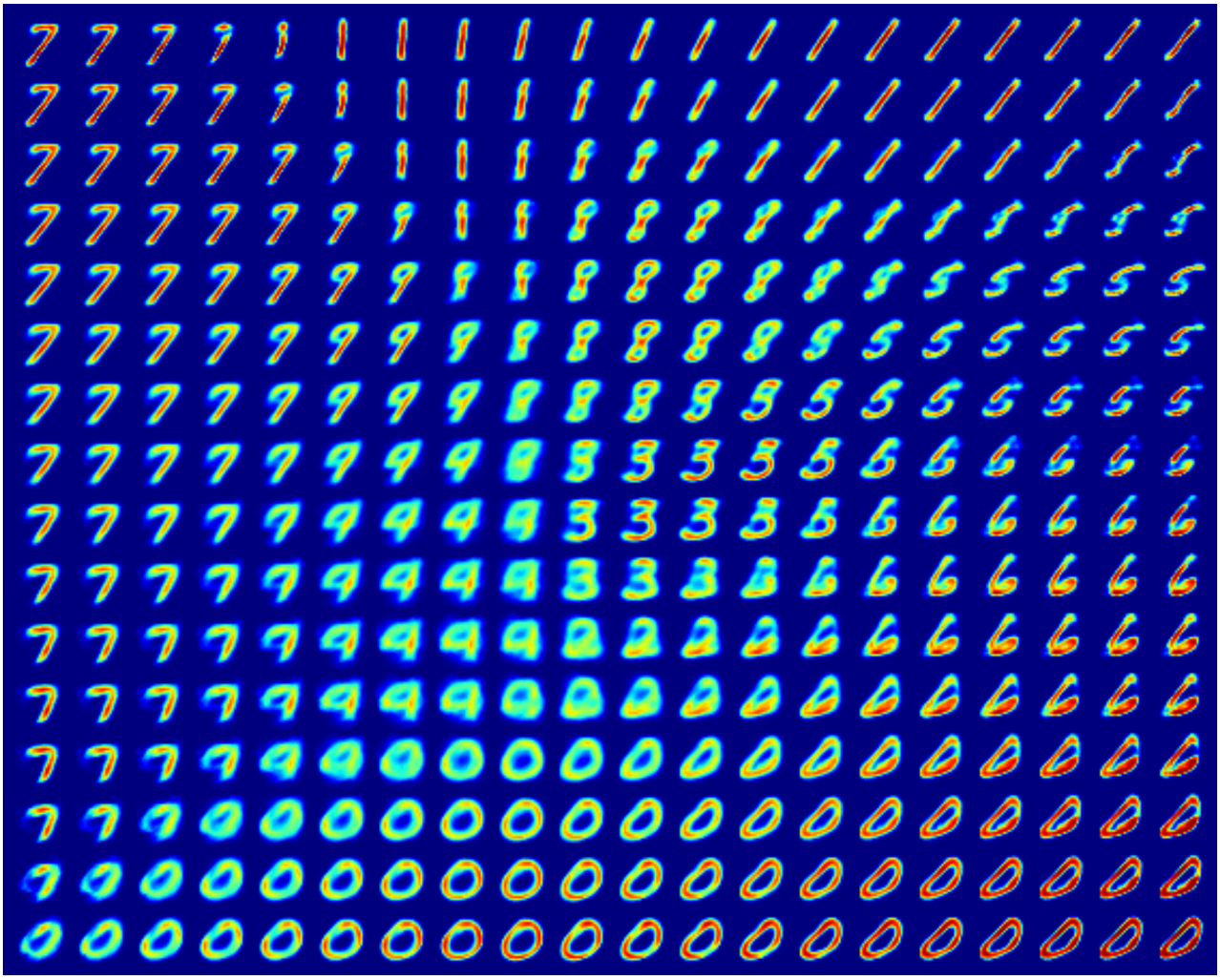
CHALMERS | UNIVERSITY OF GOTHENBURG



# Clustering Non-Stationary Data Streams with Online Deep Learning

Bachelor's thesis in Computer Science and Engineering

AURÉLIEN HONTABAT
MAGNUS RISING

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden 2016

**Clustering Non-Stationary Data Streams with Online Deep Learning**
AURÉLIEN HONTABAT
MAGNUS RISING



Examiner: Peter Lundin, Thorsten Berger

ISSN 1654-4676

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000



Cover:
The image depicts a way to visualize the latent space of a Variational Autoencoder, by plotting its generated reconstructions at the points in the latent space for which they have been generated.

Chalmers Reproservice
Göteborg, Sweden 2016

# Clustering Non-Stationary Data Streams with Online Deep Learning

AURÉLIEN HONTABAT
MAGNUS RISING
*Department of Computer Science and Engineering, Chalmers University of Technology*

University of Gothenburg

Bachelor's thesis

## Abstract

With more devices connected, sensor data logged and people active in social networks, the trend towards working with dynamic data is clear. The number of applications where it becomes essential to perform real time analysis on data streams grows accordingly, each with its own challenges. From this area of data stream analysis we benchmark the performance of current state of the art clustering algorithms: CluStream, DenStream and ClusTree. We also adapt a Variational Autoencoder to perform in the context of non-stationary data streams and assess its generative capabilities for dimensionality reduction. From this limited lab experiment we show that while there is a significant improvement in the clustering accuracy of high dimensional datasets after a dimensionality reduction with a Variational Autoencoder, not all clustering algorithms benefit in the same way from it. Additionally we show that regardless of the clustering algorithm, no relevant improvement in the purity of the clusters could be obtained after the dimensionality reduction.

