	0.04	0.82	0.03	0.88	0.06	0.94	0.06
C 2	0.00	0.00	0.00	0.00	0.03	0.96	0.08	0.90	0.06	0.86	0.06	0.94
Novelty	0.00	0.57	0.00	0.00	0.00	0.00	0.10	0.07	0.02	0.03	0.00	0.00
Unknown	0.00	0.43	0.00	0.75	0.00	0.00	0.00	0.00	0.04	0.03	0.02	0.03
F-Measure	0.50		0.80		0.98		0.92		0.93		0.97	

In Figure 12 we have the F-Measure of the unsupervised methods over time. The methods HeCluS and HoCluS had more stable F-Measure than MINAS, especially the HeCluS method, which means that the ensembles adapted to the concept drift in a more efficient time than MINAS.

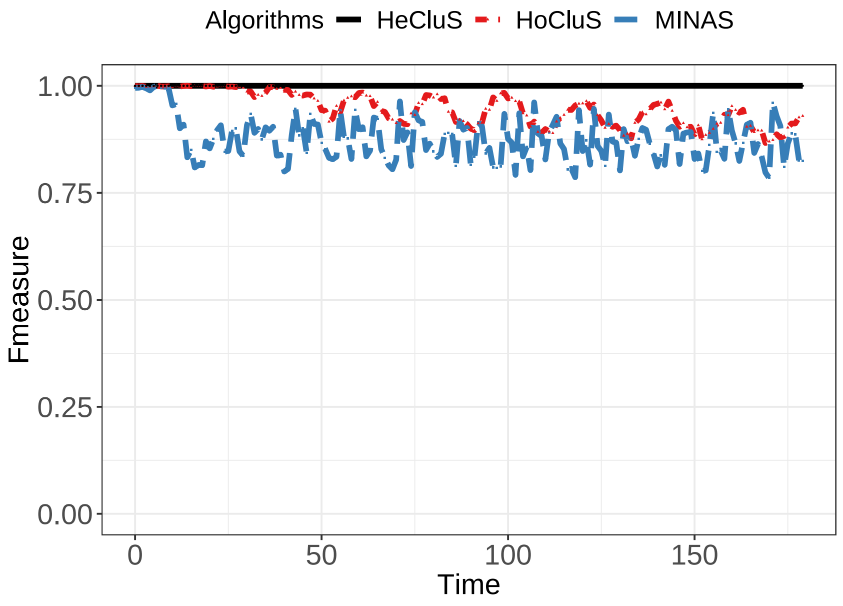


Figure 12 – F-measure of the unsupervised methods in the Gears dataset

5.5.4.2 SynD dataset

In the SynD dataset, C1 and C2 are the normal classes, and both classes have concept drift over time.

We observe in Table 10 that the unsupervised methods HeCluS and HoCluS have better performance than MINAS. We can see that these methods did not misclassify the concept drift as *novelties*. Considering the supervised methods, SAND had *F-Measure* 0.77; however, it has the highest percentage of *unknown* instances. The CLAM method did not classify any instance as *unknown* but misclassified C1 as C2 and C2 as C1. The MINER has the highest score and did not misclassify any instance as *unknown*.

Table 10 – Confusion matrix for Synd dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C 1	0.64	0.00	0.50	0.45	0.88	0.18	0.63	0.34	0.66	0.31	0.69	0.34
C 2	0.00	0.60	0.50	0.55	0.12	0.82	0.37	0.66	0.30	0.70	0.32	0.65
Novelty	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Unknown	0.36	0.40	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00
F-Measure	0.77		0.69		0.92		0.76		0.78		0.79	

In Figure 13 we have the F-Measure of the unsupervised methods over time. The methods are stable; however, they all lose a bit of F-Measure over time. The HeCLUS is slightly better than HoCluS and MINAS in most of the stream.

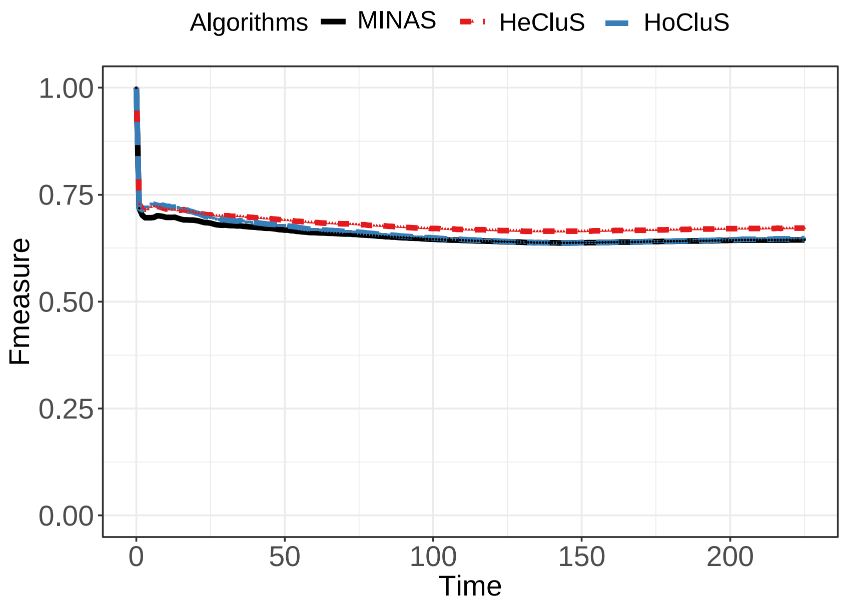


Figure 13 – F-measure of the unsupervised methods in the SynD dataset

5.5.4.3 1CDT dataset

The 1CDT dataset has two classes: a normal class (C1) and a new class (C2) with concept drift. With this dataset, we want to evaluate the performance of the models regarding novelty detection during the online phase. We note that MINAS, HoCluS and HeCluS are unsupervised methods; thus, even though they detect C2 as a novel class, their predictions are only represented as *novelty*.

Considering the unsupervised approaches, Table 11, MINAS and HoCluS have better score than HeCluS. HeCluS misclassified C2 as C1, which influenced in its F-Measure. Although the good performance of MINAS and HoCluS, these methods presented a higher percentage of *unknown* data than HeCluS. In terms of the supervised approaches, SAND presented the lowest *F-Measure*, because misclassified most data from C2 as C1. The reason for that could be that SAND gives a confidence score for each of its models, depending on their previous performance. Because of that, older models could have a higher score than new models, which could explain the misclassification of the new class C2. Both methods CLAM and MINER had a high performance for C2, combining the percentage of correct classification and *novelty*.

Table 11 – Confusion matrix for 1CDT dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C 1	1.00	0.84	1.00	0.00	1.00	0.00	0.94	0.00	0.94	0.00	1.00	0.20
C 2	0.00	0.03	0.00	0.73	0.00	0.71	0.00	0.00	0.09	0.00	0.00	0.00
Novelty	0.00	0.11	0.00	0.27	0.00	0.27	0.04	0.87	0.00	0.87	0.00	0.75
Unknown	0.00	0.02	0.00	0.00	0.00	0.02	0.06	0.12	0.00	0.13	0.00	0.06
F-Measure	0.62		0.92		0.99		0.95		0.95		0.93	

In Figure 14, we have the F-Measure of the unsupervised methods over time. We can see that the methods HoCluS and MINAS have stable F-Measure during the entire stream. In the case of HeCluS, it decreases F-Measure at the beginning of the stream, probably because of the appearance of the new class. However, HeCluS was able to update itself, which is possible to verify by the increasing F-Measure over time.

5.5.4.4 Forest Cover dataset

The Forest Cover dataset has 3 normal classes and 5 novel classes. Due to space constraints, we present the classes in the confusion matrix, Table 12, grouped in 2 groups: C1 represents the group of the *normal* classes and C2 represents the group of the *new* classes.

We can see in Table 12 that the unsupervised method HeCluS has the highest F-Measure, 0.73, of the methods, including in comparison with the supervised ones. The methods MINAS and HoCluS have both F-Measure of 0.67, which could indicate that the heterogeneous strategy of HeCluS works better for this dataset. The supervised method SAND has performance of 0.32, its low performance is explained by the misclassification of the normal and the novel classes. This could be because of its confidence score, tending to privilege majority classes, which is the case of the *normal* classes. The CLAM has a performance of 0.72, because it was able to learn most of the *novel* classes. Finally, the MINER had a F-Measure lower than CLAM, and misclassified part of C2 as C1.

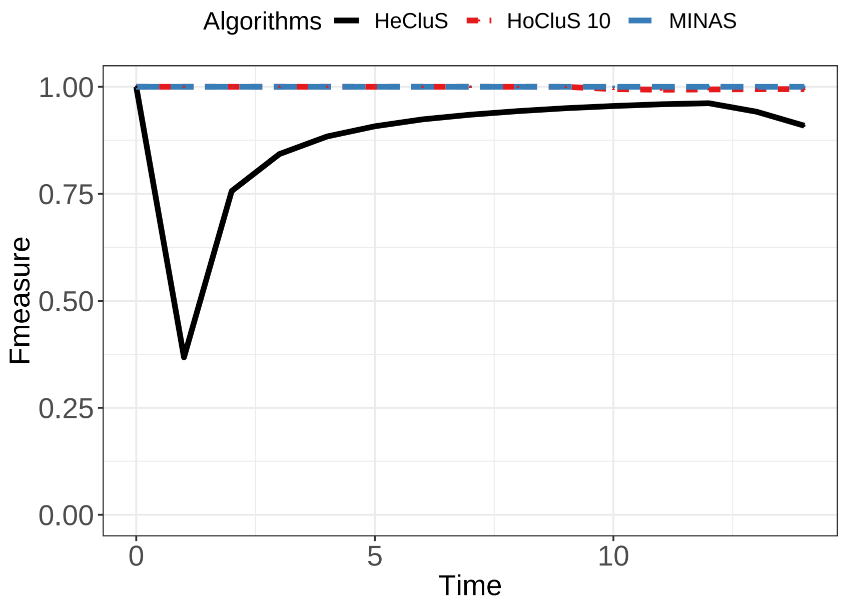


Figure 14 – F-measure of the unsupervised methods in the 1CDT dataset

This dataset has recurring classes that appear only during some periods. Due to this fact, CLAM showed higher F-Measure than MINER and SAND, because it is the only method with a mechanism for detecting recurrent classes.

The seasonal changes in this dataset affected all models, specially MINAS and HoCluS. We can observe in Figure 15 that HeCluS has higher *F-Measure* over time than MINAS and HoCluS. Until period 20 HoCluS has higher F-Measure than MINAS. After this period, MINAS has higher F-Measure than HoCluS.

Table 12 – Confusion matrix for Forest Cover dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C1	0.34	0.66	0.51	0.21	0.43	0.44	0.33	0.00	0.28	0.00	0.43	0.00
C2	0.47	0.14	0.24	0.53	0.27	0.37	0.00	0.00	0.00	0.00	0.00	0.00
Novelty	0.01	0.21	0.25	0.69	0.19	0.44	0.50	0.68	0.42	0.66	0.35	0.80
Unknown	0.00	0.03	0.22	0.07	0.00	0.00	0.03	0.05	0.17	0.14	0.04	0.02
F-Measure	0.32		0.72		0.59		0.67		0.67		0.73	

5.5.4.5 KDD dataset

The KDD dataset has 18 *normal* classes and 5 *novel* classes. Due to space constraints, in Table 13, C1 represents the group of *normal* classes and C2 represents the *novel* classes. This dataset is an example in which the methods have low F-Measure, which means that they were not able to learn the normal classes and to adapt to the changes over time.

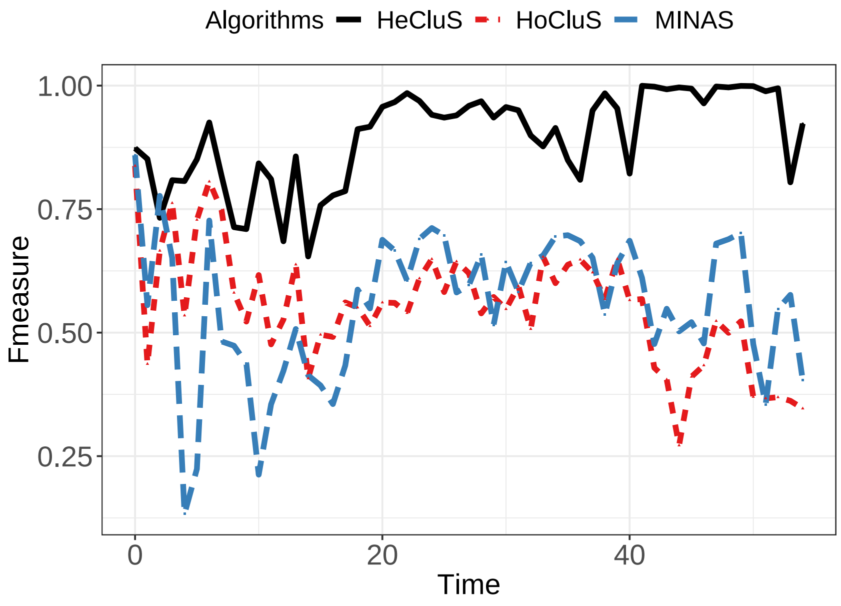


Figure 15 – F-measure of the unsupervised methods in the Forest Cover dataset

In Table 13, we see that HeCluS was slightly better than MINAS and HoCluS, because it correctly classified the normal classes. However, they are all unable to learn the new classes. The high dimensionality of this dataset could explain this. Besides that, this dataset has classes that probably were labelled as outliers. For example, see Table 7, the smallest class has only 2 instances. The CLAM was the supervised method that better classified the normal class C1; however, the rest of the data was classified as *unknown*. One can argue that classifying as unknown is better than misclassifying it as another class; however, although the method was able to recognize patterns that were different from the normal classes, it was not able to learn and adapt itself over time.

In Figure 16, we see the F-Measure overtime of the unsupervised methods. During period 0 to 15, all methods had unstable *F-Measure*; however, HeCluS had a higher score, than MINAS and HoCluS. During the time 15 to 30, HeCluS was stable with F-Measure close to 1.0, after this period it shows great instability. This could indicate that the HeCluS learned some classes that are more present during the period 15 to 30. However, we can see that MINAS and HoCluS, during the same period, did not learn the same classes. Thus, MINAS and HoCluS have low performance.

5.5.4.6 MOA dataset

The MOA dataset has 2 *normal* classes and 2 *novel* classes. Due to space constrains, in Table 14, C1 represents the group of *normal* classes and C2 represents the *novel* classes. In this dataset, the normal classes have concept drifts from time 0 to 90 and from time 30 to 55 they overlap. The first new class emerge at time 35 and second new class emerge after time 75.

Table 13 – Confusion matrix for KDD dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C 1	0.23	0.00	0.64	0.00	0.32	0.00	0.10	0.00	0.08	0.00	0.30	0.00
C 2	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.00
Novelty	0.25	0.23	0.00	0.00	0.15	0.04	0.46	0.05	0.46	0.06	0.18	0.14
Unknown	0.18	0.63	0.23	0.66	0.04	0.23	0.35	0.71	0.34	0.70	0.30	0.40
F-Measure	0.27		0.32		0.30		0.10		0.10		0.31	

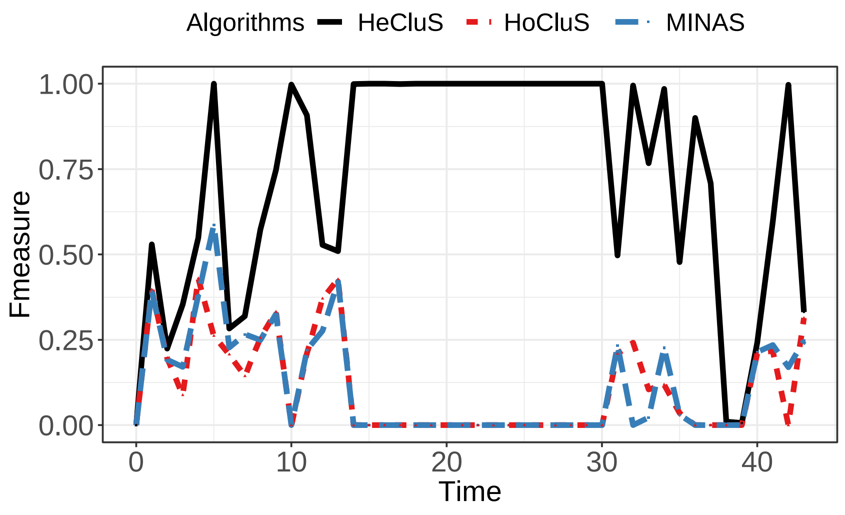


Figure 16 – F-measure of the unsupervised methods in the KDD dataset

In Table 14, we see that HeCluS has higher F-Measure than HoCluS and MINAS, mainly because it correctly classified more data from C1 than the other unsupervised methods. In terms of the supervised methods, SAND has F-Measure 0.88; however, it misclassified some data of C2 as C1 and C1 as a novelty. The CLAM classified C1 correctly; however, it was not able to learn the novelties, since all data from C2 was labelled as unknown. Finally, MINER has the highest score, 0.90.

In Figure 17, we can see that HeCluS had better predictive performance during the entire stream. Also, HeClus had more periods of stability than HoCluS and MINAS. HeCluS was less affected by the appearance of new classes (time 35 and time 75) than the other models. In terms of performance, HoCluS is not different than MINAS.