**Keywords:** Clustering, Deep Learning, Data Streams, Dimensionality Reduction

# Nomenclature

| | |
|---|---|
| **ADBN** | Adaptive Deep Belief Network |
| **ANN** | Artificial Neural Network |
| **ANOVA** | Analysis of Variance |
| **DBN** | Deep Belief Network |
| **FN** | False Negative |
| **FP** | False Positive |
| **GPU** | Graphics Processing Unit |
| **kNN** | k-Nearest Neighbors |
| **MLP** | Multilayer Perceptron |
| **MOA** | Massive Online Analysis |
| **RBM** | Restricted Boltzmann Machine |
| **TN** | True Negative |
| **TP** | True Positive |
| **VAE** | Variational Autoencoder |

# Contents

# 1 Introduction

With more and more devices and sensors connected, people active on social networks and database transactions logged, the volume of data generated every year has been increasing exponentially [27]. This trend towards working with dynamic data, often generated at high speed, means that it must be analyzed in real time. The real time analysis of data streams is only becoming more important as the number of applications in this area grows. Such applications range from identifying different topics of discussion in social media, grouping pictures according to their contents to identifying the user of a device from sensor data.

## 1.1 Background

Knowledge discovery from large quantities of data, also referred to as big data analytic, can be done by processing data in batches whenever it is possible to keep records of the data. However this is not always a realistic approach, the amount of data to be processed might constantly increase with time and holding it in memory might not be an option. Additionally some data is volatile by nature, even if it could be saved and processed again at a later time, old records would be of limited use or even have a negative impact on predictions because the model underlying the data has changed over time [1]. Consumption patterns for example, may differ during holidays from the rest of the year and cause otherwise correct predictions to suddenly be inaccurate. Data that exhibits this property is also referred to as evolving data. The alternative to storing the data is to extract patterns and structures sequentially and in real time as the data arrives. This approach, known as **data stream mining**, is a challenging setting for knowledge discovery.

One core technique used to extract knowledge in many different data mining tasks is **clustering**. Clustering is the process of identifying regions in the data space more densely occupied than others and grouping them into clusters. These regions are populated by data points that exhibit similar properties and the resulting clusters can be indicators, in the context of streams, for classes of behavior or events. When analyzing social media data these classes can for example be discussion topics in message streams. Knowing what discussion topics are trending and detecting different types of content flowing through a social network is necessary to understand public opinion and interest or detect and filter spam. More generally clustering is useful to identify structure from data that is not yet classified. Clustering data streams presents several additional challenges compared to batch clustering. High clustering accuracy is more difficult to achieve since it is not possible to perform multiple passes on the same data to refine clusters. Additionally the clustering algorithm must be performant enough to handle the data throughput, as it needs to handle the data in real time. Pausing the stream while the clustering algorithm catches up is not an option. Lastly a static snapshot of the clusters is not enough, the temporal changes need to be part of any clustering information.

Many clustering techniques cannot cope with the special nature of data streams (unbound amounts of data, continuous arrival, evolving data). A specific branch of machine learning that deals with data only being available sequentially, as a stream of data, is **online learning**. From online learning several algorithms exist to cluster data streams, they are in many cases modified versions of their batch processing counterparts such as partitive, hierarchical or density based unsupervised learning methods. These algorithms are capable of handling the arriving data in one pass.

More recent machine learning techniques such as **deep learning** algorithms are capable of extracting more complex models from higher dimensional data than the previously mentioned approaches. They develop a layered, hierarchical representation of data, where more abstract features are defined in terms of less abstract features [14]. Additionally they learn in a generative way, modeling how the data is generated instead of how the data is divided. In the context of clustering, such an algorithm can be used as a feature extractor, where the data is mapped onto a lower-dimensional space while as much as the original information is kept is order to accurately cluster the data.

## 1.2 Problem Domain and Motivation

While the data from text messages might seem very different from that of images, the problem of clustering that data is similar no matter the context. After some preprocessing most data can be treated as a vector and grouped by the same clustering algorithm. The data in the scope of this thesis has also in common that it is not previously labeled for classification purposes, meaning that the clustering algorithm has no information or feedback as to what class the data point actually belongs to. The labels are only used for external evaluation purposes. Lastly we deal exclusively with non-stationary datasets, due to evolving nature of data streams. The problem domain is therefore clustering unlabeled, non-stationary data from different contexts both real world data and synthetic data. Note that we do not restrict ourselves to high-dimensional data as we want to asses the performance change across the board.

In this domain, not much research has been done in regards to adapting deep learning techniques to improve clustering results. However adapting deep learning techniques to the constraints of data stream clustering is an important area to explore [6]. Doing so would allow for not only better models of data that don't require large databases to be trained, but could also allow to generate synthetic data (data that keeps the features and the general distribution of the underlying model without the original attributes). Synthesizing sensible and private data in real time would allow for applications to make use of information that can't be accessed due to privacy concerns, for example from connected cars. Some research regarding online deep learning has already been done: Roberto Calandra *et al.* have developed a proof of concept called Adaptive Deep Belief Network (ADBN) that can be trained online making use of the ADBNs generative capabilities [4]. However their research didn't discuss the performance of their approach for dimensionality reduction. Additionally, other deep learning techniques exist that might work better for feature extraction. We want to investigate how different clustering algorithms perform and how an alternative deep learning technique, such as the Variational Autoencoder (VAE), can be used in the online phase of data stream analysis to improve clustering performance.

## 1.3 Research Goal and Research Questions

The goal of this bachelor's thesis is to adapt a deep learning model to improve clustering performance in the context of non-stationary data streams and compare it, as a limited lab experiment, to existing approaches.

### 1.3.1 Research Questions

**RQ 1** How does the performance of different clustering algorithms differ in the context of non-stationary data streams?

**RQ 2** How do the previously assessed clustering algorithms perform on the same datasets after feature extraction by a deep learning model?

The **performance** of the different clustering algorithms is evaluated through the F1-R, the F1-P and Purity measures as described in **Section 4.4**. The **different clustering algorithms** that are evaluated are CluStream, ClusTree and DenStream as implemented in the MOA framework [3]. The **deep learning model** used for the feature extraction preprocessing step is a Variational Autoencoder (VAE), as described in **Section 2.4.1**. **20 different datasets** are used for evaluating the performance of the different clustering algorithms, they are described in **Section 4.1**.

### 1.3.2 Hypotheses

**H0 F1R** There is no statistical difference in the average F1-R measure of the tested clustering models.

**H1 F1R** There is a statistical difference in the average F1-R measure of the tested clustering models.

**H0 F1P** There is no statistical difference in the average F1-P measure of the tested clustering models.

**H1 F1P** There is a statistical difference in the average F1-P measure of the tested clustering models.

**H0 Pur** There is no statistical difference in the average Purity measure of the tested clustering models.

**H1 Pur** There is a statistical difference in the average Purity measure of the tested clustering models.

**H0 VAE-F1R** There is no improvement in the average F1-R measure of the tested clustering models when using the deep learning model for feature extraction.

**H1 VAE-F1R** There is an improvement in the average F1-R measure of the tested clustering models when using the deep learning model for feature extraction.

**H0 VAE-F1P** There is no improvement in the average F1-P measure of the tested clustering models when using the deep learning model for feature extraction.

**H1 VAE-F1P** There is an improvement in the average F1-P measure of the tested clustering models when using the deep learning model for feature extraction.

**H0 VAE-Pur** There is no improvement in the average Purity measure of the tested clustering models when using the deep learning model for feature extraction.

**H1 VAE-Pur** There is an improvement in the average Purity measure of the tested clustering models when using the deep learning model for feature extraction.

## 1.4 Scope

A couple of open challenges for data stream mining research fall outside the scope of this limited lab experiment and are therefore not further discussed:

**Missing Values** or incomplete features in the data and how to handle these is a problem extensively discussed in the setting of offline learning. Only few works address data streams or evolving data streams [13]. Our experiment will be performed without missing values.

**Class Imbalance** exists when the class prior probability of one class is very small in comparison to another. This is a frequent problem in real-world applications like fraud detection and credit-scoring [13]. Class imbalance has been studied to some extent in the setting of online clustering. Our experiment will make use of data not exhibiting any imbalance in the size of the different classes.

Our experiment makes simplified assumptions on the timing, availability and completeness of the data. While these assumptions might not always hold in the real world they give us the ability to focus on investigating cluster accuracy. Additional research on these topics needs to be done.

# 2 Theory

Before diving further into our research, we will first introduce several concepts specific to data streams as well as different clustering algorithms and deep learning.

## 2.1 Properties of Data Streams

Data streams are usually an ordered sequence of continuous data records. For example data generated by sensor arrays. Data streams are challenging to analyze for a number of different reasons, these aspects need to be taken into account when designing algorithms to analyze data streams:

**Volume** A data stream is a continuous process, possibly infinite in length. A stream mining application must therefore handle unbounded volumes of data. In this scenario not all data can be stored and not all historical data can be used for training. Incremental approaches are required that sequentially update the model as data becomes available.

**Velocity** The speed of arrival of new data is fast and continuous, meaning each item has to be processed in *one pass* and discarded afterwards. There is no way to analyze the same data again as the stream cannot be paused while the algorithm catches up.

**Variability** The flow of data to be processed can be highly inconsistent. Depending on the domain it can have periodic peaks and there is no guarantee in what order the data might arrive.

**Noise** Data streams are prone to noise and missing values. This can be caused by various factors: a data packet lost during transmission, electromagnetic interference or temporary failure of sensors. A particular challenge in the context of evolving data, is to differentiate noise and concept drift. If an algorithm is robust towards noise, it might not be sensible enough to concept drift and react only slowly. The opposite is also true and an algorithm can end up adjusting to noise, falsely identifying it as drift.

**Evolving data** It is important to take into account that the model underlying the data in the stream may change over time. This property of streams, also known as non-stationarity or concept drift, is one of the focus of this thesis and the concept of evolving data is therefore discussed in more details in the following chapter.

## 2.2 Non-Stationarity, Concept Drift and Concept Evolution

Learning in many real world domains is challenging due to a **hidden context**, not given as part of the available features, playing a role in the distribution of the values to be predicted [24]. The distribution of the values to be predicted is also called the target **concept**. A concept can be for example customer preferences, discussion topics or driving behavior. Changes in the hidden context might cause changes in the target concept, this is also referred to as **concept-drift** [24]. When a drift occurs, a model built on old data becomes inconsistent with new data. Clustering becomes very challenging if the hidden context changes while using data recorded over a long period of time. In this case the results will be dominated by outdated history [1]. An example of concept drift is visualized in **Figure 2.1**.

Drifts can occur in different ways as concepts can appear and disappear with different patterns [10]: they can be **sudden**, someone graduating from college might suddenly have different monetary concerns; **reoccuring**, when a previously existing concept reappears after some time, like seasonal consumption patterns; or **gradual**, a sensor's accuracy for example might gradually decrease over time.

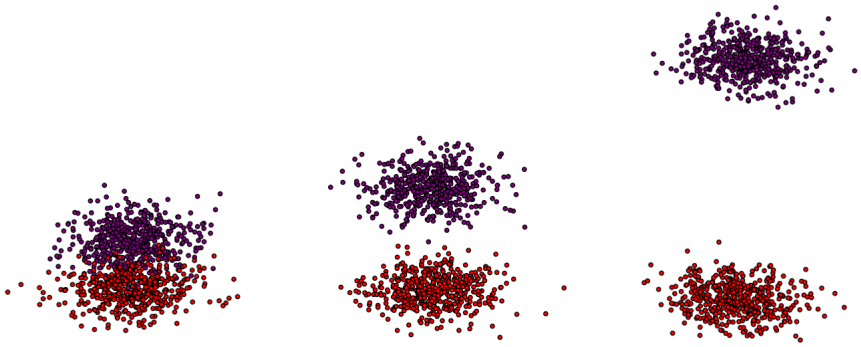


Figure 2.1: *Concept drift: The two classes in the dataset are represented by different colors, while the defining properties of the red class stay the same over time, the purple class exhibits concept drift.*

In addition to changes in the distribution, the number of clusters in a stream might also change over time. Clusters might merge or new ones appear, this is referred to as **concept evolution** [17].