5.5.4.7 UG_2C_2D dataset

The UG_2C_2D dataset has one *normal* class and one *new* class, both with concept drift and overlap. Analysing the unsupervised methods, Table 15, HeCluS classified all data as C1, which means it did not learn the new class. HoCluS had slightly higher *F-Measure* than

Table 14 – Confusion matrix for MOA dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C 1	0.62	0.25	0.79	0.00	1.00	0.00	0.31	0.00	0.31	0.00	0.60	0.00
C 2	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00
Novelty	0.18	0.75	0.00	0.00	0.00	0.00	0.66	0.99	0.62	0.98	0.40	0.99
Unknown	0.02	0.00	0.16	1.00	0.00	0.01	0.02	0.01	0.04	0.02	0.00	0.01
F-Measure	0.88		0.44		0.99		0.71		0.71		0.86	

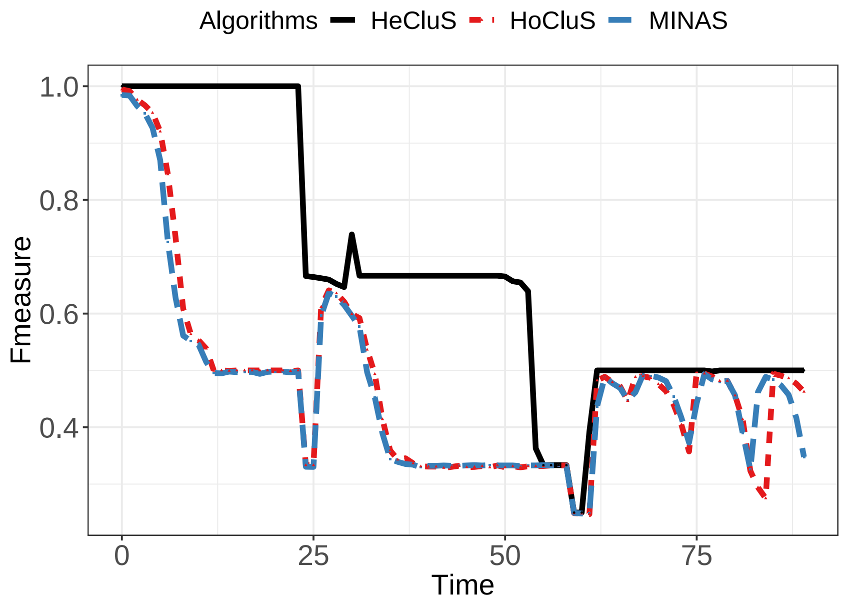


Figure 17 – F-measure of the unsupervised methods in the MOA dataset

MINAS; however, both methods misclassified a great percentage of C2 and C1. Besides that, HoCluS and MINAS learned the new class. For the supervised methods, SAND and MINER also misclassified C2 as C1. MINER also misclassified more C1 as C2 than the other methods, which explain its low F-Measure in comparison with the other supervised methods. CLAM presented the highest score, especially because did not misclassifies C2 as C1.

In Figure 18, we see that the F-Measure of HeClus is constant during all the stream. This dataset has balanced classes, see Table 15, which means that both classes appear in the stream at an equal rate. In the case of the methods HoCluS and MINAS the F-Measure was lower and unstable, especially during the period 0.25 and 0.70. After the period 0.75, HoCluS shows stability and F-Measure of 0.5, while MINAS had a lower score than the others.

Table 15 – Confusion matrix for UG_2C_2D dataset

	Supervised						Unsupervised					
	SAND		CLAM		MINER		MINAS		HoCluS		HeCluS	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C 1	0.71	0.50	1.00	0.02	0.50	0.58	0.42	0.45	0.45	0.50	1.00	1.00
C 2	0.10	0.07	0.00	0.70	0.46	0.37	0.00	0.00	0.00	0.00	0.00	0.00
Novelty	0.28	0.44	0.00	0.00	0.01	0.01	0.51	0.46	0.50	0.43	0.00	0.00
Unknown	0.01	0.06	0.00	0.28	0.03	0.04	0.07	0.09	0.05	0.07	0.00	0.00
F-Measure	0.72		0.91		0.60		0.43		0.45		0.50	

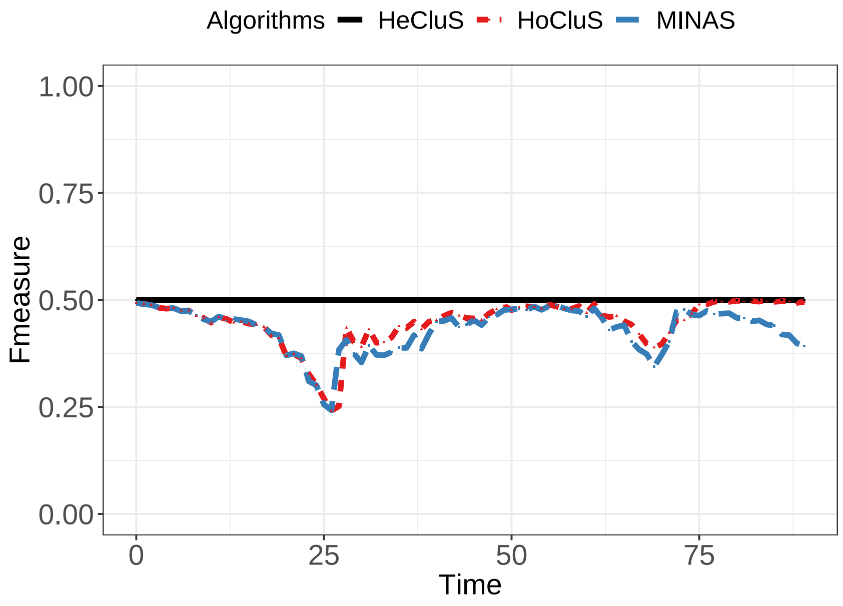


Figure 18 – F-measure of the unsupervised methods in the UG_2C_2D dataset

5.6 Conclusions

In this work, we proposed the methods HeCluS and HoCluS for unsupervised detection of novelties and concept drift in data streams. These ensembles combine several partitions from one or more clustering techniques. This allows the use of clustering techniques with different bias, in order to obtain more robust classification models. When new instances arrive, each clustering technique can independently create and update its partitions. To test the efficiency of the proposed methods, we implemented each method in a single model algorithm for novelty detection in data streams.

In the experiments, we observed that HoCluS and HeCluS are competitive with the other tested methods. In most cases, the ensembles have better performance than the single model MINAS. Observing the performance of the unsupervised methods over time, we conclude that HeCluS is usually more stable and has higher F-Measure than HoCluS and MINAS. This shows that an ensemble built with different clustering techniques can better detect the changes in the data streams. Moreover, the use of the heterogeneous ensemble is a promising strategy, because its bias can be more suitable for a given data stream or for specific periods in the same data stream. The proposed methods have an advantage of not need the labels to update themselves; still, they have a competitive performance with the supervised methods.

The experiments also showed that the performance of all tested methods were affected by the changes in the data streams. Thus, a method can have better performance in periods of the stream. Therefore, as future work, we would like to investigate a mechanism to automatic adapt an algorithm hyperparameters since this could improve predictive performance over time.

CHAPTER

6

AN ENSEMBLE OF AUTONOMOUS AUTO-ENCODERS FOR HUMAN ACTIVITY RECOGNITION

Collaborating authors

Cláudio Rebelo de Sá
University of Twente, Enschede, The Netherlands

Mannes Poel
University of Twente, Enschede, The Netherlands

Tiago Carvalho
University of Porto, Porto, Portugal

João Mendes-Moreira
LIAAD-INESC TEC, University of Porto, Porto, Portugal

João M.P. Cardoso
University of Porto, Porto, Portugal

André C. P. L. F. de Carvalho
University of São Paulo, São Carlos, Brazil

Joost N. Kok
University of Twente, Enschede, The Netherlands

Abstract

Human Activity Recognition is focused on the use of sensing technology to classify human activities and to infer human behavior. While traditional machine learning approaches use hand-crafted features to train their models, recent advancements in neural networks allow for automatic feature extraction. Auto-encoders are a type of neural network that can learn complex representations of the data and are commonly used for anomaly detection. In this work we propose a novel multi-class algorithm which consists of an ensemble of auto-encoders where each auto-encoder is associated with a unique class. We compared the proposed approach with other state-of-the-art approaches in the context of human activity recognition. Experimental results show that ensembles of auto-encoders can be efficient, robust and competitive. Moreover, this modular classifier structure allows for more flexible models. For example, the extension of the number of classes, by the inclusion of new auto-encoders, without the necessity to retrain the whole model.

6.1 Introduction

Human Activity Recognition (HAR) is a research field focused on the use of sensing technology to classify human activities and to infer human behaviour (DOBBINS; RAWASSIZADEH; MOMENI, 2017). A HAR system can use data from different sources, like wearables, sensors from objects and cameras. These systems have been successfully applied for health and well-being (BAÑOS *et al.*, 2014b), tracking and mobile security (SPINSANTE *et al.*, 2016) and elderly care (YAO *et al.*, 2017).

Most HAR machine learning approaches found in the literature, such as: decision trees (LARA; LABRADOR, 2013), support vector machines (MANNINI *et al.*, 2013) and k-Nearest Neighbor (YAO *et al.*, 2017) rely on the use of heuristic hand-crafted feature extraction to train their models. That includes, for example, time-domain calculations, mean and standard deviation for each sensor signal and correlation (Pearson correlation) between axes for the 3D sensors.

In our previous work (GARCIA *et al.*, 2019b) we studied a semi-supervised ensemble, EkVN, which combined 3 different algorithms (k-Nearest Neighbour, Very Fast Decision Tree and Naive Bayes). This method relies on heuristic hand-crafted feature extraction for HAR. The features were extracted from the raw data of different types of sensors: accelerometer, gyroscope and magnetometer sensors. We investigated the impact of some hyperparameters in the accuracy of EkVN. We found that the accuracy of EkVN is more sensitive to data from different users, to the window size and to the overlapping factor. We also found that the feature extraction process has a relatively high energy and time costs. This can have implications, for example in mobile applications, where the use of resources must be carefully managed in order to keep the application efficiently working for long periods of time.

An alternative to the manual extraction of features is the automatic feature extraction with neural networks (PLÖTZ; HAMMERLA; OLIVIER, 2011). One type of neural network commonly used as a powerful tool for discovery of features is the Auto-Encoder (AE). This type of neural network tries to learn two functions, an encoder, which maps the input to the hidden layers (the bottleneck), and a decoder, which maps the hidden layers to the output layer. In other words, an AE can learn compact representations of the input data in an unsupervised manner (WANG *et al.*, 2019). Therefore, the output of an auto-encoder is the reconstruction of its input.

In this work, an extension of (GARCIA *et al.*, 2019b), we propose a classification approach which is an Ensemble of AEs (EAE). In this EAE each AE is trained with data from one class [1]. Thus, in the context of HAR, each AE is associated with a label/activity. As new data arrives for classification, the reconstruction loss is calculated for each AE. The data is then classified with the label from the AE which obtained the lowest reconstruction loss. When used in online learning, the ensemble model can be updated with the user's data when the reconstruction loss drops below a given threshold. To the best of our knowledge there are no approaches that use AEs as an ensemble classifier.

We tested two variants of EAE in HAR data, an online and an offline one. Both variants learn from the same train data, however the first also learns incrementally when the loss increases more than an user-defined threshold. Experimental results show that the EAEs are efficient, robust and competitive with state-of-the-art approaches.

This paper is structured as follows. Section 6.2 presents the related work on machine learning for HAR. Section 6.3 describes the method proposed in this study. The results obtained are presented and discussed in Section 6.4. Finally, Section 6.5 summarises the main conclusions and points out future work directions.

6.2 Related Work

The main goal of HAR is to recognize human physical activities from sensing data. In this research area many approaches were presented in the last decade (NIAZI *et al.*, 2017; ZHENG *et al.*, 2016; ZOU *et al.*, 2018; SEYFIOGLU; ÖZBAYOGLU; GURBUZ, 2018). These approaches vary depending on the sensor technologies used to collect the data, the machine learning algorithm and the features created to train the model. In relation to extraction and selection of features, the models can be trained using hand-crafted feature extraction or automatic feature extraction.

The conventional approaches in HAR use hand-crafted feature extraction, which means that these approaches rely on human domain knowledge. Those features often include statistical information, such as: mean, variance, standard deviation, frequency and Pearson Correlation (FIGO *et al.*, 2010). These approaches use traditional machine learning methods such

[1] This type of ensemble might also be known as Mix of Experts (MAKKUVA *et al.*, 2019).

as: SVM classifiers, *k*-Nearest Neighbour, decision tree, Naive Bayes classifiers, Random Forest (MANNINI *et al.*, 2013; BEDOGNI; FELICE; BONONI, 2012; LARA; LABRADOR, 2013; DOBBINS; RAWASSIZADEH; MOMENI, 2017). Others focus on the combination of these machine learning approaches as ensembles in order to improve accuracy (GARCIA *et al.*, 2019b; DOBBINS; RAWASSIZADEH; MOMENI, 2017). It is generally known that ensembles with bagging and boosting techniques can increase the performance of classifiers (DIETTERICH, 2000). In most of these studies the improvements proposed are more focused in the tuning of hyperparamenters that are common in HAR (e.g. window size and overlapping factor) (BAÑOS *et al.*, 2014b) and feature construction (DOBBINS; RAWASSIZADEH; MOMENI, 2017).

In contrast to that, Neural Networks methods (a.k.a Deep Learning) have the capacity to automatically learn relevant features from raw data without human domain knowledge (WANG *et al.*, 2019). Many different deep learning architectures have been proposed, such as Convolutional Neural Networks (CNN) (ZHENG *et al.*, 2016; SEYFIOGLU; ÖZBAYOGLU; GURBUZ, 2018; PANWAR *et al.*, 2017), recurrent neural networks (ZOU *et al.*, 2018) and AEs (WANG, 2016; GAO *et al.*, 2019; VINCENT *et al.*, 2010).

Mostly used for computer vision, CNN models have also demonstrated to be effective in natural language processing (KIM, 2014), speech recognition (ABDEL-HAMID *et al.*, 2012) and text analysis (SANTOS; GATTI, 2014). In terms of HAR, CNNs have also been used to extract features from sensing data and to classification tasks (WANG *et al.*, 2019). Approaches for HAR based on CNNs can learn the correlation between nearby signals and be scale-invariant for different frequencies (WANG *et al.*, 2019; PANWAR *et al.*, 2017). Some of these approaches process each dimension of a signal (e.g. a 3D accelerometer signal) as a channel. In other words, that means that to each channel is applied a 1D convolution. After that, the outputs from all channels are flattened to unified layers. Chen and Xue (CHEN; XUE, 2015) used a CNN model with a modified convolution kernel to adapt to the characteristics of 3D signals. On the other hand, 2D convolutions can present better results compared to 1D convolutions. Ha and Choi (HA; CHOI, 2016) proposed 2D convolutions where CNNs were used with partial and full weight sharing structures to investigate the performance of different weight-sharing techniques. Weight-sharing is a technique used to incorporate invariance, to reduce complexity and to speed up the training process of CNNs (WANG *et al.*, 2019). To use 2D convolutions, some approaches resize the inputs from the signals as virtual 2D images (HA; YUN; CHOI, 2015).To learn the dependencies between signals they applied a CNN using a 2D convolution kernel and a 2D pooling kernel. Following this idea Jiang and Yin (JIANG; YIN, 2015) designed a more complex process to transform the signals into 2D image description and applied 2D convolution to extract features.

AEs are one family of neural networks which can learn a compact representation of the input signals. Stacked Auto-Encoder (SAE), for example, stack the learned features which can later be used to build a classification model (WANG *et al.*, 2019). Wang et al. (WANG, 2016) proposed a Continuous AE that converts high-dimensional continuous data to low-dimensional data in the encoding process. The features are extracted by AEs with multiple hidden layers. Gao et al. (GAO *et al.*, 2019) proposed the combination of Stacking Denoising AE for feature extraction with LightGBM as the classifier. Ensemble of AEs can also be used for unsupervised outlier detection. For example, Chen et al. (CHEN *et al.*, 2017) proposed an ensemble of AEs randomly connected with different structures and connection densities, which reduces computational costs. The outliers are detected by computing the median of the AEs reconstruction error. In HAR, the features learned by Denoising Stacked AEs can be used by a random forest algorithm to build an ensemble classifier (THOMAS; BOUROBOU; LI, 2018).

6.3 Methodology

In this section we start by describing the E*k*VN method with the hand-crafted feature extraction, presented in (GARCIA *et al.*, 2019b). Afterwards, we describe the proposed method, the Ensemble of Auto-Encoders (EAE). Both methods are semi-supervised learning approaches. This means that they are incrementally updated after the data from a specific user is classified.

6.3.1 Ensemble of kVN

The E*k*VN is an ensemble model composed by three classifiers: *k*NN, Very Fast Decision Tree (VFDT) and Naive Bayes. The implementation of the ensemble classifier is the combination of Democratic Co-Learning and Tri-Training (ZHOU; LI, 2005). This method uses a vector of hand-crafted features as input, both in its training and test phase, as illustrated in Figure 19.

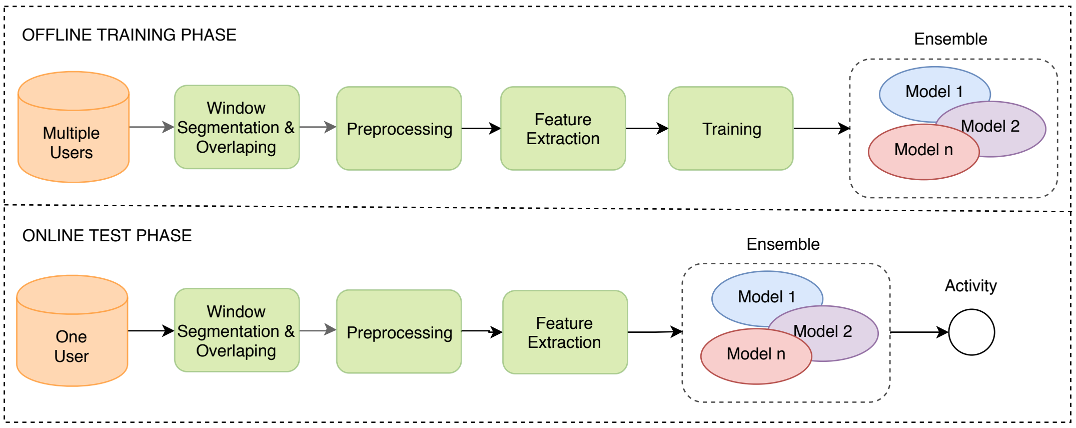


Figure 19 – Overview of the ensemble model, E*k*VN, for HAR.

The top pipeline in Figure 19 shows the offline training using raw data extracted from different wearables and/or smartphone sensors. In the first step, *window segmentation & overlapping*, the raw data is stored in sliding windows and consecutive windows are overlapped. The window size (w) and overlap factor (ovl) are user defined values.

Data from sensors is usually susceptible to noise, especially accelerometer data (DOBBINS; RAWASSIZADEH; MOMENI, 2017). Thus, the *preprocessing* step is important for calibration and filtering of the input data in order to reduce the noise. After that, a new instance is created containing the features that will be used to train the model, the *feature extraction* step. These features include time-domain calculations, specifically the mean, the standard deviation and the Pearson Correlation of each axis for the 3D sensors. Afterwards these instances are used to train (*training* step) one model from each one of the algorithms: *k*NN, VFDT and Naive Bayes. Then, they are combined as an ensemble of models.