A data stream with concept-drift exhibits a **non-stationary** behavior, as the statistical properties (eg. mean, variance) of the different classes change over time.

## 2.3 Clustering Data Streams

The basic idea behind clustering is to partition the data into different groups where the points in the same group are as similar as possible to each other and as different as possible to points in other groups.

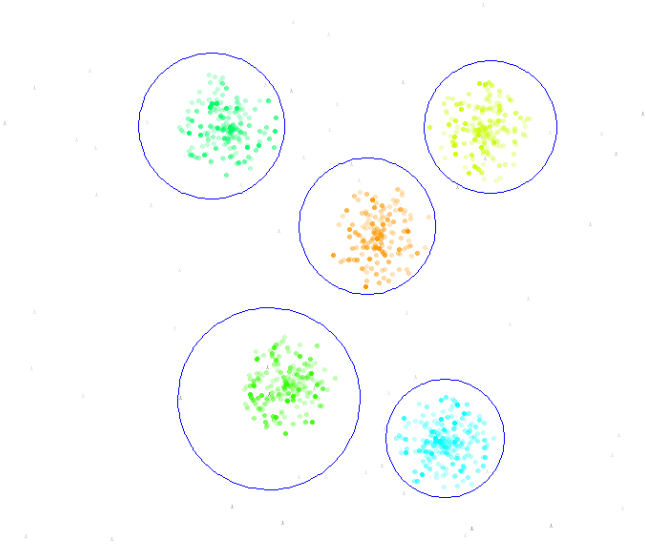


Figure 2.2: *Possible clusters in a dataset visualized by MOA. The colors represent the true classes (the ground truth the clustering algorithm tries to identify), the circles represent the clusters identified by a clustering algorithm.*

This is useful for discovering different classes or types of behavior that were previously unknown. An example could be an image viewer, clustering can be used in order to sort the images by their content without having

any extra information of what the images contain. All images in a cluster might have something in common such as being pictures of a cat, while the images in a different cluster might be pictures of food. What classes exist in the data is not known beforehand and might change over time. Clustering can also be an intermediate step for other data mining problems like classification or outlier analysis [3]. An example for clustering results you might obtain is displayed in **Figure 2.2**

One downside of most clustering algorithms is that it measures the similarity of two points as their distance in the feature space. Two points close to each other in the feature space are similar, while two points far away from each other are different. While it is possible to accurately measure distances, like the euclidean distance, in low dimensional space (2d, 3d) the performance of clustering algorithms degenerates rapidly with increasing dimensions [9]. Hence the need to reduce the dimensionality of high dimensional datasets prior to clustering.

### 2.3.1 Online Learning

While a batch learning method (**Algorithm 1**) can build the model after processing the entire dataset, methods capable of online learning (**Algorithm 2**) update the model incrementally as new instances are processed. They observe a stream of data and make a prediction for each incoming data point with the currently available information, before updating the model. Most algorithms used for handling concept drift are online learning algorithms as with non-stationary data (data that exhibits concept drift) not all data is equally relevant for processing a value.

**Algorithm 1** Simple *batch* learning algorithm

```
for number of epochs do                          ▷ Process the entire dataset multiple times
    for each data point do
        compute weights and bias deltas for current point
        accumulate the deltas
    end for
    adjust weights and bias values using the accumulated deltas
end for
```

**Algorithm 2** Simple *online* learning algorithm

```
for each data point do
    compute weights and bias deltas for current point
    adjust weights and bias values using deltas
end for
```

## 2.4 Deep Learning

Deep learning has gained a lot of attention from the academic community for its state of the art performance in many research domains, from speech recognition to computer vision [21]. Deep learning refers to a group of machine learning techniques, based on Artificial Neural Networks (ANNs), that can automatically learn a hierarchical representation of the data. They achieve that by stacking multiple layers of artificial neurons on top of each other, as visualized in **Figure 2.3**. That way abstract features deeper in the network are defined by less abstract features in the previous layers [1]. **A feature** can be a type of input data or a transformation of the input into some representation that can be better exploited by a machine learning algorithm. In a scenario where we try to automatically recognize faces, a lower level feature could be a simple edge or a corner while a more abstract feature, derived from the previous two could be a mouth or a nose.

Additionally deep learning models work very well with high dimensional data [26]. A specific deep learning method, a Variational Autoencoder (VAE), is used in this thesis. Its features are discussed in the following subsection.

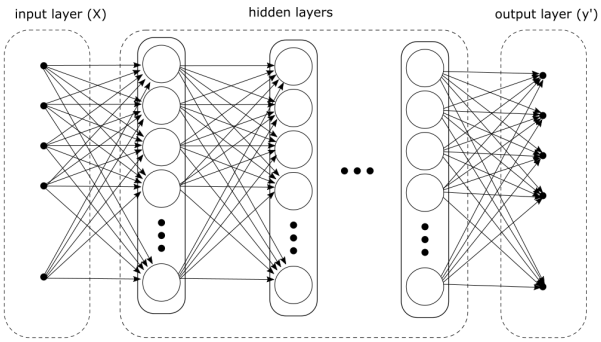



Figure 2.3: *Structure of a Multilayer Perceptron (MLP), a kind of deep neural network. In the case of classification, each output neuron stands for a separate class.*

### 2.4.1 Variational Autoencoders

While a conventional ANN cannot take advantage of unlabeled data and is mostly used for classification problems, an Autoencoder has a deep architecture that makes it capable of learning features from unlabeled data. It achieves this by reconstructing the original input and updating its weights based on the reconstruction error between the original input and the reconstructed version. An Autoencoder consists of an encoder and decoder component (the generative part of the model) as presented in **Figure 2.4**. A special type of Autoencoder is the Variational Autoencoder (VAE) and was introduced by Kingma and Welling [11].