In the online phase, new data is collected from a specific user. This data is preprocessed as described in the steps from the training phase: *window segmentation & overlapping*, *preprocessing* and *feature extraction*. Each new instance is classified by the ensemble, which provides a confidence factor for the classification. The instances classified with high confidence, more than 99%, are used to update the model.

6.3.2 Ensemble Of Auto-Encoders

A basic AE is a neural network model in which the output replicates the input, $y^i = x^i$ (WANG, 2016). An AE consists of two parts, an Encoder and a Decoder, Figure 20. The encoder learns to compress the inputs into a smaller number of encoded features, which is called the bottleneck. Given the encoded features, the decoder learns how to reconstruct the original input. Therefore, the output of an AE is an approximate reconstruction of the input (ZOU *et al.*, 2018).

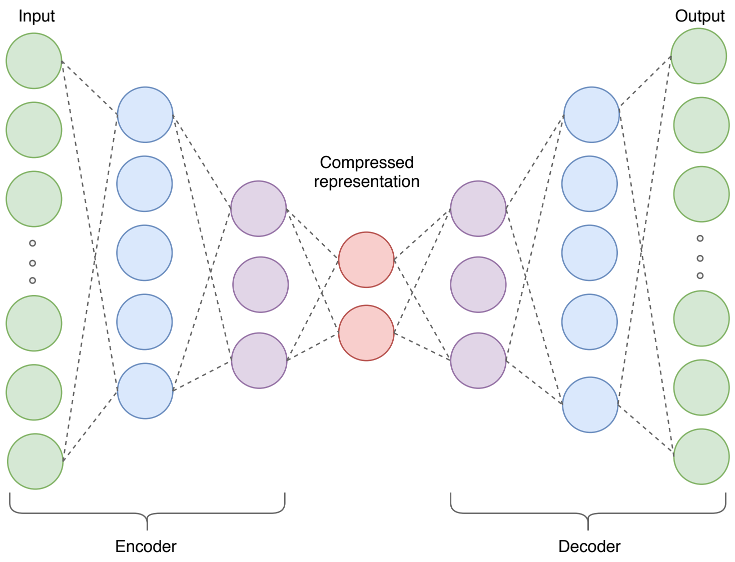


Figure 20 – An auto-encoder fully-connected structure with 3 hidden layers.

In this work, we propose to use a set of AE, as an ensemble, for classification. The code is available on GitHub [2]. Figure 21 illustrates the steps for training the Ensemble of Auto-Encoders

[2] <https://github.com/Keh/EAE.git>

(EAE) (offline phase) and how it can be used for classification (online phase). In the offline phase a batch of data from multiple users is used to train one AE per class. Thus, each AE learns a different activity.

In the online phase, as new data arrives, each AE tries to reconstruct the original input. Then the AE with the smallest reconstruction error, *minError*, is selected. The data is then classified with the label corresponding to the AE with the *minError*. During this online phase, each AE is updated whenever the reconstruction error falls below a user-defined threshold, T. By default, we define this threshold as X standard deviations of the training error. The threshold is a hyperparameter to set a high confidence factor, as in the method explained in Section 6.3.1. In both offline and online phases, the raw data is segmented according to a user defined window size (w) and an overlapping factor (ovl).

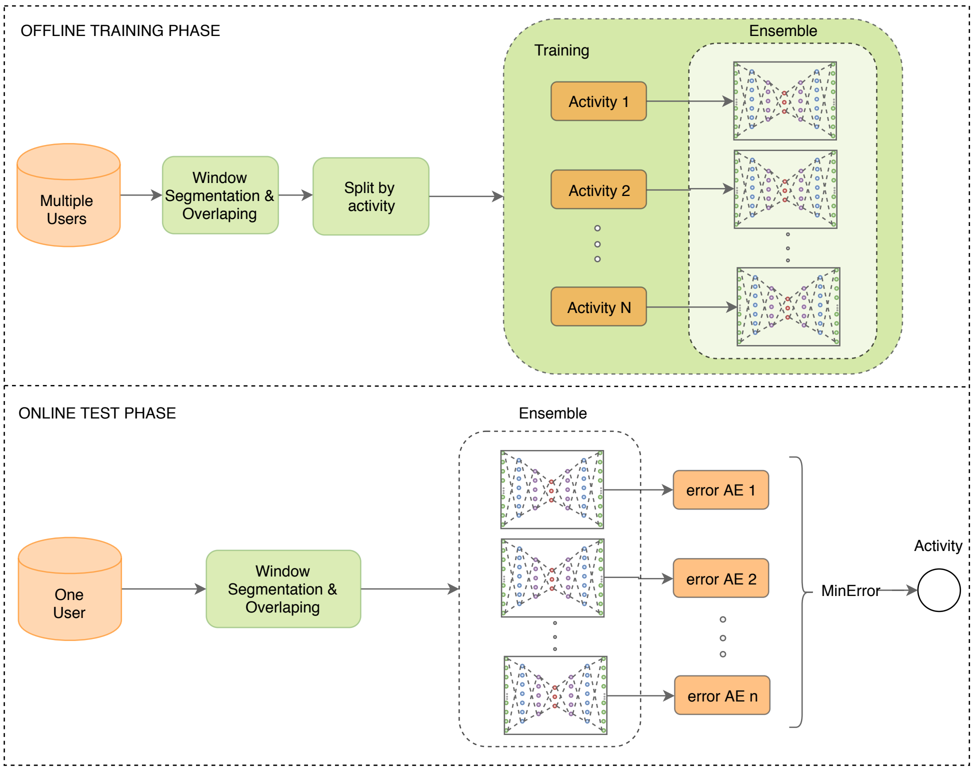


Figure 21 – Overview of the ensemble model, EAE, for HAR.

To illustrate how the EAE works, we present in Figure 22 a simple example. The red line represents the real signal (used as the input data) and the blue line depicts the reconstructed signal by each AE. In this example, the model is composed by 6 AEs, where each was trained with data from one of the following activities: *Walking Downstairs*, *Jogging*, *Sitting*, *Standing*, *Walking Upstairs* and *Walking*. In Figure 22 one can see that the AE which better reconstructs the signal is the *Sitting* AE. Therefore, the model classifies this activity as *Sitting*. Finally, if its error is below the defined threshold T, the AE is updated with this new signal.

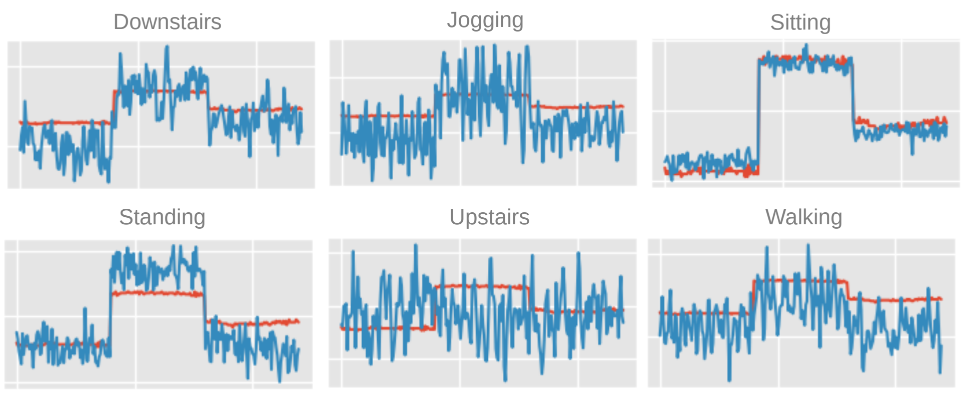


Figure 22 – The reconstruction of the signal by each AE. The color *red* represents the original signal and the color *blue* represents the reconstructed signal.

6.4 Experiments

We conducted several experiments to compare the predictive performance of the EAE, with the E*k*VN and 5 other deep learning approaches. These methods are briefly described in Section 6.2 and as in (JORDAO *et al.*, 2018) will be referred by the name of their author as: ChenXue (CHEN; XUE, 2015), HaChoi (HA; CHOI, 2016), Haetal (HA; YUN; CHOI, 2015), JiangYin (JIANG; YIN, 2015), Panwaretal (PANWAR *et al.*, 2017). The performance of all the methods was tested in 3 datasets commonly used in the literature of HAR.

6.4.1 Datasets

In Table 16 we can see some statistics with a brief description of each dataset. Figure 23 illustrates the frequency of activities that each dataset has. They all include standard activities, such us: *Walking*, *Jogging/Running*, *Standing*, *Sitting* and *Climbing Stairs*. The activities can be divided in Static, such us *Standing* and *Sitting*, and Dynamic, such us *Walking* and *Jogging*. The datasets MHealth and PAMAP2 also have more complex activities, such us house cleaning or sports. By complex activities we refer to activities that can be decomposed into others activities. For example, *Vacuum Cleaning* can be decomposed in: *Standing*, *Walking* and *Bending Forward*.

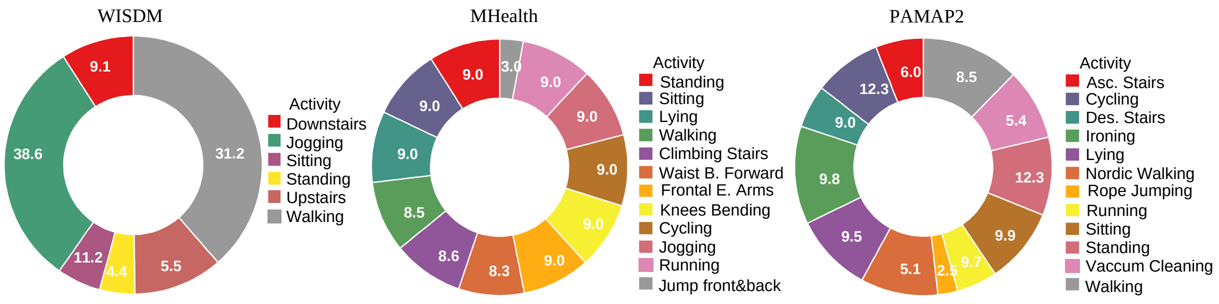


Figure 23 – Frequency per class of each dataset.

The *WISDM* dataset (KWAPISZ; WEISS; MOORE, 2010) contains sensor data from phone-based accelerometers [3]. The data was collected by an application installed on each user‘s phone. It has 1.098.209 records, of a 3-axis accelerometer sensor, from 29 users carrying a smart-phone placed on their front pants‘ pocket. In this dataset, there is no information about the age, gender or physical/behaviour characteristics of the users. The data was collected at $20Hz$ samples per second. The distribution of the classes can be seen in Figure 23. The most common activities are: *Jogging* and *Walking*.

The *MHealth* dataset (BAÑOS *et al.*, 2014c) has sensor data from 10 users performing 12 activities [4]. The data was collected from 3 devices with the following embedded sensors: a 3-axis accelerometer, a 3-axis gyroscope, a 3-axis magnetometer and an electrocardiogram sensor. These sensors were placed on different body locations, such as, chest, hand and ankle. There is also no personal information about the users. This dataset has 1.215.745 instances in total and has reasonably well balanced classes. The class with less data is *Jump front & Back*.

Finally, the *PAMAP2* dataset (REISS; STRICKER, 2012) is a public dataset of human physical activities [5]. The data was collected from 3 devices positioned in different body locations: hand, chest and ankle. Each device has three embedded sensors: a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. This dataset contains 1.926.896 samples of raw sensor data from 9 different users and 18 activities. The authors divided these activities in: basic activities (*Walking* and *Running*), posture activities (*Lying* and *Standing*) and house cleaning (*Ironing* and *Vacuum Cleaning*). Also, part of the users performed optional activities, such us *Rope Jumping*.

Table 16 – Details of the 3 HAR datasets used in this work (A=accelerometer, G=gyroscope, M=magnetometer, C=electrocardiograph).

Dataset	#Users	S. Rate	#Activity	#Samples	Sensors	body location
WISDM	36	20 Hz	6	1.098.209	A	front pants‘ pocket
MHealth	10	50 Hz	12	1.215.745	A,G,M,C	Chest,Hand,Ankle
PAMAP2	9	100 Hz	18	1.926.896	A,G,M	Chest,Hand,Ankle

6.4.2 Experimental setup

We analysed the performance of all the tested methods in terms of accuracy and computational cost. The latter was measured in seconds both in training and testing. For a fair comparison between the models, we only used the accelerometer data from each dataset. We trained the models with a fixed window size, w, of 160. Although other alternatives can be considered for the choice of the window size, for example, dynamic window size (MA *et al.*, 2020), it would require additional steps such as compression or concept drift detectors, which would increase

[3] <http://www.cis.fordham.edu/wisdm/dataset.php>
[4] <http://archive.ics.uci.edu/ml/datasets/mhealth+dataset>
[5] <http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring>

the computational cost. Therefore, for simplicity, we used a fixed window $w = 160$. In practical terms, this represents 8.0 seconds for WISDM (20 Hz), 3.2 seconds for MHealth (50 Hz) and 1.6 seconds for PAMAP2 (100 Hz). Each consecutive window is overlapped with ovl, overlap factor, of 20% (GARCIA *et al.*, 2019b). For the evaluation we used the leave-one-user-out approach.

We note that in the case of the proposed EAE the input is a vector with 480 entries. Which consists of the 3 components of the accelerometer sensor: x-acceleration, y-acceleration, and z-acceleration.

For the EkVN, we created the features: mean, standard deviation and Pearson Correlation. We also use the confidence factor of 99% for updating the model.

In the EAE method, each AE is composed of 8 hidden layers. The encoder has one input layer with 480 nodes, and 4 hidden layers with respectively, 200, 100, 80 and 32 nodes. The decoder has the opposite structure: 4 hidden layers of 32, 80, 100 and 200 nodes and with an output layer of 480. The first and the second layers of the AE have the *ReLU* activation function and the third and last layer a *linear* activation function. Each AE was trained for 250 epochs with a shuffled batch of size 256. The loss function used was the MAE and the optimiser was the adaptive moment estimation (Adam). In the online phase, the AE is updated when the *minError* (Section 6.3) is less $minError \leq threshold, T$, where $T = 0.01$.

To train the methods ChenXue, HaChoi, Haetal, JiangYin, Panwaretal, we used the same configuration proposed by the authors. The only difference is that we use the same window size $w = 160$, for a fair comparison with the other methods.

For each dataset, we analyse the mean and the dispersion of the accuracy per model. For that, in the first analysis, we are including all the experiments in which the data was divided by user. In the second analysis we focus in the accuracy of the models per user. Due to space constrains, we only present the results from the experiments with data collected from accelerometers placed in one body location, the hand or the pocket. Finally, we measure how the accuracy varies with the data collected from an accelerometer placed on different body locations. In this third analysis we present the average accuracy of the models per body location.

6.4.3 Results and Discussion

In this section, we present and discuss the main results from the experiments. We note that in the experiments that includes the deep learning models, we present the results of our method with incremental learning, called EAE, and the model without incremental learning, called EAE_Off. We present both versions so we can analyse the improvement of online model update and also for a fair comparison with the other deep learning models that are not updated online.

6.4.3.1 WISDM Dataset

Considering the WISDM dataset, we can observe a plot containing 8 violin box-plots representing the variation/dispersion of the accuracy per model Figure 24. In this graph, each model is represented by a different colour. Each box-plot has the results of all the experiments concerning the accuracy of each model per user.

In Figure 24 we notice that EAE has less dispersion in accuracy than the other models. The median of the accuracy is around 87% for EAE, while for E*k*VN it is around 80%. As for the other models, the median accuracy is around 87%, however their variance is larger than the variance of EAE. As for the lowest accuracy, it can reach in some cases, less than 25%.

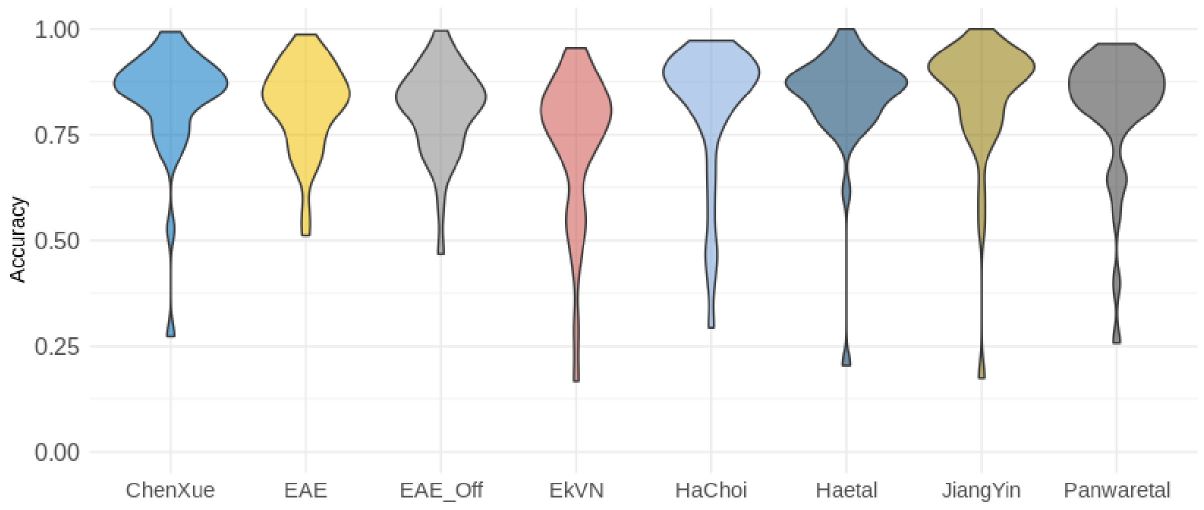


Figure 24 – Violin box-plot showing the dispersion of accuracy of the models in the dataset WISDM.

We can see in Table 17 that the deep learning models have similar average accuracy, however the models Haetal and JiangYin show slightly better results. The average accuracy of EAE and E*k*VN models are 0.82 and 0.73, respectively. In terms of computational cost, we see that the model JiangYin took more time to train than the other methods. The EAE model has an average training time similar to the HaChoi, Haetal and Panwaretal models. The models E*k*VN model and ChenXue have the lowest training time. The ChenXue model has the simplest deep learning architecture, so it is reasonable that its time consumption is lower than the others. In terms of the testing computational cost, the EAE has the highest one. This can be due to the number of AEs and also the incremental learning step. Overall, the results show that both the EAE are learning meaningful representations of the activities in a reasonable time. However, the time for prediction is superior due to the number of AE models and its incremental learning.