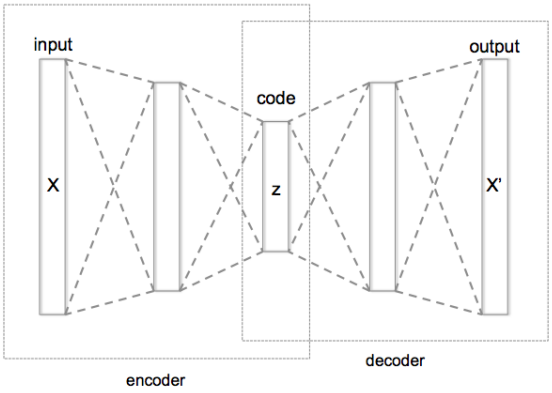



Figure 2.4: *Structure of an Autoencoder, with its two components the encoder and the decoder. In our implementation, each of these components has the structure of an MLP, the latent variables being the output of the encoder and the input for the decoder and the reconstruction being the output of the decoder.*
*Source: by Chervinskii, licensed under CC BY-SA 4.0 [7].*

# 3 Related Work

Of the research on clustering non-stationary data streams, only few papers deal with deep learning models for dimensionality reduction and to our knowledge, no paper compares the performance of several online stream clustering algorithms before and after a dimensionality reduction with a variational autoencoder. In this section, we provide a brief overview of existing work in the field of performance measures for stream clustering as well as a comparison between our research and related papers.

## 3.1 Performance Measures for Stream Clustering

The MOA framework[1] developed by A. Bifet *et al.* [3] is intended to provide an environment for implementing algorithms and running experiments for online learning and non-stationary data stream. S.B. Tartar *et al.* used it for benchmarking various stream clustering algorithms for churn detection [23]. In their review they compare the performance of four different algorithms, CluStream, DenStream, ClusTree and Flockstream. Just as in this thesis, the performance of the different algorithms are compared using several metrics provided in MOA: F1-P, F1-R and Purity. Their work however focused on the topic of churn detection and the datasets used [23] have unbalanced classes, a specific scenario that is not part of the scope of this thesis. Additionally while some of their datasets have many attributes, they do not perform any dimensionality reduction on them.

## 3.2 Deep Online Learning Methods

Several papers apply deep online learning to data stream analysis, R. Calandra *et al.* propose a proof of concept for DBNs called ADBN [4] that can be incrementally trained on non-stationary data streams. While the ADBN is just like the VAE a generative model that can handle high-dimensionality data, the results were not applied to dimensionality reduction and clustering. Additionally no other concept-drifting dataset was used besides MNIST. J. Read *et al.* investigated the use of deep learning methods for stream analysis [19]. They use a restricted Boltzmann machine (RBM) in combination with different popular data stream classifiers like k-nearest neighbors (kNN) and used the MOA framework for their classifier implementations. The RBM would first perform a feature space transformation (dimensionality reduction) similar as in this thesis. However J. Read *et al.* investigate in particular the case of a data stream being partially labeled, while in this thesis all elements of the data stream are assumed to be unlabeled. We therefore compare several clustering algorithms instead of different classifiers and have to use different metrics to evaluate their performance.

[1] MOA: Massive Online Analysis, http://moa.cms.waikato.ac.nz/

# 4 Methodology

This chapter presents the methodology of our thesis which follows the experimentation framework described by Basili *et al.* [2]. First we define the experiment using the framework, then we plan the design and evaluation of the experiment. After planning we describe the operation of the experiment with preparation, execution and analysis. Lastly we describe interpretation and impact of the experiment results.

## 4.1 Experiment Definition

Our motivation for the experiment is to asses the performance of clustering algorithms, with the purpose of evaluating the clustering accuracy in the domain of non-stationary data streams.

We use the same experimental study structure to evaluate all of our hypotheses. We change the clustering algorithm that is the object of the experiment and perform the experiment with and without dimensionality reduction of the data stream.

## 4.2 Experiment Design

As shown in **Figure 4.1** the experiment workflow is as follows: for each collected dataset, first perform clustering evaluation with CluStream, DenStream and ClusTree, while recording the measures described in **Section 4.4**. Then run dimensionality reduction on the dataset using the VAE before performing the clustering evaluation again.

After all datasets have been clustered and the measurements have been recorded, we perform the experiment analysis using statistical models. For hypotheses **H0 F1R**, **H0 F1P** and **H0 Pur** we use a one-way analysis of variance (ANOVA) to evaluate the statistical difference of performance of different clustering algorithms. For hypotheses **H0 VAE-F1R**, **H0 VAE-F1P** and **H0 VAE-Pur** we use a paired $t$–test to evaluate the improvement of clustering performance when performing dimensionality reduction.

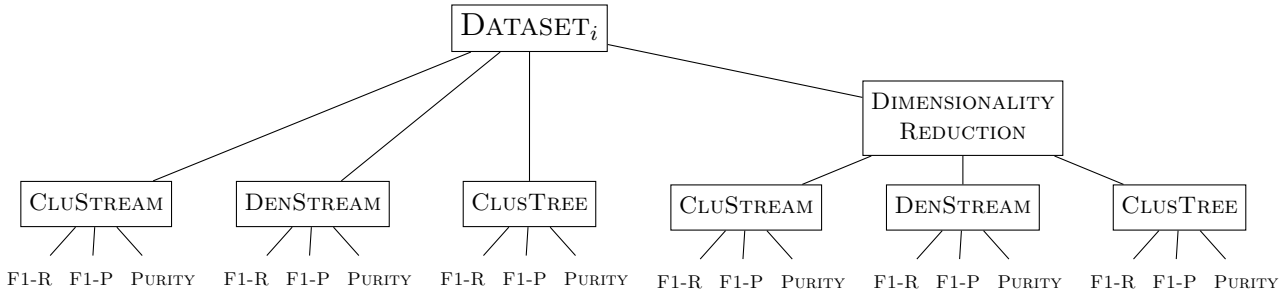



Figure 4.1: *Experiment workflow*

## 4.3 Evaluation Criteria

We will investigate several criteria in order to evaluate the performance of the selected clustering algorithms. The first criteria is the reconstruction accuracy of the Variational Autoencoder, in order to compare the performance of the different clustering algorithms with and without performing a dimensionality reduction we first need to evaluate how accurate the reduction was performed. The second criteria is the accuracy of the model over time. The performance of a clustering algorithm depends upon its ability to group data points in a way that points within a cluster are more similar than points outside that cluster. The average accuracy of the model gives therefore insight over how well an algorithm performed overall on that data. Lastly we

will also evaluate the average additional time it takes to process a new entry when performing dimensionality reduction. This criteria allows us to compare how much overhead the dimensionality reduction process adds in comparison to its benefit. This is important in the context of data streams, where every new data point needs to be evaluated in real time.