In Figure 25 we see the accuracy per user for the models EAE and the E*k*VN. The EAE obtained a higher accuracy in 78% of the users as compared with E*k*VN. One of the most striking differences is in user 30, where the accuracy of the EAE model is 71% while the accuracy of E*k*VN model was only 16.7%. As mentioned before, we do not have demographic information

Table 17 – Average Accuracy(Acc) and Time (Train/Test) for the models in the WISDM dataset.

	ChenXue	HaChoi	Haetal	JiangYin	Panwaretal	EAE	EAE_Off	EkVN
Acc	0.83	0.81	**0.84**	**0.84**	0.81	0.82	0.81	0.73
Time (s)	**65/0.1**	104/0.1	196/0.1	430/0.1	109/0.1	172/20.0	172/14.0	50/0.2

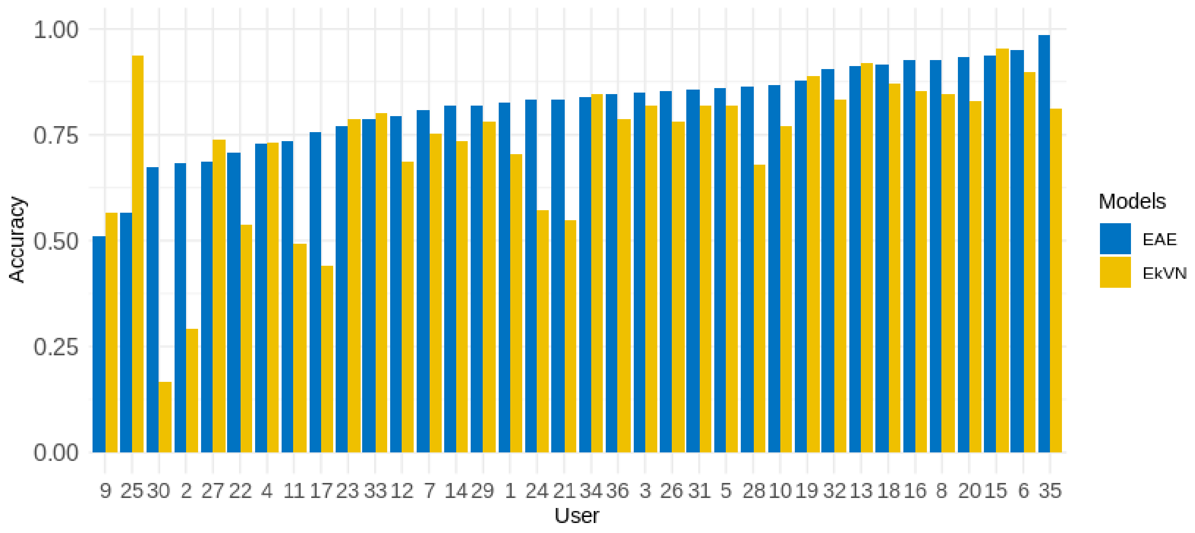


Figure 25 – Accuracy per user for the EAE and the E*k*VN models for the WISDM dataset considering the body location *Pocket*.

about the users, however we observe that the misclassification between *Walking* and *Jogging* was more evident in some users than others. Since the difference between the activities is in the intensity of the movement, it could have been useful to compare physical characteristics of the users with the classification.

When looking at the confusion matrix of the EAE (Table 18), we can observe that the classes with higher misclassification are *Downstairs* and *Upstairs*. They are often misclassified with each other or with *Walking*. One difference between the classes *Downstairs* and *Upstairs* is the orientation of the activity: one is descending stairs and the other is ascending stairs. This concept might be hard to learn only from accelerometer data, since this sensor does not capture the orientation of the movement. On top of that, we also notice that the AEs *Downstairs* and *Upstairs* were trained with less data than others classes (Figure 23) which makes it even more difficult for the models to learn them.

6.4.3.2 MHealth Dataset

In Figure 26, we can observe the dispersion of accuracy obtained by the models in the MHealth dataset. The median of accuracy of the EAE model is above 90%. We note that the EAE has less variance than EAE_Off, meaning that the incremental learning reduces variance. The models ChenXue, HaChoi and JiangYin had higher variance than the other models. Although the lowest variance is E*k*VN, its median accuracy of 75%.

Table 18 – Average confusion matrix for the EAE model in the WISDM dataset considering the body location *Pocket*. The columns represent the ground truth and rows represent the predicted.

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	**0.40**	0.06	0.01	0.00	0.22	0.30
Jogging	0.01	**0.96**	0.00	0.00	0.01	0.02
Sitting	0.02	0.03	**0.91**	0.04	0.01	0.00
Standing	0.01	0.00	0.04	**0.95**	0.00	0.00
Upstairs	0.06	0.11	0.00	0.00	**0.58**	0.24
Walking	0.09	0.00	0.00	0.00	0.04	**0.87**

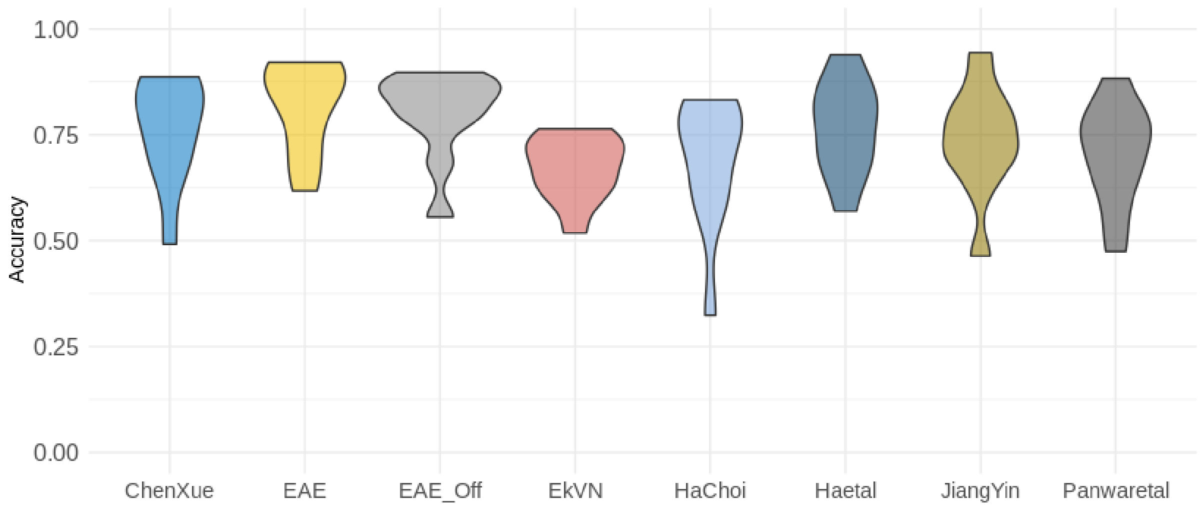


Figure 26 – Violin box-plot showing the dispersion of accuracy of the models in the dataset MHealth.

Table 19 – Average Accuracy(Acc) and Time (Train/Test) for the models in the MHealth dataset.

	ChenXue	HaChoi	Haetal	JiangYin	Panwaretal	EAE	EAE_Off	EkVN
Acc	0.76	0.69	0.77	0.74	0.70	**0.82**	0.75	0.67
Time (s)	65.1/0.1	**49.4/0.1**	385.1/0.4	83.3/0.1	197.0/0.1	209.9/36	209.9/31	79.3/1.1

For this experiment we consider only data from the body location *hand*. In terms of average accuracy and time consumption, we can see in Table 19 that both the EAE and the EAE_-Off are competitive results with the deep learning models. However, because of the incremental learning of the EAE it obtained an even higher average accuracy than other models. On the other hand, the only model that uses hand-crafted features, E*k*VN, had the lowest accuracy.

In terms of time consumption, one more, the models with simpler architectures are faster to train (HaChoi and ChenXue). The EAE takes more time in the prediction phase, specially because this phase includes the incremental learning of the model. Considering that this is an ensemble, the amount of models influences on the time consumption of its testing phase.

When comparing the accuracy per user of the models EAE and E*k*VN (Figure 27) we can observe that the EAE was better for all individuals. This shows, once again, that the proposed

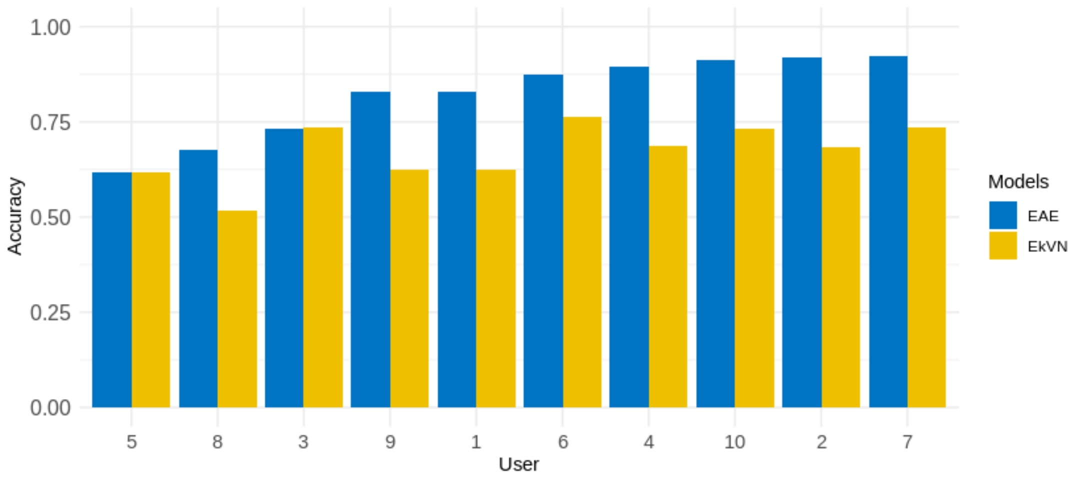


Figure 27 – Accuracy per user for the EAE and the E*k*VN models in the MHealth datasets, considering the body location *Hand*.

method can learn meaningful representations of the activities.

By looking at the confusion matrix of the EAE model (Table 20) we see that the class *Stairs* has an average accuracy of 93%. This class has data from *Downstairs* and *Upstairs* combined. This shows that the model can learn better from the classes which are independent of the orientation. The class *Running* was more misclassified as *Jogging* than the other way around, which might be related to the pace that each individual takes to perform these activities.

Table 20 – Average confusion matrix for EAE model for MHealth dataset considering only the body location *Hand*. The columns represent the ground truth and rows represent the predicted.

	1	2	3	4	5	6	7	8	9	10	11	12
1 - Stand	**0.89**	0.04	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2 - Sit	0.11	**0.45**	0.22	0.11	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00
3 - Lay	0.00	0.11	**0.78**	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00
4 - Walk	0.01	0.00	0.00	**0.95**	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5 - Stairs	0.00	0.01	0.00	0.02	**0.93**	0.02	0.01	0.02	0.00	0.00	0.00	0.00
6 - Waist Bend	0.01	0.01	0.00	0.01	0.02	**0.74**	0.00	0.22	0.00	0.00	0.00	0.00
7 - Elevation	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
8 - Knees Bend	0.00	0.01	0.00	0.04	0.06	0.23	0.00	**0.66**	0.00	0.00	0.00	0.00
9 - Cycle	0.00	0.19	0.00	0.00	0.00	0.00	0.00	0.00	**0.81**	0.00	0.00	0.00
10 - Jog	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.92**	0.08	0.00
11 - Run	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.27	**0.73**	0.00
12 - Jump	0.00	0.00	0.02	0.06	0.02	0.00	0.00	0.00	0.00	0.02	0.00	**0.88**

6.4.3.3 PAMAP2 dataset

For the PAMAP2 dataset, we see in Figure 28 a high dispersion in the accuracy of the models, specially for ChenXue, HaChoi and Panwaretal. Although E*k*VN also has a high dispersion of the accuracy, it is the model with the highest median accuracy, around 70%. The

Table 21 – Average Accuracy(Acc) and Time (Train/Test) for the models in the PAMAP2 dataset.

	ChenXue	HaChoi	Haetal	JiangYin	Panwaretal	EAE	EAE_Off	EkVN
Acc	0.63	0.57	0.70	0.64	0.63	0.63	0.62	**0.71**
Time (s)	**79.3**/0.4	654.1/**0.3**	563.7/0.5	347.2/**0.3**	960.1/**0.3**	249.4/38.2	249.4/31	130.4/2.0

model EAE has the median of accuracy slightly above 60%, presenting a small improvement compared with the offline variant, EAE_Off.

The maximum accuracy reached by EAE is 0.91, which is the highest of the deep learning methods. We also observed that the minimum accuracy of the EAE is always the highest in all the datasets tested. In this case, the lower value is 0.44 which is the same as for E*k*VN.

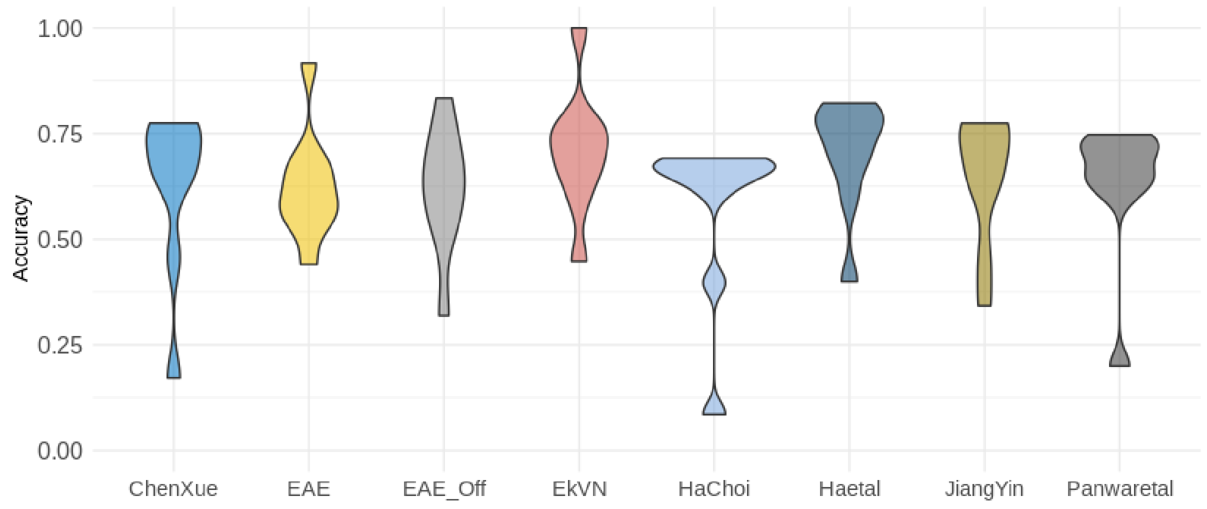


Figure 28 – Violin box-plot showing the dispersion of accuracy of the models in the dataset PAMAP2.

In Table 21 we see that the average accuracy of the E*k*VN the highest, meaning that traditional models can achieve better performance in some datasets. Haetal is the deep learning model with the highest average accuracy. All the others, JiangYin, Panwaretal, ChenXue, EAE, EAE_Off and HaChoi obtained very similar average accuracy.

In terms of time consumption, we see that ChenXue has a faster training time, however the JiangYin and Panwaretal are faster for testing. We notice that, although EAE is an ensemble of 12 AEs the time of training is not higher that some other deep learning models (e.g. Panwaretal). However the EAE model is the slowest in testing time. The time performance depends on the complexity of the models, the amount of models and the amount of data. Therefore it is natural that EAE shows a higher consumption time.

Considering the results per user of the models EAE and E*k*VN (Figure 29), we see that the accuracy of E*k*VN was slightly higher for all individuals. However, in this dataset, the analysis per user is not an easy task because the users did not perform the activities in equal proportion. For example, the user 9 only performed the activity *Jumping*. This is reflected in

Figure 23 where we can see that there is less data for some classes. This less amount of data has obvious implications in the deep learning methods which are known to require more data than classical approaches. This is more evident for the activities *Ascending Stairs*, *Nordic Walking* and *Rope Jumping*. This dataset also have activities like *Ironing* and *Vacuum Cleaning* which are a mix of activities, such us, *Walking* and *Standing*.

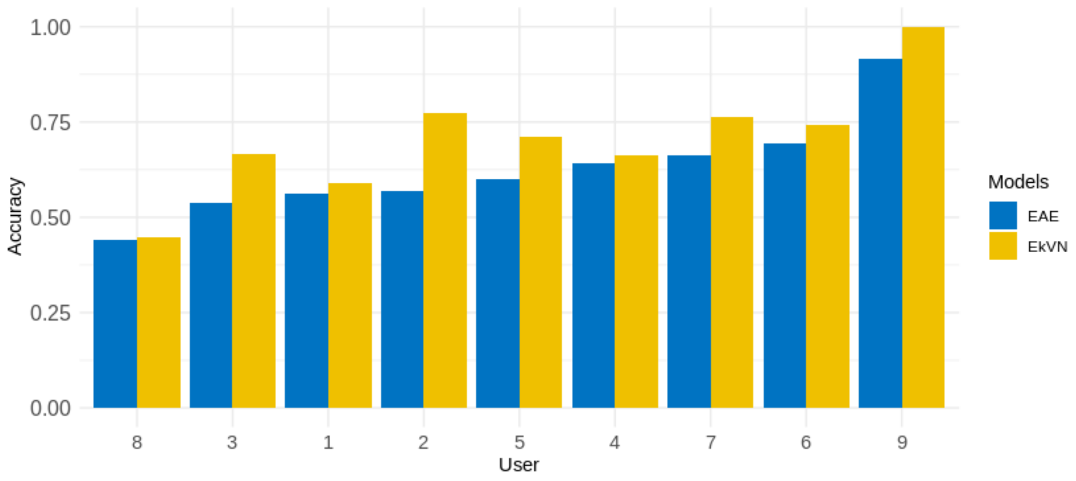


Figure 29 – Accuracy per user for the EAE and the E*k*VN models in the PAMAP2 datasets, considering the body location *Hand*.