## 4.4 Evaluation Measures

Several performance measures exist to evaluate the previously mentioned criteria. The measures are described in detail in the following subsections.

### 4.4.1 Reconstruction Accuracy

The reconstruction accuracy of the Variational Autoencoder, is measured through the loss in the reconstruction as described in **Section 2.4.1**. The higher the loss, the worse the reconstruction of the original data is. We can use the reconstruction loss to assess the performance of the VAE in reducing the dimensionality of the data. The more accurate the reconstructions of the VAE, the better the clustering algorithms should perform.

### 4.4.2 Clustering Accuracy

There are two ways to evaluate the performance of clustering algorithms also known as cluster validation: **internal evaluation**, in which only the information available to the algorithm is used to measure the cluster validity and **external evaluation**. In external evaluation the cluster validity is measured with data not available for clustering, for example predefined labels in the dataset (ground truth). An internal evaluation would on the other hand assign a higher score for an algorithm that produce results with a high similarity within, and a low similarity between clusters. This score however is biased towards algorithms that use a comparable distance measure to calculate the similarity of two data points [8]. For this reason all measures used are external evaluation methods. In order to perform the external evaluation of our models we gathered datasets that already store the correct label for each entry.

Given the correct labels, one way to measure the clustering accuracy of an algorithm is by using the **F1-measure**. The F1-measure is a value between 0 and 1, the higher it is the more accurate our clustering algorithm is performing. The F1-measure has two components: the **precision** and **recall**. The precision looks for the ratio of true positives over true positives and false positives ($TP/TP + FP$), it is high if the algorithm doesn't return many FP. The recall looks at the true positives over true positives and false negatives ($TP/TP + FN$), it is high if the algorithm doesn't return many FN. Combining the two, the F1-measure gives a balanced picture of the overall accuracy.

The MOA framework refers to the F1-measure as **F1-R**. In addition to F1-R, a variant called **F1-P** introduced by Moise *et al.* [18] was also used. The main idea of F1-P is to calculate the total F1-score for each found cluster, instead of for each ground truth class. This gives a more complete picture of the clustering performance as the accuracy of F1-R drops if an incorrect number of clusters is detected, however this does not necessarily mean that the clustering algorithm is not performing well and maybe the ground truth doesn't give the whole picture. The MOA framework also provides the **Purity** measure which was used in this thesis, it is basically the average precision of all clusters and gives us an indication as to how contaminated the clusters are (how many points they contain that they shouldn't).

Let $m$ be the true number of classes in the dataset and $n$ be the number of clusters detected by the model. Let $CR_i$ be one of the clusters detected by the algorithm, while $CS_j$ is one of the true classes. $v_{ij}$ is the number of points assign to cluster $i$ and belonging to class $j$. The precision and recall for a cluster $CR_i$ can then be calculated as follows [23]:

$$precision_{CR_i} = \frac{max(v_{i1}, ..., v_{im})}{\sum_{j=1}^{m} v_{ij}} \qquad recall_{CR_i} = \frac{max(v_{i1}, ..., v_{im})}{\sum_{i=1}^{m} v_{ij}}$$

For the precision, first the maximum overlap between class $j$ and cluster $i$ (which are the TP) is found and then divided by all point within that cluster. For the recall, the FN are the points belonging to class $j$, where the overlap between a cluster $i$ and class $j$ is greatest, but are not in the cluster $i$. From precision and recall we can now calculate the **Purity** for all $n$ clusters and the **F1-measure** for $CR_i$.

$$Purity = \frac{\sum_i^n precision_{CR_i}}{n} \qquad F_1\ Measure_{CR_i} = 2 * \frac{precision_{CR_i} * recall_{CR_i}}{precision_{CR_i} + recall_{CR_i}}$$

With the F1-measure we can finally calculate **F1-P**, which is simply put the average F1-measure across all $n$ clusters. **F1-R** is calculated in a similar fashion, but over the true number of classes instead of the number of clusters detected by the algorithm.

$$F1 - P = \frac{\sum_i^n F1\ Measure_{CR_i}}{n} \qquad F1 - R = \frac{\sum_j^m F1\ Measure_{CS_j}}{m}$$

### 4.4.3 Execution Time

Lastly we measure the execution time of the dimensionality reduction on every new arriving data point. We also calculate the average running time for the VAE on each dataset. These values allow us to estimate how applicable the dimensionality reduction of the VAE would be when handling high velocity data.

## 4.5 Preparation

To prepare the experiment it's required to collect and preprocess non-stationary datasets, implement algorithms for clustering and dimensionality reduction and lastly evaluating what parameters to use for the different algorithms.

### 4.5.1 Data Collection

We require the datasets to be numeric, non-stationary and with both high and low dimensionality. They also need to be preprocessed so that there are no missing values, since that is out of the scope for this thesis. The datasets we collected are listed in **Table 4.1**.

All the synthetic datasets fits our requirements of concept drift well, since they exhibit incremental and gradual changes over time [22, 28]. The non-synthetic datasets also exhibit concept drift [16, 20, 25, 28] but some of them require preprocessing as described in the following section.

### 4.5.2 Data Preprocessing

The datasets in **Table 4.1** have already been preprocessed and most of them can be used as they are, however to be able to view the clustering graphically in MOA the values of all features need to be scaled within the range [0, 1]. We have therefore normalized each dataset as follows, let $n$ be the number of data points, $m$ be the number of features and $x_{ij}$ be the value of the $j^{\text{th}}$ feature for the $i^{\text{th}}$ data point in the dataset, then calculate:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} = \frac{x_{ij} - min(x_{1j}, ..., x_{nj})}{max(x_{1j}, ..., x_{nj}) - min(x_{1j}, ..., x_{nj})}$$

Some datasets required additional processing before normalization. The **GasSensors** dataset contains gas measurements over 36 months divided in 10 batches, in the batches the measurements are sometimes ordered with the same class appearing continuously for over 500 data points. We shuffled each batch to make sure that

more classes can be visible within a window of 1000 data points, the concept drift is still exhibited between the batch files. For **Gestures** we concatenated the 7 files (*_va3.csv) included in the dataset in alphabetical order and changed the class labels to numerical values. For the **EyeState** dataset we removed 4 extreme outliers so that the difference between other datapoints would be more distinguishable after normalization.

Table 4.1: Description of datasets, **Classes** are the true clusters, **Features** can be seen as dimensions and **Length** is the number of data points in the dataset.

| Dataset | Classes | Features | Length | Type | Description |
|---|---|---|---|---|---|
| 1CDT[1] | 2 | 2 | 16,000 | Synthetic | One Class Diagonal Translation |
| 1CHT[1] | 2 | 2 | 16,000 | Synthetic | One Class Horizontal Translation |
| 2CHT[1] | 2 | 2 | 16,000 | Synthetic | Two Classes Horizontal Translation |
| 4CR[1] | 4 | 2 | 144,400 | Synthetic | Four Classes Rotating Separated |
| 4CRE-V1[1] | 4 | 2 | 125,000 | Synthetic | Four Classes Rotating with Expansion V1 |
| 4CRE-V2[1] | 4 | 2 | 183,000 | Synthetic | Four Classes Rotating with Expansion V2 |
| 5CVT[1] | 5 | 2 | 24,000 | Synthetic | Five Classes Vertical Translation |
| 1CSurr[1] | 2 | 2 | 55,283 | Synthetic | One Class Surrounding another Class |
| 4CE1CF[1] | 5 | 2 | 173,250 | Synthetic | Four Classes Expanding and One Class Fixed |
| FG-2C-2D[1] | 2 | 2 | 200,000 | Synthetic | Two Bidim. Classes as Four Gaussians |
| UG-2C-2D[1] | 2 | 2 | 100,000 | Synthetic | Two Bidim. Unimodal Gaussian Classes |
| UG-2C-3D[1] | 2 | 3 | 200,000 | Synthetic | Two 3-dim. Unimodal Gaussian Classes |
| UG-2C-5D[1] | 2 | 5 | 200,000 | Synthetic | Two 5-dim. Unimodal Gaussian Classes |
| MG-2C-2D[1] | 2 | 2 | 200,000 | Synthetic | Two Bidim. Mulitimodal Gaussian Classes |
| GEARS-2C-2D[1] | 2 | 2 | 200,000 | Synthetic | Two Rotating Gears |
| HyperP[2] | 5 | 10 | 100,000 | Synthetic | Hyper Plane stream |
| Keystroke[1] | 4 | 10 | 1,600 | Real | Timing of keystrokes |
| Gestures[3] | 5 | 32 | 9,873 | Real | Gesture phase segmentation |
| GasSensors[4] | 6 | 129 | 13,910 | Real | Gas sensors at different concentrations |
| EyeState[5] | 2 | 15 | 14,976 | Real | EEG measurements with closed and open eyes |

### 4.5.3 Algorithm Implementation

The clustering evaluation is performed using the MOA framework[6] (release 16.04) developed by A. Bifet *et al.* [3]. MOA is an open-source framework software in Java that makes it possible to run experiments on evolving data streams and measure clustering performance. We use the implementations of the clustering algorithms, CluStream [1], DenStream [5] and ClusTree [12], that are included in MOA.

The implementation of the Variational Autoencoder was written in the Python programming language. We chose Python due to the wide range of available machine learning and scientific libraries available. Two libraries used for both the implementation of the algorithm and their performance evaluation where scikit-learn[7] and TensorFlow[8]. Both rely on the scientific computing libraries SciPy[9] and NumPy[10]. For data handling the Python data structure and the data analysis library Pandas[11] were used.

[1] https://sites.google.com/site/nonstationaryarchive/ [22]
[2] http://www.cse.fau.edu/~xqzhu/stream.html [28]
[3] https://archive.ics.uci.edu/ml/datasets/Gesture+Phase+Segmentation [15, 16]
[4] https://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset+at+Different+Concentrations [15, 20, 25]
[5] https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State [15]
[6] http://moa.cms.waikato.ac.nz/
[7] http://scikit-learn.org/
[8] https://www.tensorflow.org/
[9] https://www.scipy.org/
[10] http://www.numpy.org/
[11] http://pandas.pydata.org/

### 4.5.4 Algorithm Parameters

The MOA framework provides recommended parameter settings for all included algorithms. **Table 4.2** lists the default settings together with a parameter description. We leave both of the parameters *horizon* and *evaluateMicroClustering* at their default value during the experiment for all algorithms. Parameters are also provided in MOA to control the display horizon of the data stream and frequency of clustering performance measurements. Both the data stream horizon and evaluation frequency, are set to 1000 by default. We leave those parameters at default except if the data stream contains less then 100,000 data points, in which case we set data stream horizon and evaluation frequency to 100. This is done to save a similar amount of measurements across datasets.