The average Confusion Matrix of PAMAP2 (Table 22) shows that the misclassification occurs between classes that are not related. For example, *Descending Stairs* and *Ironing*. From this, we can conclude that the EAE model did not learn the activities as good as in the other datasets. The reason for that might be because this dataset was collected with a frequency of 100Hz (see Table 16). Because of the high frequency, the window size of 480 data points (160 data points per accelerometer axis) represents only 1.6 seconds for each activity, which is not sufficient to learn meaningful representations of each activity. Thus, a bigger window should have been used for this dataset.

6.4.4 Accuracy per body location

In Table 23, we present the average accuracy per body location of the models EAE and E*k*VN, considering WISDM, MHealth and PAMAP2 dataset. For the WIDSM dataset, that only have the body location Front Pocket, the average accuracy of EAE is higher than the E*k*VN model. In terms of the sensors on different body locations for the dataset MHealth, the average accuracy of the EAE for each position is higher than the E*k*VN. Moreover, we can see that the accuracy of the EAE models is practically the same across the different body locations, while the E*k*VN varies. Additionally, as expected, the combination of all the sensors placed on different body locations (HCA) improved the results of both models. For the dataset PAMAP2 we see that the average accuracy of the E*k*VN for each body location is higher than the EAE model. This is

Table 22 – Average confusion matrix for EAE model for PAMAP2 dataset considering only the body location *Hand*. The columns represent the ground truth and rows represent the predicted.

	1	2	3	4	5	6	7	8	9	10	11	12
1 - Asc. Stairs	**0.55**	0.00	0.10	0.05	0.01	0.01	0.00	0.00	0.03	0.03	0.09	0.13
2 - Cycle	0.01	**0.57**	0.01	0.12	0.02	0.00	0.00	0.00	0.13	0.08	0.05	0.01
3 - Des. Stairs	0.16	0.00	**0.42**	0.21	0.01	0.00	0.00	0.00	0.03	0.05	0.06	0.05
4 - Ironing	0.02	0.01	0.04	**0.60**	0.02	0.01	0.00	0.00	0.14	0.07	0.09	0.01
5 - Lay	0.01	0.00	0.01	0.06	**0.60**	0.00	0.00	0.00	0.28	0.03	0.01	0.00
6 - Nord. Walk	0.16	0.00	0.02	0.08	0.00	**0.40**	0.01	0.00	0.09	0.01	0.21	0.02
7 - Jump	0.01	0.00	0.01	0.09	0.00	0.01	**0.60**	0.01	0.07	0.02	0.19	0.00
8 - Run	0.01	0.00	0.00	0.02	0.01	0.00	0.00	**0.76**	0.01	0.02	0.17	0.00
9 - Sit	0.02	0.00	0.01	0.09	0.03	0.01	0.00	0.00	**0.73**	0.10	0.02	0.00
10 - Stand	0.06	0.00	0.03	0.06	0.07	0.00	0.00	0.00	0.12	**0.61**	0.03	0.01
11 - Vac. Clean	0.10	0.00	0.02	0.24	0.01	0.01	0.00	0.00	0.03	0.03	**0.52**	0.03
12 - Walk	0.21	0.00	0.11	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.06	**0.59**

specially evident in HCA. However the EAE model has a lower variance, since all values are around 60.0%.

Table 23 – Average accuracy for the EAE and the E*k*VN models for each dataset separated by body location (HCA=Hand,Chest,Ankle).

	Hand		Chest		Ankle		HCA		F.Pocket	
	EAE	EkVN	EAE	EkVN	EAE	EkVN	EAE	EkVN	EAE	EkVN
WISDM	-	-	-	-	-	-	-	-	80.8	73.1
MHealth	82.0	67.2	82.8	74.4	82.0	69.2	94.8	83.4	-	-
PAMAP2	60.0	70.7	60.0	60.8	59.0	70.7	56.0	80.8	-	-

6.4.5 Aggregation of Classes

One advantage of the EAE structure is the possibility of aggregating classes in different hierarchies in a simple manner. We can combine the AEs that represent similar classes and consider super-classes.

Considering the dataset PAMAP2, for example, we can aggregate its classes into the following super-classes: *House Cleaning* (which includes *Ironing* and *Vacuum Cleaning*), *Dynamic Positions* (*Ascending/Descending Stairs* and *Walking*), *Static Positions* (*Lying*, *Sitting* and *Standing*) and *Sports* (*Cycling*, *Nordic Walking*, *Rope Jumping* and *Running*). In this experiment, we still use the 12 AEs, however we consider as a true positive, when the classification is correct, any class belonging to the super-class (Table 24). The average accuracy was 74.1%, which is higher than the 60.0% showed in Table 22. This shows that AE from similar activities obtain smaller errors. In particular, the misclassification between the *Dynamic Positions* and *Static Positions* is quite low.

Table 24 – Average confusion matrix for EAE model for PAMAP2 dataset considering the aggregation of AEs.

	Dynamic Positions	Static Positions	Sports	House Cleaning
Dynamic Positions	**0.82**	0.05	0.00	0.13
Static Positions	0.05	**0.86**	0.01	0.09
Sports	0.09	0.13	**0.56**	0.23
House Cleaning	0.10	0.16	0.02	**0.72**

6.5 Conclusion

In this paper we proposed a new classification algorithm which we refer as Ensemble of Auto-Encoders (EAE). It uses a set of AEs where each is trained to reconstruct the sensor measurements from one unique class. This set of AE is then used as an ensemble for classification by predicting the class which corresponds to the AE with the lowest reconstruction error. We tested two variants of the EAE (one with online learning and other without) in HAR datasets and compared them with other methods. One was an ensemble of traditional approaches, E*k*VN, and the remaining are state-of-the-art deep learning approaches.

Experimental results show that the proposed EAE is competitive with existing methods found in HAR literature. We observed that the minimum accuracy of the EAE is always the highest in all the datasets tested. From this we can conclude that the EAE is more robust to data from different users, which is also supported by the low variance in accuracy.

We note that the presented results were obtained from models trained with accelerometer data only, which is usually a more challenging classification task. Moreover, a simple and unique architecture was used for all AE in all datasets without hyperparameter tuning.

The modular structure of the EAE proposed in this work has the advantage of making the model easily adapted. First of all, in the case of online learning, only the AEs corresponding to the most frequent activities are updated which can save computation time. In this way it is not necessary to retrain the whole model, as it would be necessary for most machine learning models. Therefore the EAE can specialise in the most performed (or preferred) activities of each user. Moreover this modular structure has also the advantage for the inclusion of new activities when is needed. For that, it is only necessary to add more AE and train each one with each new class. Likewise, it could be similarly adapted to forget activities, by simply removing the respective AE from the ensemble. Finally, another advantage of the EAE is that each AE can have its own architecture and even use different types of layers, such as Recurrent or Convolutional.

In terms of time consumption we see that models with more complex architectures are slower to train than simpler ones. In that sense the EAE, even though it has multiples models, has a similar time consumption to other deep learning models. However, since the concept of EAE can use many different architectures of the AEs, the time consumption can be reduced with different ones. Moreover, in terms of test/prediction, the time consumption represented in the results consider all the instances used in each experiment. Which means that the prediction of each instance took less than half a second.

As future work we intend to combine AE with different architectures in the same ensemble.

The final publication is available at <link.springer.com/chapter/10.1007/978-3-319-46307-0_1>

CHAPTER

7

A STUDY ON HYPERPARAMETER CONFIGURATION FOR HUMAN ACTIVITY RECOGNITION

Collaborating authors

Tiago Carvalho
University of Porto, Porto, Portugal

João Mendes-Moreira
LIAAD-INESC TEC, University of Porto, Porto, Portugal

João M.P. Cardoso
University of Porto, Porto, Portugal

André C. P. L. F. de Carvalho
University of São Paulo, São Carlos, Brazil

Abstract

Wearable technologies and smartphones have become more ubiquitous, and a large amount of information about a person's life has become available. Since each person has a unique way of performing physical activities, a Human Activity Recognition system needs to be adapted to the characteristics of a person in order to maintain or improve accuracy. Additionally, when smartphones devices are used to collect data, it is necessary to manage its limited resources, so the system can efficiently work for long periods of time. In this paper, we present a semi-supervised ensemble algorithm and an extensive study of the influence of hyperparameter configuration in classification accuracy. We also investigate how the classification accuracy is affected by the person and the activities performed. Experimental results show that it is possible

to maintain classification accuracy by adjusting hyperparameters, like window size and window overlap, depending on the person and activity performed.

7.1 Introduction

Advanced mobile devices, such as smartphones, are usually integrated with several sensors capable of any-time sensing and data collection. The different types of motions sensors, such as accelerometers, gyroscopes and magnetometers, allow mobile devices to obtain substantial user-related information by monitoring and tracking movements of their users (MILUZZO *et al.*, 2012).

Human Activity Recognition (HAR) is a machine learning task focused on the use of sensing technologies to classify human activities and to infer human behaviour (KRISHNAN; COOK, 2014). Extensive research has been carried out in this area in the last decade (AGGARWAL; RYOO, 2011; LARA; LABRADOR, 2013; RAMAMURTHY; ROY, 2018; SHOAIB *et al.*, 2015), for applications like health and well-being (DOBBINS; RAWASSIZADEH; MOMENI, 2017), mobile security (MILUZZO *et al.*, 2012; PISANI; LORENA, 2013) and elderly care (KRISHNAN; COOK, 2014).

Most approaches of HAR found in the literature are based on supervised learning algorithms and assume that the true data label is always available. However, this assumption may not be feasible in real online scenarios, when labelled data is rare, and the system feedback has to occur at runtime. As an example, in a fall detection system for elderly care, the classification feedback must occur as close as possible to the real moment of the user's fall (KRISHNAN; COOK, 2014).

Besides, as human beings perform activities differently, dissonant input signals are expected for the same activity (CARDOSO; MENDES-MOREIRA, 2016). To keep accuracy over time, classification models, used in HAR systems, need to be adapted to the current user. However, due to the limitations of most mobile devices, different hardware resources need to be managed, such as battery and execution power, in order to keep the system efficiently working and accurate over time. Thus, there is a trade-off between the amount of processed information and the resources available.

This work is based on an ensemble classifier firstly described in (REISS; STRICKER, 2012). This algorithm has two phases, an offline and an online phase. In the beginning, the offline phase, the ensemble model is trained with labelled data from several users. In the online phase, this ensemble is used as a basic model to classify activities from a specific user, not present in the ensemble training. The ensemble model can be updated online with the user's data, if the classification has a high confidence factor.

The main contributions of this work are the extensive study of two hyperparameters important for HAR classification: the window size and the overlapping between windows (overlap factor). We analyse the impact of these hyperparameters in the model classification accuracy. Additionally, we conducted experiments with an ODROID-XU+E board [1] to evaluate the impact of these hyperparameters regarding energy consumption and execution time in a hardware similar to a smartphone.

This paper is structured as follows. Section 7.2 presents the related work on HAR and window parameterisation. In Section 7.3, we describe the methodology applied in this study. The results obtained with the experiments are presented and discussed in Section 7.4. Finally, in Section 7.5, we summarise our main conclusions and point out future work directions.

7.2 Related Work

Dobbins et al. (DOBBINS; RAWASSIZADEH; MOMENI, 2017) propose an approach that uses personal data to infer on better lifestyle choices for its users. Considering only labelled data, they evaluate the predictive performance of 10 supervised HAR classifiers in terms of accuracy and mobile system performance (execution time and energy consumption). Their experimental setup is based on a fixed window size of 512 samples and overlap factor of 0.5, i.e., 256 samples are reused from the previous window and only 256 new samples are used for the current window. They suggest that the sensing data should be processed in the cloud and not in the device. However, personal privacy and Internet connection are not considered. Furthermore, all data used is labelled, which cannot be guaranteed in a real online mobile system. The datasets used in the experiments contain complex activities and different user's data, but the results are not compared in terms of accuracy per user.

Mannini et al. (MANNINI *et al.*, 2013) propose an SVM classifier to detect 4 activities from 33 different users. The classifier performance was tested for different window sizes, but not for the overlap between consecutive windows. Also, they do not compare, in terms of execution time, the classification task with different window sizes. The results show large variability among users performing the same activity, due to the problem of different sensor body location.

Other authors have also discussed window size. For example, (BAÑOS *et al.*, 2014b; HARASIMOWICZ; DZIUBICH; BRZESKI, 2014; BAÑOS *et al.*, 2014a; NIAZI *et al.*, 2017) compare the predictive performance of classifiers over a set of window sizes. However, most studies do not consider the use of overlap factor and the impact of the user on the obtained accuracy.

In (BAÑOS *et al.*, 2014b) it is presented an extensive review of the literature in window size and HAR. The accuracy of several classifiers was analysed for different window sizes, but

[1] ODROID-XU+E is a board mainly consisting of an Exynos5 Octa SoC, which includes 2 quad cores ARM CPUs and a PowerVR GPU, and a power measurement circuit to measure CPU, GPU and DRAM power consumption. The Exynos5 Octa SoC is used in numbers of families of smartphones.

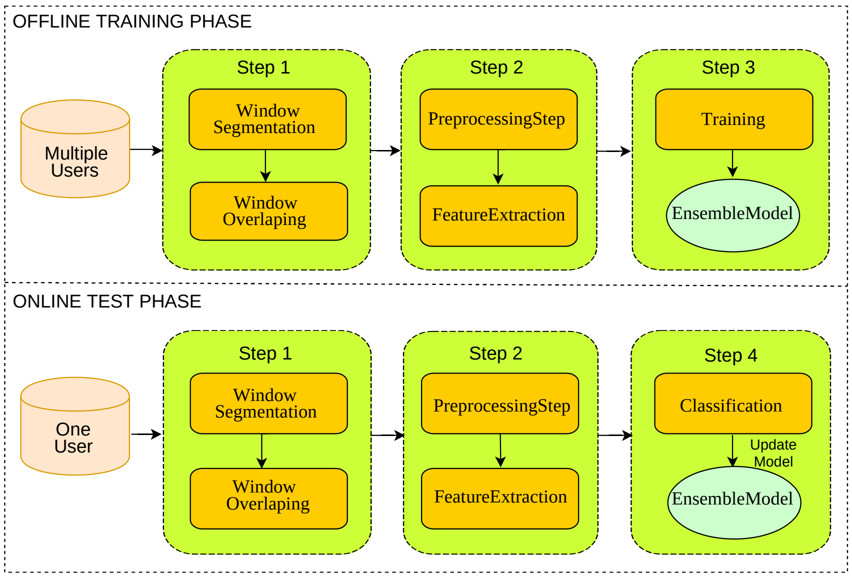


Figure 30 – Overview of the semi-supervised ensemble model for HAR.

not regarding users. Additionally, the experimental setup was not elaborated with a leave-one-user-out, which would be a more realist approach. Instead, they used a cross-validation approach, which is more affected by user variability than leave-one-participant-out.

The study conducted in this paper uses the PAMAP2 public dataset (REISS; STRICKER, 2012). PAMAP2 includes a vast number of sensors and more complex activities than the data used by many other studies. This dataset allows the study of the impact on HAR accuracy for different window sizes, users and activities.

7.3 Activity Recognition Overview

The HAR classification task can be split into 4 main steps, as illustrated in Figure 30. The steps $1, 2$ and 3 are the training phase with multiples users. The steps $1, 2$ and 4 are used for the online user-specific classification.

Figure 30 shows a batch with raw data extracted from different wearable sensors and/or smartphones. The raw data samples are stored in a sliding window with a fixed size. Ideally, a sliding window should contain data from a unique activity. However, perfect segmentation is not always feasible, so a between-window overlap factor can be used to include samples from sequential activities. Also, the size of the sliding window is reduced by the overlap factor, allowing a reduction of stored data. Thus, the step 1 is the window segmentation of the raw data and the overlapping of sequential windows.

Sensor's data are usually susceptible to noise, especially the accelerometers data (DOBBINS; RAWASSIZADEH; MOMENI, 2017). Thus, it is essential to process and convert the data into meaningful values. A preprocessing step (step 2) may also include calibration and filtering of the input signals in order to reduce noise. Sequential to that, a Feature Extraction (step 2) is used to calculate a single instance containing features that are then used for building the ensemble model. These features (see, e.g., (FIGO *et al.*, 2010)) include time-domain calculus, specifically mean and standard deviation for each sensor signal and correlation (Pearson correlation) between axes for the 3D sensors.

Each new instance is used to train (step 3) an ensemble model composed of three classifiers: kNN, VFDT and Naive Bayes. The implementation of the ensemble classifier is the combination of Democratic Co-Learning (ZHOU; GOLDMAN, 2004) and Tri-Training (ZHOU; LI, 2005).

After training the ensemble model, in the online phase, sensors data are acquired from a single user. This data is preprocessed, and features are extracted from them, similar to the processes (step 1 and 2) described in the training phase. Each generated instance is classified by the ensemble (step 4), which classifies the instance and provides a confidence factor for that classification. The instances classified with high confidence, more than 99% value, are used to update the ensemble model.

7.4 Experimental Results

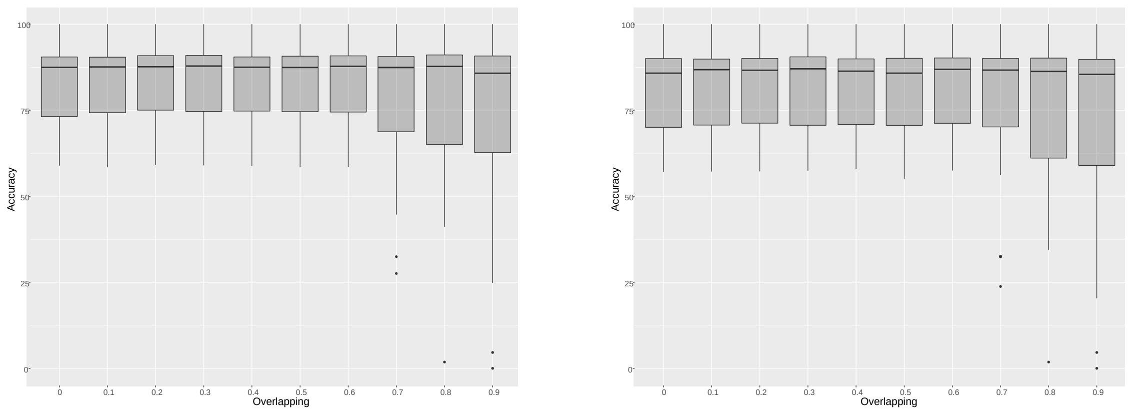
We conducted several experiments with our approach using the PAMAP2 dataset (REISS; STRICKER, 2012). The objectives of these experiments are: compare the accuracy of a supervised HAR versus a semi-supervised HAR when using different configurations of the hyperparameters: window size and overlap factor. We also intend to study a HAR system behaviour with the different hyperparameters configurations in terms of classification accuracy, energy consumption and execution time.

7.4.1 The PAMAP2 Dataset

The PAMAP2 (REISS; STRICKER, 2012) is a public dataset for human physical activities [2]. The data was collected from tree devices positioned in different body areas: wrist, chest and ankle. Each device has three sensors embedded: a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer.

The PAMAP2 dataset contains 1.926.896 samples of raw sensor data from 9 different users and 18 different activities. The activities executed by the users are divided into basic activities (walking, running, Nordic walking and cycling), posture activities (lying, sitting and standing), everyday activities (ascending and descending stairs), household (ironing and vacuum cleaning) and fitness activities (rope jumping). Also, the users were encouraged to perform

[2] http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring



(a) Semi-supervised (b) Supervised

Figure 31 – Supervised vs Semi-Supervised Ensemble accuracy: Overlap factor influence.

optional activities (watching TV, computer work, car driving, folding laundry, house cleaning, and playing soccer).

7.4.2 Experimental setup

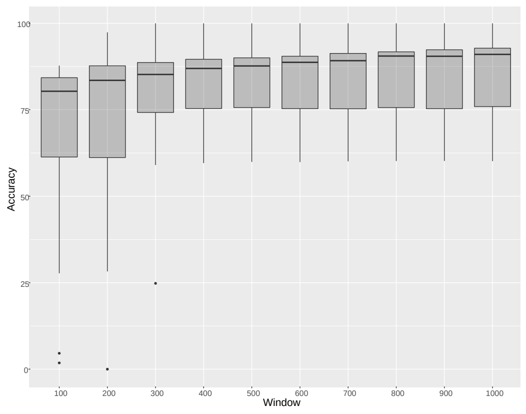
Using the PAMAP2 dataset (REISS; STRICKER, 2012), each ensemble was trained with data from 8 users and tested with one isolated user, not presented in the training process. This approach is called leave-one-user-out.

We conducted four experiments with two ensemble models, each one consisting of three classifiers: kNN, Naive Bayes, and Hoeffding Tree (VFDT), as in (CARDOSO; MENDES-MOREIRA, 2016). As verified in (DOBBINS; RAWASSIZADEH; MOMENI, 2017), these three classifiers have good classification performance in HAR problems. Thus, we analyse the accuracy performance of one ensemble model with a semi-supervised approach and another ensemble model with a supervised approach.

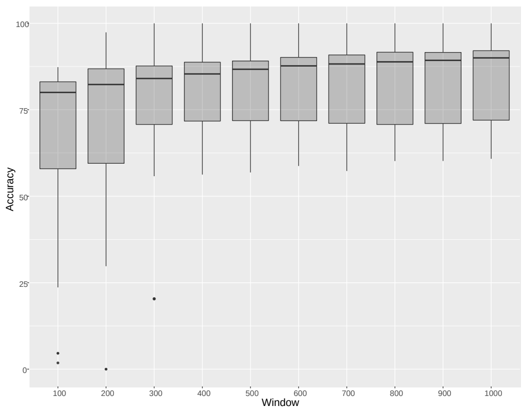
The box-plots correspond to the variance in accuracy for different values of window size (from 100 to 1000 with increments of 100), overlap factor (from 0.0 to 0.9 with increments of 0.1) and users (from 1 to 9 with increments of 1 user per experiment).

7.4.3 HAR Accuracy Results

Figure 31 presents the *accuracy* (axis y) for each value of the overlap factor, *overlapping* (axis x). The semi-supervised model reduces accuracy variance, compared with supervised model, for most of the overlapping and has average accuracy close to 90%. For both models, overlapping has more influence on accuracy for values higher than 0.7, but the semi-supervised model is less susceptible to that influence than the supervised model. As shown in Figure 32, variance of *accuracy* (axis y) and *window size* (axis x), the semi-supervised model reduces

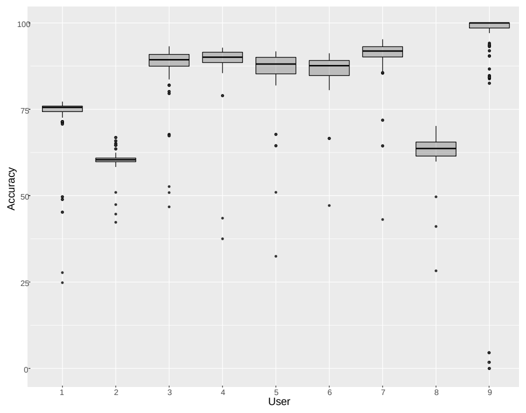


(a) Semi-supervised

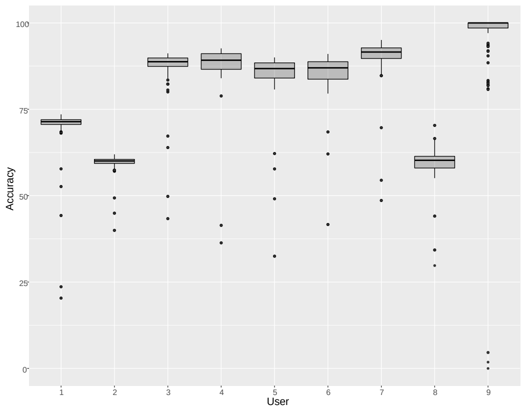


(b) Supervised

Figure 32 – Supervised vs Semi-Supervised Ensemble accuracy: Overlap factor influence.



(a) Semi-supervised



(b) Supervised

Figure 33 – Supervised vs Semi-Supervised Ensemble accuracy: Overlap factor influence.

accuracy variance for each value of window size. We also notice that windows with small sizes have worse results, especially for sizes of 100 and 200.

We analyse the models' accuracy for each user. For that, we analyse the variance of *accuracy* (axis y) when varying the hyperparameters *window size* and *overlapping* for each *user* (axis x). In Figure 33, for both models, user 5 and 6 have variance higher than users 2 and 1. The semi-supervised model reduces accuracy variance for users 4 and 8. An interesting case to analyse is user 9. For most of the cases, the accuracy is 100%; however, user 9 only has instances for the Rope Jumping activity, which means that this user influences the results to higher values. In some cases, the individual accuracy can be lower, as we can see with users 2, 8 and 1, justifying the analyses by the user instead of analysing all population.

With the results, we can see that window size and overlapping do influence the accuracy of the models. Based on these results and depending on the HAR application, one can decide about the window size and overlap factor level that make possible a certain minimum desired classification accuracy. The exhaustive exploration allows us also to understand the acceptable ranges to explore within an autotuning runtime system, e.g., to keep a minimum accuracy (e.g. 80%). These ranges can be used, at runtime, to search for the best combination of the hyperparameters that provide the best results, e.g., in terms of execution time or energy consumption. The following subsection shows the impact on execution time and energy consumption of different window sizes and overlap factor.

7.4.4 Execution Time and Energy Consumption

We also analyse the execution time and energy consumption for processing all the data from the PAMAP2 dataset. For that, we conducted experiments in an ODROID-XU+E [3] system running Android. The experiments focus on a single user, user 6, and the execution time and energy required to process 250.096 raw samples.

The first experiment is about the execution time necessary to process all the data of user 6. The execution time was divided into three parts. The first part, *samplingTime*, represents the time required to access all the data from the user and the instantiation of each data window as an instance. The second part, *featureTime* is the feature extraction and the "final instance" instantiation, this part depends on the *window size* and the *overlap* factor used. The last part, *classificationTime*, is the total time required to classify all the instances calculated in the feature extraction phase.

In Figure 34, since the feature extraction depends on the window size, the time to calculate all instances increases as the window size also increases, despite the decreasing number of calculated features. This means that the feature extraction phase is sensitive to the number of raw instances to process. Furthermore, as we increase the overlap factor, due to the increased number of instances that are calculated, the execution time also increases. The classification time is rather short and slightly increases as the number of calculated features augments.

The second experiment is presented as a heat map representing the energy, *Joules*, consumed to process raw data from user 6. For the different *window sizes* and *overlap factors*. The colours represent a range of *Joules*, where the red colour depicts higher energy consumed and, reversely, green colour depicts less energy consumed. The *accuracy* is also shown in the map over each circle depicting the energy colour to compare the energy consumed with the classification accuracy.

In Figure 35, we can see that smaller windows result in less energy consumption than bigger windows. This is due to the increased effort to calculate features for larger window sizes. It is also perceivable that increasing the overlap factor also increases the energy consumed, essentially due to the increased number of feature calculations and classifications to be carried out.

Relating the energy consumption with the accuracy achieved for a given configuration,

[3] https://www.hardkernel.com/

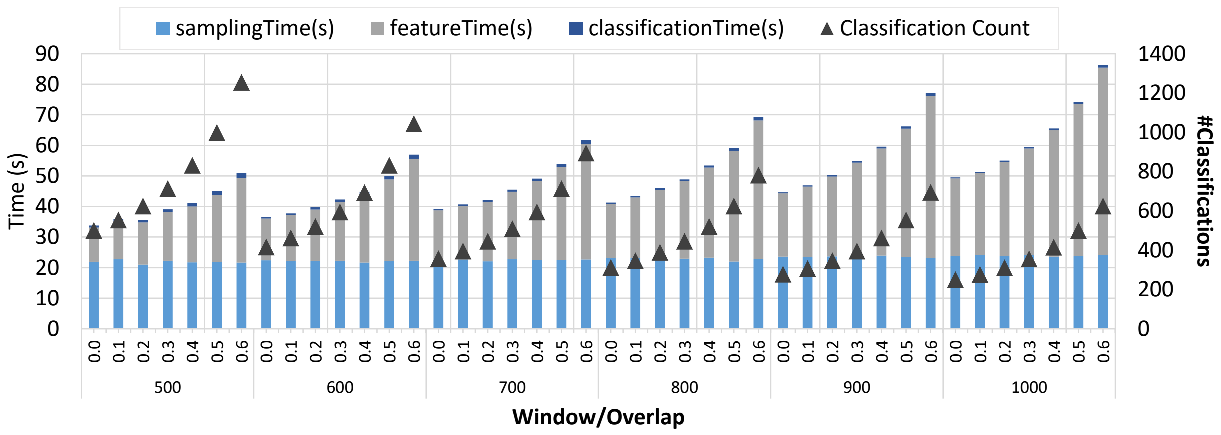


Figure 34 – Total execution time required (left axis) to process the PAMAP2 dataset, per window size and overlap factor, divided in three parts: sampling (data extraction), features extraction and classification. The number of classifications per configuration (right axis) is shown as triangle marks.

it is observable that the best accuracy values reside in more "heated" zones, i.e., where energy consumption is high. Smaller window sizes present lower accuracy, while larger window sizes provide higher accuracy. For instance, in configurations without overlapping (i.e., with an overlap factor of 0), the accuracy rises from 85% for a window of size 500 to 90% for a window of size 1000.

The overlap shows more fluctuations in terms of accuracy; however, with the best factors concentrated between 0.1 and 0.5. This shows that it is not trivial to select a single-window size and overlap factor if it is intended to have two possible scenarios, one where accuracy is the most important factor and another one where energy consumption is the top priority but still with a minimum accuracy value in mind.

7.5 Conclusion

In this work, we presented an analysis of the impact of hyperparameters, as window size and overlap factor, on HAR classification accuracy, execution time and energy consumption. The analysis was focused on a public dataset, which includes raw sensor data from 9 different users and 18 physical activities.

The experimental results confirm the need for adapting the classification model to the current user. Due to the impact of window size and overlap factor, each activity requires a specific configuration of these hyperparameters in order to improve classification accuracy.

Furthermore, the results also motivate the development of a system that is able to adapt the application at runtime when trade-offs between performance accuracy and energy consumption need to be considered. Bearing in mind this, the window size and overlap factor can be used to develop runtime strategies able to adapt these parameters according to the target goals.

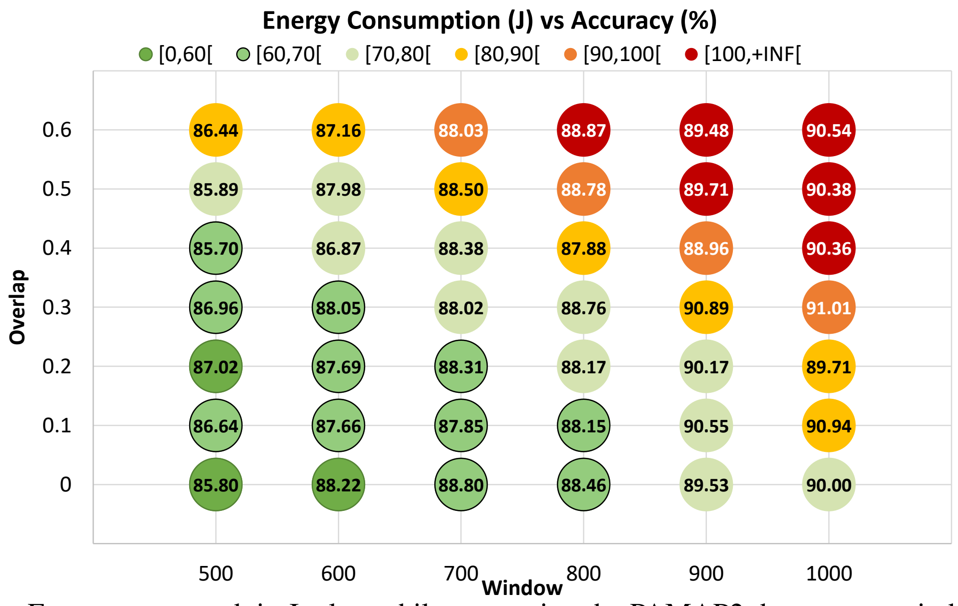


Figure 35 – Energy consumed, in Joules, while processing the PAMAP2 dataset, per window size and overlap factor. Green values represent less energy consumed while red values represent higher energy consumed. The values in each configuration represent the accuracy, in percentage, of that configuration.

As future work, we plan to implement a system able to adjust the window size and overlap factor dynamically and aware of activities and users. The dynamic adaptation needs to consider an exploration of possible parameter configurations to find the best configurations for each adaptation scenario, and thus, the experimental results presented in this paper are also part of that exploration phase.

CHAPTER

8

CONCLUSIONS AND FUTURE RESEARCH

Everything flows and nothing abides; everything gives way and nothing stays fixed.

Heraclitus of Ephesus

Learning from data stream is challenging because the data presents characteristics that are limitations to classical machine learning approaches. Among the limitations, we addressed the problem of the learning concept changes over time. These limitations require solutions capable of incremental learning, so that the model can keep its predictive performance.

In this thesis, we proposed methods [1] capable of automatically choose the moment to update the model to concept drift and novelties. The proposed methods can cope with changes even in the case of unlabelled data. We considered solutions that are able to detect more than one concept change at the same time. Our contributions are all incremental learning capable of adapting their models to the changes of the data stream.

We first addressed the problem of concept drift by proposing a method that selects representative data from the stream to update the model. Then, considering the problem of concept drift and novelties, we proposed a method that incrementally updates its model every time a new instance is classified. Considering the limitations of finding a suitable clustering partition, we also proposed a method based on ensemble of partitions. Moreover, we proposed two variations of this method: an ensemble of clustering from the same algorithm; and an ensemble of clustering from different algorithms. Our final contribution is based on neural networks, which is also unsupervised. In this contribution, we proposed an ensemble of specialist models. We applied our last contribution in human activity recognition. We next present a summary of the major contributions.

[1] All algorithms presented in this thesis are available for consultation in<https://github.com/keh>

8.1 Conclusions

In Chapter 3, we presented our first contribution. We developed a new method for data streams, based on the algorithm *k*NN, to address the problem of concept drift. This chapter is addressed to answer *RQ1*:

> *How to reduce the amount of data used to train a model and how to select the most representative data to update a model in data streams?*

To answer RQ1, we proposed the method *SWC* for the selection and the reduction of the prototypes used to train the classification model. SWC, stores only the representative data from the stream into a fixed-length sliding window. For that, in its online phase, SWC selects the data to be stored based on a probabilistic test. The method also decides when the data should be compressed. For the task of reducing the data, it is applied a clustering approach based on the *k*Means algorithm. Moreover, SWC only requires the label of a new data when the model is not confident about its classification decision. In the experiments, we compared our method with other methods well-known in the literature. According to the experimental results, SWC has competitive predictive performance, with lower processing and memory cost than the compared methods. This work has been published in the following paper:

- Garcia, K. D., de Carvalho, A. C., & Mendes-Moreira, J. (2018, November). A cluster-based prototype reduction for online classification. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 603-610). Springer.

We presented our second contribution in Chapter 4. We proposed an unsupervised learning method, called *Higia*, to learn novelties and concept drift in data streams. Thus, this chapter is addressed to answer *RQ2*:

> *How to incrementally learn concept changes in data streams, considering an unsupervised approach, without storing data for future analysis?*

Our contribution is a method capable of incrementally update its model every time a new data is classified. Higia uses micro-clusters as representatives to the concepts presented in the stream. The micro-clusters are a variation of the CF-Cluster, presented in Chapter 2. Higia can incrementally form new clusters, without the necessity of waiting for an event to happen. The new clusters can be incorporated into the model, or as novelties, or as concept drift, depending on how different they are from the normal concepts. Moreover, Higia allows for using more than one *k* (prototype) during the classification of new data. That way, the label of a new data (normal, extension or unknown) is decided by the number of prototypes that is similar to the new data. We compared Higia with the MINAS algorithm, the state-of-the-art algorithm for unsupervised learning in data streams. According to the experimental results, Higia presents a better predictive performance than MINAS. In the experiments, Higia shows to be faster in adapting its model to the current concepts than the MINAS algorithm. That shows the efficiency of the incremental update of the model to each new arriving data. This work has been published in the following paper:

- Garcia, K. D., Poel, M., Kok, J. N., & de Carvalho, A. C. (2019, September). Online Clustering for Novelty Detection and Concept Drift in Data Streams. In the EPIA Conference on Artificial Intelligence (pp. 448-459). Springer.

In Chapter 5, we presented our third contribution. We studied how to create clustering partitions that could better represent the data stream. For that, we investigated how to combine different clustering partitions into an ensemble for concept change detection in data streams. Thus, we focused on answering the *RQ3*:

> *How to combine clustering partitions from different clustering techniques and use them as a classification model in data streams?*

In Chapter 5, we also addressed the *RQ4*:

> *In which data streams can an ensemble model of clusters from different clustering approaches achieve higher predictive performance than an ensemble model of clusters from the same clustering approach?*

Considering *RQ3*, we based our contribution on two methods for ensemble clustering: one homogeneous ensemble obtained by the combination of the CluStream algorithm (called *HoCluS*); and one heterogeneous ensemble obtained by the combination of different clustering technique (called *HeCluS*). These methods allow for the combination of clustering partitions with different bias to obtain models that could better represent the data over time. In the experiments, we observed that the ensemble methods HoCluS and HeCluS have better predictive performance than the compared single model from the unsupervised method MINAS. The method HeCluS is usually more stable and has higher F-Measure than HoCluS and MINAS. The proposed methods showed to be competitive with the other tested methods. In most cases, the ensembles had better performance than the single models.

The experiments also helped us to answer *RQ4*, as they showed that the HeCluS has higher F-Measure than HoCluS, especially when applied on datasets with only concept drift. The use of the heterogeneous ensemble is a promising strategy because the combination of different biases creates a model that better represents the data. Both proposed methods have the advantage of not needing the labels to update themselves. Moreover, both proposed methods have a competitive performance with the compared supervised methods. Furthermore, the experiments showed us that all tested methods had better performance just in certain periods of the stream. Thus, the predictive performance of all tested methods was affected by the changes in the data streams. We concluded that a model with a single bias can be only suitable for a given data stream or for specific periods in the same data stream. This work has been published in the following paper:

- Garcia, K. D., de Faria, E. R., de Sá, C. R., Mendes-Moreira, J., Aggarwal, C. C., de Carvalho, A. C., & Kok, J. N. (2019, October). Ensemble Clustering for Novelty Detection in Data Streams. In International Conference on Discovery Science (pp. 460-470). Springer.

This work was extended and submitted to the Machine Learning Journal, and it is under review.

Our fourth contribution, Chapter 6, is a new classification method called Ensemble of Auto-Encoders (*EAE*). In this work, we focus on answering the *RQ5*.

How to use a set of auto-encoders for classification of data streams?

EAE is a set of auto-encoders in which each auto-encoder is trained to reconstruct the sensor measurements from one unique class. This set of auto-encoders is then used as an ensemble for classification. The classification of an instance is defined by the auto-encoder with the lowest reconstruction error. EAE has a modular structure, which makes the model easily adapted to concept changes. It is not necessary to retrain the whole model, as it would be necessary for most machine learning models. In the context of human activity recognition, for example, EAE can be specialised in the most performed (or preferred) activities of each user; which makes it a personalised model to a specific user. This modular structure also has the advantage for the inclusion of new activities, when new activities are detected. For that, it is only necessary to add more auto-encoders and train each one with the data from each new class. Likewise, it could be similarly adapted to forget activities by simply removing the respective auto-encoder from the ensemble. Finally, another advantage of the EAE is that each auto-encoder can have its architecture and even use different types of layers, such as Recurrent or Convolutional.

We tested EAE in the context of human activity recognition. In the experiments, we tested two variants of our method to analyse the improvement of the online learning. One variant of EAE is with online learning and other variant is without online learning. We compared the EAE variants with an ensemble of traditional approaches and with deep learning methods. The experimental results shows that the minimum accuracy of the EAE is always the highest, in all the datasets tested, in comparison with the other tested methods. The variant of EAE with online learning has less variance that the variant of EAE without online learning. We concluded that the

EAE is more robust to data from different users, which is also supported by its low variance of accuracy. This work has been published in the following journal paper:

- Garcia, K. D., de Sá, C. R., Poel, M., Carvalho, T., Mendes-Moreira, J., Cardoso, J. M., de Carvalho, A. C. & Kok, J. N. (2021). An Ensemble of Autonomous Auto-Encoders for Human Activity Recognition. Neurocomputing 439 (2021): 271-280.

In Chapter 7, we analysed the impact of varying the hyperparameters from a classification algorithm. For this problem, we considered a Human Activity Recognition system running in a smartphone. The analysis was based on the accuracy of the classification model responsible for classifying the user's physical activities; on the execution time of the system, and the energy consumption of the mobile device. The experimental results show that each activity requires a specific configuration of the hyperparameters to improve classification accuracy. The results also confirm the need of adapting the classification model to a specific user. Considering that the accuracy performance was considerably different from user to user, even when performing the same physical activity. Furthermore, the results also motivate the development of a system that can adapt the application at runtime, when trade-offs between performance accuracy and energy consumption need to be considered. This work has been published in the following papers:

- Garcia, K. D., Carvalho, T., Mendes-Moreira, J., Cardoso, J. M., & de Carvalho, A. C. (2019, May). A Study on Hyperparameter Configuration for Human Activity Recognition. In International Workshop on Soft Computing Models in Industrial and Environmental Applications (pp. 47-56). Springer.

8.2 Future Research

In this section, we discuss some possible open problems that could be explored in future work.

- **New strategies to forget old concepts**: in data streams, the act of learning also requires that a learning algorithm must forget the outdated information. In this thesis, we only focused on forgetting information when the buffer reaches a predefined size. Nevertheless, it may be interesting also to eliminate data that has been in memory for a specific amount of time. Depending on the application, old data can still be relevant, so, it should not be eliminated. Thus, it could be investigated alternatives solutions to discard outdated information.
- **Automatic and dynamic adaptation of hyperparameters**: hyperparameters defined in the beginning of a data stream could be unsuitable in other periods because of unpredictable changes in the data stream. Thus, a learning algorithm should have mechanisms to

automatic adapt its hyperparameters, since this could improve its predictive performance over time. The dynamic adaptation could be the exploration of possible hyperparameter configurations to find the best configurations. During a certain period of time, it could also estimate the hyperparameter according to the statistical distribution of that period. Among these hyperparameters, the dynamic window size could be energy-efficient and improve predictive performance; thus, it is an interesting aspect which should be investigated further.

- **Develop methods of meta-learning for data streams** In Chapter 5, we saw that an appropriate model at a certain period may rapidly become obsolete in other periods, requiring updating the model or its replacement. As there are several learning algorithms available, an interesting future investigation could be how to select the learning algorithm with the most suitable bias for a period of the stream.

- **Study on different architecture configuration to train the auto-encoders**: As we presented in Chapter 6, a set of auto-encoders can be used as a robust ensemble model for human activity recognition. However, there are still possibilities for improvements, especially in the combination of auto-encoders with different architectures. Another interesting aspect that could be investigated further is the possibility of each EAE using different types of layers, such as Recurrent or Convolutional neural networks. Besides, it should be investigated the efficiency of EAE when new classes appear in the stream. Likewise, how the method could forget unused auto-encoders from the ensemble. Another possibility is to explore how to use transfer learning to adapt the ensemble to different domains.

BIBLIOGRAPHY

ABDALLAH, Z. S.; GABER, M. M.; SRINIVASAN, B.; KRISHNASWAMY, S. Anynovel: detection of novel concepts in evolving data streams. **Evolving Systems**, v. 7, n. 2, p. 73–93, 2016. Citations on pages 41, 56, and 68.

ABDEL-HAMID, O.; MOHAMED, A.; JIANG, H.; PENN, G. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In: **2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012**. [s.n.], 2012. p. 4277–4280. Available: <https://doi.org/10.1109/ICASSP.2012.6288864>. Citation on page 90.

AGGARWAL, C. C. **Data streams: models and algorithms**. [S.l.]: Springer Science & Business Media, 2007. Citations on pages 29, 30, 32, 37, 38, 40, and 55.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: **VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany**. [S.l.: s.n.], 2003. p. 81–92. Citations on pages 38, 39, 47, 55, 57, 59, 67, 72, and 73.

AGGARWAL, J. K.; RYOO, M. S. Human activity analysis: A review. **ACM Comput. Surv.**, v. 43, n. 3, p. 16:1–16:43, 2011. Available: <https://doi.org/10.1145/1922649.1922653>. Citation on page 108.

AL-KHATEEB, T.; MASUD, M. M.; KHAN, L.; AGGARWAL, C. C.; HAN, J.; THURAISINGHAM, B. M. Stream classification with recurring and novel class detection using class-based ensemble. In: **12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012**. [S.l.: s.n.], 2012. p. 31–40. Citations on pages 56, 59, 60, 66, 68, and 76.

AMINI, A.; TEH, Y. W.; SABOOHI, H. On density-based data streams clustering algorithms: A survey. **J. Comput. Sci. Technol.**, v. 29, n. 1, p. 116–141, 2014. Citations on pages 56 and 67.

ASUNCION, A.; NEWMAN, D. **UCI Machine Learning Repository**. 2007. Available: <http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html>. Citation on page 60.

BAÑOS, O.; GALVEZ, J. M.; DAMAS, M.; GUILLÉN, A.; HERRERA, L. J.; POMARES, H.; ROJAS, I. Evaluating the effects of signal segmentation on activity recognition. In: **International Work-Conference on Bioinformatics and Biomedical Engineering, IWBBIO 2014, Granada, Spain, April 7-9, 2014**. [s.n.], 2014. p. 759–765. Available: <http://iwbbio.ugr.es/2014/papers/IWBBIO_2014_paper_83.pdf>. Citation on page 109.

BAÑOS, O.; GALVEZ, J. M.; DAMAS, M.; POMARES, H.; ROJAS, I. Window size impact in human activity recognition. **Sensors**, v. 14, n. 4, p. 6474–6499, 2014. Available: <https://doi.org/10.3390/s140406474>. Citations on pages 88, 90, and 109.

BAÑOS, O.; GARCÍA, R.; TERRIZA, J. A. H.; DAMAS, M.; POMARES, H.; RUIZ, I. R.; SAEZ, A.; VILLALONGA, C. mhealthdroid: A novel framework for agile development of mobile health applications. In: **Ambient Assisted Living and Daily Activities - 6th International Work-Conference, IWAAL 2014, Belfast, UK, December 2-5, 2014. Proceedings**. [S.l.: s.n.], 2014. p. 91–98. Citation on page 95.

BEDOGNI, L.; FELICE, M. D.; BONONI, L. By train or by car? detecting the user's motion type through smartphone sensors data. In: IEEE. **2012 IFIP Wireless Days**. [S.l.], 2012. p. 1–6. Citation on page 90.

BIFET, A.; GAVALDÀ, R. Learning from time-changing data with adaptive windowing. In: **Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA**. [S.l.: s.n.], 2007. p. 443–448. Citations on pages 31 and 45.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. MOA: massive online analysis. **Journal of Machine Learning Research**, v. 11, p. 1601–1604, 2010. Citation on page 46.

BIFET, A.; HOLMES, G.; PFAHRINGER, B.; KRANEN, P.; KREMER, H.; JANSEN, T.; SEIDL, T. MOA: massive online analysis, a framework for stream classification and clustering. In: **Proceedings of the First Workshop on Applications of Pattern Analysis, WAPA 2010, Cumberland Lodge, Windsor, UK, September 1-3, 2010**. [s.n.], 2010. p. 44–50. Available: <http://www.jmlr.org/proceedings/papers/v11/bifet10a.html>. Citation on page 59.

BIFET, A.; PFAHRINGER, B.; READ, J.; HOLMES, G. Efficient data stream classification via probabilistic adaptive windows. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013**. [S.l.: s.n.], 2013. p. 801–806. Citations on pages 31, 44, 45, 46, 47, 48, 49, 56, and 57.

BISHOP, C. M. **Pattern recognition and machine learning, 5th Edition**. Springer, 2007. (Information science and statistics). ISBN 9780387310732. Available: <http://www.worldcat.org/oclc/71008143>. Citation on page 67.

BREIMAN, L. Bagging predictors. **Machine Learning**, v. 24, n. 2, p. 123–140, 1996. Available: <https://doi.org/10.1007/BF00058655>. Citation on page 69.

CAO, F.; ESTER, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: **Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA**. [S.l.: s.n.], 2006. p. 328–339. Citations on pages 39, 59, 72, and 73.

CARDOSO, H. L.; MENDES-MOREIRA, J. Improving human activity classification through online semi-supervised learning. In: **Proceedings of the Workshop on Large-scale Learning from Data Streams in Evolving Environments (STREAMEVOLV 2016) co-located with the 2016 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2016), Riva del Garda, Italy, September 23, 2016**. [s.n.], 2016. Available: <http://ceur-ws.org/Vol-2069/STREAMEVOLV4.pdf>. Citations on pages 108 and 112.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. **ACM Comput. Surv.**, v. 41, n. 3, p. 15:1–15:58, 2009. Available: <https://doi.org/10.1145/1541880.1541882>. Citation on page 40.

CHEN, J.; SATHE, S.; AGGARWAL, C. C.; TURAGA, D. S. Outlier detection with autoencoder ensembles. In: **Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017**. [s.n.], 2017. p. 90–98. Available: <https://doi.org/10.1137/1.9781611974973.11>. Citation on page 91.

CHEN, Y.; XUE, Y. A deep learning approach to human activity recognition based on single accelerometer. In: **2015 IEEE International Conference on Systems, Man, and Cybernetics, Kowloon Tong, Hong Kong, October 9-12, 2015**. [s.n.], 2015. p. 1488–1492. Available: <https://doi.org/10.1109/SMC.2015.263>. Citations on pages 90 and 94.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, v. 7, p. 1–30, 2006. Citation on page 50.

DHEERU, D.; TANISKIDOU, E. K. **UCI Machine Learning Repository**. 2017. Available: <http://archive.ics.uci.edu/ml>. Citation on page 48.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. **International workshop on multiple classifier systems**. [S.l.], 2000. p. 1–15. Citation on page 90.

DING, X.; LI, Y.; BELATRECHE, A.; MAGUIRE, L. P. An experimental evaluation of novelty detection methods. **Neurocomputing**, v. 135, p. 313–327, 2014. Citations on pages 54 and 66.

DOBBINS, C.; RAWASSIZADEH, R.; MOMENI, E. Detecting physical activity within lifelogs towards preventing obesity and aiding ambient assisted living. **Neurocomputing**, v. 230, p. 110–132, 2017. Available: <https://doi.org/10.1016/j.neucom.2016.02.088>. Citations on pages 29, 42, 88, 90, 92, 108, 109, 110, and 112.

FARIA, E. R.; GAMA, J.; CARVALHO, A. C. P. L. F. Novelty detection algorithm for data streams multi-class problems. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013**. [S.l.: s.n.], 2013. p. 795–800. Citations on pages 37, 40, 41, 44, 46, 54, 56, 57, 59, 60, 66, 67, 68, 69, and 73.

FARIA, E. R.; GONÇALVES, I. J. C. R.; CARVALHO, A. C. P. L. F. de; GAMA, J. Novelty detection in data streams. **Artif. Intell. Rev.**, v. 45, n. 2, p. 235–269, 2016. Citations on pages 41, 59, 60, 73, and 74.

Farid, D. M.; Rahman, C. M. Novel class detection in concept-drifting data stream mining employing decision tree. In: **2012 7th International Conference on Electrical and Computer Engineering**. [S.l.: s.n.], 2012. p. 630–633. Citation on page 68.

FARID, D. M.; ZHANG, L.; HOSSAIN, M. A.; RAHMAN, C. M.; STRACHAN, R.; SEXTON, G.; DAHAL, K. P. An adaptive ensemble classifier for mining concept drifting data streams. **Expert Syst. Appl.**, v. 40, n. 15, p. 5895–5906, 2013. Available: <https://doi.org/10.1016/j.eswa.2013.05.001>. Citation on page 68.

FERREIRA, L. E. B.; GOMES, H. M.; BIFET, A.; OLIVEIRA, L. S. Adaptive random forests with resampling for imbalanced data streams. In: **International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019**. [s.n.], 2019. p. 1–6. Available: <https://doi.org/10.1109/IJCNN.2019.8852027>. Citation on page 29.

FIGO, D.; DINIZ, P. C.; FERREIRA, D. R.; CARDOSO, J. M. P. Preprocessing techniques for context recognition from accelerometer data. **Personal and Ubiquitous Computing**, v. 14, n. 7, p. 645–662, 2010. Available: <https://doi.org/10.1007/s00779-010-0293-9>. Citations on pages 89 and 111.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **J. Comput. Syst. Sci.**, v. 55, n. 1, p. 119–139, 1997. Available: <https://doi.org/10.1006/jcss.1997.1504>. Citation on page 69.

GAMA, J. **Knowledge Discovery from Data Streams**. CRC Press, 2010. (Chapman and Hall / CRC Data Mining and Knowledge Discovery Series). ISBN 978-1-4398-2611-9. Available: <http://www.crcpress.com/product/isbn/9781439826119>. Citations on pages 30, 31, 38, 41, 42, and 54.

____. A survey on learning from data streams: current and future trends. **Progress in AI**, v. 1, n. 1, p. 45–55, 2012. Available: <https://doi.org/10.1007/s13748-011-0002-6>. Citation on page 41.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. P. Learning with drift detection. In: **Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings**. [s.n.], 2004. p. 286–295. Available: <https://doi.org/10.1007/978-3-540-28645-5_29>. Citation on page 31.

GAMA, J.; ZLIOBAITE, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. **ACM Comput. Surv.**, v. 46, n. 4, p. 44:1–44:37, 2014. Citations on pages 30, 40, 44, 45, 46, 54, 56, 57, 66, and 69.

GAO, X.; LUO, H.; WANG, Q.; ZHAO, F.; YE, L.; ZHANG, Y. A human activity recognition algorithm based on stacking denoising autoencoder and lightgbm. **Sensors**, v. 19, n. 4, p. 947, 2019. Citations on pages 90 and 91.

GARCIA, D.; CARVALHO, A. C. P. L. F. de; MENDES-MOREIRA, J. A cluster-based prototype reduction for online classification. In: **Intelligent Data Engineering and Automated Learning - IDEAL 2018 - 19th International Conference, Madrid, Spain, November 21-23, 2018, Proceedings, Part I**. [S.l.: s.n.], 2018. p. 603–610. Citations on pages 34, 54, 56, 57, 68, and 69.

GARCIA, D.; FARIA, E. R. de; SÁ, C. R. de; MENDES-MOREIRA, J.; AGGARWAL, C. C.; CARVALHO, A. C. P. L. F. de; KOK, J. N. Ensemble clustering for novelty detection in data streams. In: **Discovery Science - 22nd International Conference, DS 2019, Split, Croatia, October 28-30, 2019, Proceedings**. [s.n.], 2019. p. 460–470. Available: <https://doi.org/10.1007/978-3-030-33778-0_34>. Citations on pages 34 and 67.

GARCIA, K. D.; CARVALHO, T.; MENDES-MOREIRA, J.; CARDOSO, J. M. P.; CARVALHO, A. C. P. L. F. de. A study on hyperparameter configuration for human activity recognition. In: **14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019) - Seville, Spain, May 13-15, 2019, Proceedings**. [S.l.: s.n.], 2019. p. 47–56. Citations on pages 34, 88, 89, 90, 91, and 96.

GARCIA, K. D.; POEL, M.; KOK, J. N.; CARVALHO, A. C. P. L. F. de. Online clustering for novelty detection and concept drift in data streams. In: **Progress in Artificial Intelligence, 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3-6, 2019, Proceedings, Part II**. [s.n.], 2019. p. 448–459. Available: <https://doi.org/10.1007/978-3-030-30244-3_37>. Citation on page 34.

GARCIA, K. D.; SÁ, C. R. de; POEL, M.; CARVALHO, T.; MENDES-MOREIRA, J.; CARDOSO, J. M.; CARVALHO, A. C. de; KOK, J. N. An ensemble of autonomous auto-encoders for human activity recognition. **Neurocomputing**, Elsevier, v. 439, p. 271–280, 2021. Citation on page 34.

GOGOI, P.; BHATTACHARYYA, D. K.; BORAH, B.; KALITA, J. K. A survey of outlier detection methods in network anomaly identification. **Comput. J.**, v. 54, n. 4, p. 570–588, 2011. Available: <https://doi.org/10.1093/comjnl/bxr026>. Citation on page 41.

GUO, J.; ZHOU, X.; SUN, Y.; PING, G.; ZHAO, G.; LI, Z. Smartphone-based patients' activity recognition by using a self-learning scheme for medical monitoring. **J. Medical Systems**, v. 40, n. 6, p. 140:1–140:14, 2016. Available: <https://doi.org/10.1007/s10916-016-0497-2>. Citations on pages 40 and 42.

HA, S.; CHOI, S. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In: **2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016**. [s.n.], 2016. p. 381–388. Available: <https://doi.org/10.1109/IJCNN.2016.7727224>. Citations on pages 90 and 94.

HA, S.; YUN, J.; CHOI, S. Multi-modal convolutional neural networks for activity recognition. In: **2015 IEEE International Conference on Systems, Man, and Cybernetics, Kowloon Tong, Hong Kong, October 9-12, 2015**. [s.n.], 2015. p. 3017–3022. Available: <https://doi.org/10.1109/SMC.2015.525>. Citations on pages 90 and 94.

HAQUE, A.; KHAN, L.; BARON, M. Semi supervised adaptive framework for classifying evolving data stream. In: **Advances in Knowledge Discovery and Data Mining - 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part II**. [s.n.], 2015. p. 383–394. Available: <https://doi.org/10.1007/978-3-319-18032-8_30>. Citations on pages 68 and 76.

HARASIMOWICZ, A.; DZIUBICH, T.; BRZESKI, A. Accelerometer-based human activity recognition and the impact of the sample size. In: **Proceedings of the 13th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, Gdansk, Poland**. [S.l.: s.n.], 2014. p. 15–17. Citation on page 109.

HAYAT, M. Z.; HASHEMI, M. R. A DCT based approach for detecting novelty and concept drift in data streams. In: **Second International Conference of Soft Computing and Pattern Recognition, SoCPaR 2010, Cergy Pontoise / Paris, France, December 7-10, 2010**. [S.l.: s.n.], 2010. p. 373–378. Citations on pages 56 and 68.

IENCO, D.; ZLIOBAITE, I.; PFAHRINGER, B. High density-focused uncertainty sampling for active learning over evolving stream data. In: **Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, BigMine 2014, New York City, USA, August 24, 2014**. [s.n.], 2014. p. 133–148. Available: <http://jmlr.org/proceedings/papers/v36/ienco14.html>. Citation on page 59.

JIANG, W.; YIN, Z. Human activity recognition using wearable sensors by deep convolutional neural networks. In: **Proceedings of the 23rd Annual ACM Conference on Multimedia Conference, MM '15, Brisbane, Australia, October 26 - 30, 2015**. [S.l.: s.n.], 2015. p. 1307–1310. Citations on pages 90 and 94.

JORDAO, A.; JR., A. C. N.; SOUZA, J. S. de; SCHWARTZ, W. R. Human activity recognition based on wearable sensor data: A standardization of the state-of-the-art. **CoRR**, abs/1806.05226, 2018. Available: <http://arxiv.org/abs/1806.05226>. Citation on page 94.

KIM, Y. Convolutional neural networks for sentence classification. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL**. [s.n.], 2014. p. 1746–1751. Available: <https://doi.org/10.3115/v1/d14-1181>. Citation on page 90.

KRANEN, P.; ASSENT, I.; BALDAUF, C.; SEIDL, T. The clustree: indexing micro-clusters for anytime stream mining. **Knowl. Inf. Syst.**, v. 29, n. 2, p. 249–272, 2011. Available: <https://doi.org/10.1007/s10115-010-0342-8>. Citations on pages 39 and 72.

KRISHNAN, N. C.; COOK, D. J. Activity recognition on streaming sensor data. **Pervasive and Mobile Computing**, v. 10, p. 138–154, 2014. Available: <https://doi.org/10.1016/j.pmcj.2012.07.003>. Citations on pages 42 and 108.

KWAPISZ, J. R.; WEISS, G. M.; MOORE, S. Activity recognition using cell phone accelerometers. **SIGKDD Explorations**, v. 12, n. 2, p. 74–82, 2010. Citation on page 94.

LARA, O. D.; LABRADOR, M. A. A survey on human activity recognition using wearable sensors. **IEEE Communications Surveys and Tutorials**, v. 15, n. 3, p. 1192–1209, 2013. Available: <https://doi.org/10.1109/SURV.2012.110112.00192>. Citations on pages 42, 88, 90, and 108.

LOSING, V.; HAMMER, B.; WERSING, H. KNN classifier with self adjusting memory for heterogeneous concept drift. In: **IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain**. [s.n.], 2016. p. 291–300. Available: <https://doi.org/10.1109/ICDM.2016.0040>. Citations on pages 31, 44, 45, 48, 54, and 56.

MA, C.; LI, W.; CAO, J.; DU, J.; LI, Q.; GRAVINA, R. Adaptive sliding window based activity recognition for assisted livings. **Information Fusion**, v. 53, p. 55–65, 2020. Available: <https://doi.org/10.1016/j.inffus.2019.06.013>. Citation on page 95.

MAKKUVA, A. V.; VISWANATH, P.; KANNAN, S.; OH, S. Breaking the gridlock in mixture-of-experts: Consistent and efficient algorithms. In: **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**. [S.l.: s.n.], 2019. p. 4304–4313. Citation on page 89.

MANNINI, A.; INTILLE, S. S.; ROSENBERGER, M.; SABATINI, A. M.; HASKELL, W. Activity recognition using a single accelerometer placed at the wrist or ankle. **Medicine and science in sports and exercise**, NIH Public Access, v. 45, n. 11, p. 2193, 2013. Citations on pages 42, 88, 90, and 109.

MARKOU, M.; SINGH, S. Novelty detection: a review - part 1: statistical approaches. **Signal Processing**, v. 83, n. 12, p. 2481–2497, 2003. Citation on page 54.

MASUD, M. M.; CHEN, Q.; KHAN, L.; AGGARWAL, C. C.; GAO, J.; HAN, J.; SRIVASTAVA, A. N.; OZA, N. C. Classification and adaptive novel class detection of feature-evolving data streams. **IEEE Trans. Knowl. Data Eng.**, v. 25, n. 7, p. 1484–1497, 2013. Available: <https://doi.org/10.1109/TKDE.2012.109>. Citations on pages 31, 41, and 66.

MASUD, M. M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. M. Classification and novel class detection in data streams with active mining. In: **Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II**. [S.l.: s.n.], 2010. p. 311–324. Citations on pages 39, 41, and 68.

____. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Trans. Knowl. Data Eng.**, v. 23, n. 6, p. 859–874, 2011. Citations on pages 56, 59, 60, 68, 69, 73, and 76.

MELLO, R. F. de; VAZ, Y.; FERREIRA, C. H. G.; BIFET, A. On learning guarantees to unsupervised concept drift detection on data streams. **Expert Syst. Appl.**, v. 117, p. 90–102, 2019. Available: <https://doi.org/10.1016/j.eswa.2018.08.054>. Citations on pages 40 and 41.

MILUZZO, E.; VARSHAVSKY, A.; BALAKRISHNAN, S.; CHOUDHURY, R. R. Tapprints: your finger taps have fingerprints. In: **The 10th International Conference on Mobile Systems, Applications, and Services, MobiSys'12, Ambleside, United Kingdom - June 25 - 29, 2012**. [s.n.], 2012. p. 323–336. Available: <https://doi.org/10.1145/2307636.2307666>. Citation on page 108.

MORENO-TORRES, J. G.; RAEDER, T.; ALAÍZ-RODRÍGUEZ, R.; CHAWLA, N. V.; HERRERA, F. A unifying view on dataset shift in classification. **Pattern Recognit.**, v. 45, n. 1, p. 521–530, 2012. Available: <https://doi.org/10.1016/j.patcog.2011.06.019>. Citation on page 40.

NIAZI, A. H.; YAZDANSEPAS, D.; GAY, J. L.; MAIER, F. W.; RAMASWAMY, L.; RASHEED, K.; BUMAN, M. P. Statistical analysis of window sizes and sampling rates in human activity recognition. In: **Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2017) - Volume 5: HEALTHINF, Porto, Portugal, February 21-23, 2017**. [s.n.], 2017. p. 319–325. Available: <https://doi.org/10.5220/0006148503190325>. Citations on pages 89 and 109.

PANG-NING, T.; STEINBACH, M.; KUMAR, V. *et al.* Introduction to data mining. In: **Library of Congress**. [S.l.: s.n.], 2006. p. 74. Citation on page 38.

PANWAR, M.; DYUTHI, S. R.; PRAKASH, K. C.; BISWAS, D.; ACHARYYA, A.; MAHARATNA, K.; GAUTAM, A.; NAIK, G. R. CNN based approach for activity recognition using a wrist-worn accelerometer. In: **2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jeju Island, South Korea, July 11-15, 2017**. [s.n.], 2017. p. 2438–2441. Available: <https://doi.org/10.1109/EMBC.2017.8037349>. Citations on pages 90 and 94.

PISANI, P. H.; LORENA, A. C. A systematic review on keystroke dynamics. **J. Braz. Comp. Soc.**, v. 19, n. 4, p. 573–587, 2013. Available: <https://doi.org/10.1007/s13173-013-0117-7>. Citation on page 108.

PLÖTZ, T.; HAMMERLA, N. Y.; OLIVIER, P. Feature learning for activity recognition in ubiquitous computing. In: **IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2011. p. 1729–1734. Citation on page 89.

RAMAMURTHY, S. R.; ROY, N. Recent trends in machine learning for human activity recognition - A survey. **Wiley Interdiscip. Rev. Data Min. Knowl. Discov.**, v. 8, n. 4, 2018. Available: <https://doi.org/10.1002/widm.1254>. Citation on page 108.

REISS, A.; STRICKER, D. Creating and benchmarking a new dataset for physical activity monitoring. In: **The 5th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2012, Heraklion, Crete, Greece, June 6-9, 2012**. [s.n.], 2012. p. 40. Available: <https://doi.org/10.1145/2413097.2413148>. Citations on pages 95, 108, 110, 111, and 112.

SANTOS, C. dos; GATTI, M. Deep convolutional neural networks for sentiment analysis of short texts. In: **Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers**. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, 2014. p. 69–78. Available: <https://www.aclweb.org/anthology/C14-1008>. Citation on page 90.

SEYFIOGLU, M. S.; ÖZBAYOGLU, A. M.; GURBUZ, S. Z. Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities. **IEEE Trans. Aerospace and Electronic Systems**, v. 54, n. 4, p. 1709–1723, 2018. Citations on pages 89 and 90.

SHOAIB, M.; BOSCH, S.; INCEL, Ö. D.; SCHOLTEN, H.; HAVINGA, P. J. M. A survey of online activity recognition using mobile phones. **Sensors**, v. 15, n. 1, p. 2059–2085, 2015. Available: <https://doi.org/10.3390/s150102059>. Citation on page 108.

SILVA, J. de A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. P. L. F. de; GAMA, J. Data stream clustering: A survey. **ACM Comput. Surv.**, v. 46, n. 1, p. 13:1–13:31, 2013. Citations on pages 38, 54, and 66.

SPINOSA, E. J.; CARVALHO, A. C. P. de Leon Ferreira de; GAMA, J. Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: **Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16-20, 2008**. [S.l.: s.n.], 2008. p. 976–980. Citations on pages 40, 41, and 56.

____. Novelty detection with application to data streams. **Intell. Data Anal.**, v. 13, n. 3, p. 405–422, 2009. Citations on pages 41, 56, 66, 67, and 68.

SPINSANTE, S.; ANGELICI, A.; LUNDSTRÖM, J.; ESPINILLA, M.; CLELAND, I.; NUGENT, C. D. A mobile application for easy design and testing of algorithms to monitor physical activity in the workplace. **Mobile Information Systems**, v. 2016, p. 5126816:1–5126816:17, 2016. Citation on page 88.

STREET, W. N.; KIM, Y. A streaming ensemble algorithm (SEA) for large-scale classification. In: **Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001**. [S.l.: s.n.], 2001. p. 377–382. Citation on page 48.

TAN, S. C.; TING, K. M.; LIU, F. T. Fast anomaly detection for streaming data. In: **IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011**. [s.n.], 2011. p. 1511–1516. Available: <https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-254>. Citation on page 68.

THOMAS, S.; BOUROBOU, M.; LI, J. Ensemble of deep autoencoder classifiers for activity recognition based on sensor modalities in smart homes. In: **Data Science - 4th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2018, Zhengzhou, China, September 21-23, 2018, Proceedings, Part II**. [s.n.], 2018. p. 273–295. Available: <https://doi.org/10.1007/978-981-13-2206-8_24>. Citation on page 91.

VEGA-PONS, S.; RUIZ-SHULCLOPER, J. A survey of clustering ensemble algorithms. **IJPRAI**, v. 25, n. 3, p. 337–372, 2011. Available: <https://doi.org/10.1142/S0218001411008683>. Citations on pages 67 and 69.

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; MANZAGOL, P. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. **J. Mach. Learn. Res.**, v. 11, p. 3371–3408, 2010. Citation on page 90.

WANG, J.; CHEN, Y.; HAO, S.; PENG, X.; HU, L. Deep learning for sensor-based activity recognition: A survey. **Pattern Recognition Letters**, v. 119, p. 3–11, 2019. Citations on pages 89 and 90.

WANG, L. Recognition of human activities using continuous autoencoders with wearable sensors. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 2, p. 189, 2016. Citations on pages 90 and 92.

YAO, S.; HU, S.; ZHAO, Y.; ZHANG, A.; ABDELZAHER, T. F. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In: **Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017**. [S.l.: s.n.], 2017. p. 351–360. Citation on page 88.

ZDRAVEVSKI, E.; LAMESKI, P.; TRAJKOVIK, V.; KULAKOV, A.; CHORBEV, I.; GOLEVA, R.; POMBO, N.; GARCIA, N. M. Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering. **IEEE Access**, v. 5, p. 5262–5280, 2017. Available: <https://doi.org/10.1109/ACCESS.2017.2684913>. Citation on page 40.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. BIRCH: an efficient data clustering method for very large databases. In: **Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996.** [S.l.: s.n.], 1996. p. 103–114. Citations on pages 38, 39, and 55.

ZHENG, Y.; LIU, Q.; CHEN, E.; GE, Y.; ZHAO, J. L. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. **Frontiers Comput. Sci.**, v. 10, n. 1, p. 96–112, 2016. Citations on pages 89 and 90.

ZHOU, Y.; GOLDMAN, S. A. Democratic co-learning. In: **16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15-17 November 2004, Boca Raton, FL, USA**. [s.n.], 2004. p. 594–602. Available: <https://doi.org/10.1109/ICTAI.2004.48>. Citation on page 111.

ZHOU, Z.; LI, M. Tri-training: Exploiting unlabeled data using three classifiers. **IEEE Trans. Knowl. Data Eng.**, v. 17, n. 11, p. 1529–1541, 2005. Available: <https://doi.org/10.1109/TKDE.2005.186>. Citations on pages 91 and 111.

ZLIOBAITE, I.; BIFET, A.; PFAHRINGER, B.; HOLMES, G. Active learning with drifting streaming data. **IEEE Trans. Neural Netw. Learning Syst.**, v. 25, n. 1, p. 27–39, 2014. Citations on pages 44, 48, and 54.

ŽLIOBAITĖ, I.; BIFET, A.; READ, J.; PFAHRINGER, B.; HOLMES, G. Evaluation methods and decision theory for classification of streaming data with temporal dependence. **Machine Learning**, Springer, v. 98, n. 3, p. 455–482, 2015. Citation on page 42.

ZLIOBAITE, I.; KUNCHEVA, L. I. Determining the training window for small sample size classification with concept drift. In: **ICDM Workshops 2009, IEEE International Conference on Data Mining Workshops, Miami, Florida, USA, 6 December 2009**. [s.n.], 2009. p. 447–452. Available: <https://doi.org/10.1109/ICDMW.2009.20>. Citation on page 31.

ŽLIOBAITĖ, I.; PECHENIZKIY, M.; GAMA, J. An overview of concept drift applications. In: **Big data analysis: new algorithms for a new society**. [S.l.]: Springer, 2016. p. 91–114. Citation on page 40.

ZOU, H.; ZHOU, Y.; YANG, J.; JIANG, H.; XIE, L.; SPANOS, C. J. Deepsense: Device-free human activity recognition via autoencoder long-term recurrent convolutional network. In: **2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018**. [S.l.: s.n.], 2018. p. 1–6. Citations on pages 89, 90, and 92.