Table 4.2: Default parameter settings in MOA.

| Algorithm | Parameter | Default Value | Description |
|---|---|---|---|
| CluStream | horizon | 1000 | Range of the window |
| | maxNumKernels | 100 | Maximum number of micro kernels to use |
| | kernelRadiFactor | 2 | Multiplier for the kernel radius |
| | evaluateMicroClustering | false | Evaluate micro- instead of macro-clustering |
| DenStream | horizon | 1000 | Range of the window |
| | epsilon | 0.02 | Defines the epsilon neighbourhood |
| | beta | 0.2 | Beta constant |
| | mu | 1.0 | Mu constant |
| | initPoints | 1000 | Number of points to use for initialization |
| | offline | 2.0 | Offline multiplier for epsilon |
| | lambda | 0.25 | Lambda constant |
| | processingSpeed | 100 | Number of incoming points per time unit |
| | evaluateMicroClustering | false | Evaluate micro- instead of macro-clustering |
| ClusTree | horizon | 1000 | Range of the window |
| | maxHeight | 8 | The maximal height of the tree |
| | evaluateMicroClustering | false | Evaluate micro- instead of macro-clustering |

**CluStream** In addition to *horizon* and *evaluateMicroClustering*, CluStream also has the parameters *maxNumKernels* and *kernelRadiFactor*. We could not find that *kernelRadiFactor* had any affect on the performance when we ran CluStream in MOA, therefore we left it at the default value. The parameter *maxNumKernels* on the other hand made some difference. We tested different values between 1 and 2000 for *maxNumKernels* on five of our datasets and found that the best results, with regard to F1-R, occurred between 5 and 800. We ended up with 15 configurations for CluStream where all parameters are at default except for *maxNumKernels*, that uses the following values: $5, 10, 15, 20, 25, 30, 40, 50, 60, 80, 100, 200, 400, 600, 800$.

**DenStream** With 9 different parameters, DenStream is the algorithm that depends most on parameter settings to perform well. Most of the parameters directly affect the clustering results, however when we had found the best possible clustering result by only changing *epsilon*, changing other parameters did rarely improve the clustering results at all. Therefore we only try different values for *epsilon* in the experiment, and leave other constants at default. The other parameter that we change is *initPoints*, which we set to the same value as the evaluation frequency (described in the beginning of this section), so that DenStream produces a clustering result in time for the first evaluation. After trying different values for *epsilon* on most of our datasets, we found that an *epsilon* between 0.005 and 0.2 resulted in the highest F1-R score. For the experiment we use 40 configurations of DenStream with all parameters at default except *epsilon* and *initPoints*, where *initPoints* is equal to the evaluation frequency and the value of *epsilon* lies between 0.005 and 0.2 in steps of 0.005.

**ClusTree** The only parameter except for *horizon* and *evaluateMicroClustering* in ClusTree is *maxHeight*. We tested different values for *maxHeight* on five of our datasets and found that a *maxHeight* over 15 didn't improve clustering results. Therefore we decided to use 15 configurations of ClusTree for experiment where *maxHeight* ranges between 1 and 15.

**Variational Autoencoder** The learning rate for the VAE stays at 0.01 for the whole experiment. However different structures need to be tested against a dataset to be able to find the best configuration. All structures used in the experiment are listed in **Table 4.3**.

Table 4.3: Structures of the VAE used in the experiment.

| | |
|---|---|
| 10-2-10 | 10-4-10 |
| 50-2-50 | 50-4-50 |
| 100-2-100 | 100-4-100 |
| 500-2-500 | 500-4-500 |
| 800-2-800 | 800-4-800 |
| 10-10-2-10-10 | 10-10-4-10-10 |
| 50-50-2-50-50 | 50-50-4-50-50 |
| 100-100-2-100-100 | 100-100-4-100-100 |
| 500-500-2-500-500 | 500-500-4-500-500 |
| 800-800-2-800-800 | 800-800-4-800-800 |

## 4.6 Execution

We start the experiment by performing dimensionality reduction on all datasets in **Table 4.1**. We run the VAE on each dataset using the 20 different structures listed in **Table 4.3**. When dimensionality reduction has completed we can see the cost for each structure, which shows how it performed over time on the dataset. We then look at how each structure performed on average for the dataset and pick the best one. For some datasets we also pick the worst structure to be able to see how it performs in comparison during clustering evaluation.

After dimensionality reduction has been performed, clustering evaluation in MOA takes place. All datasets in **Table 4.1** including their counterparts that have been processed in the VAE, will be clustered with each of the 70 algorithm configurations described in the previous section. Since this amounts to over 2,800 tests, we run the clustering evaluation with a script that calls the command-line interface in MOA, as shown in **Listing 4.1**.

Listing 4.1: Command for running clustering evaluation in MOA using ClusTree with *maxHeight*=12.

```
java -cp moa.jar moa.DoTask EvaluateClustering -l (clustree.ClusTree -H 12) \
        -s (SimpleCSVStream -f GasSensors.csv -c -h 100 -e 100)
```

## 4.7 Analysis

The results of the experiments, the F1-R measure, the F1-P measure and the Purity, are presented in **Table 5.1** for each dataset for all 3 clustering models. It also present the average performance over all datasets combined. We perform a one way ANOVA test on the results, to verify if there are statistical differences to the performances. Lastly we plot the F1-R measure over time of the different models for each dataset, allowing us to visualize how the different models compare to each other and adapt to concept drift, **Figure 5.1**.

The performance of the dimensionality reduction by the VAE is displayed as an average error cost in **Table 5.2** for each dataset. We can use this cost as an indicator to how well the dimensionality reduction was performed. Additionally the cost over time is also plotted to visualize how the VAE performs when subject to concept-drift for all datasets, **Figure 5.2**.

**Table 5.3** displays the results for each dataset of all 3 clustering models after the dimensionality reduction. We finally compare these results in **Table 6.3** by displaying the clustering performance change of the various models. To further investigate the results of the different models, we performed a **statistical analysis**. Using a paired $t$-test we evaluate if there is a statistically relevant difference in the overall accuracy gain or loss of two models, this is done with a confidence interval of 0.95. We plot the clustering performance over time of all 3 clustering models after dimensionality reduction, **Figure 5.3** and **5.3**.

## 4.8 Interpretation

The results of our experiments are intended to give insight into the general performance of several widely used clustering algorithms with and without a dimensionality reduction step by a VAE. The performances are measured in the context of real-time non-stationary data only. The datasets used are a mix of synthetic and real world datasets. This setting allows us to present valuable information on how our selected algorithms perform on non-stationary data in general. This is relevant in many applications for example sensory data, where non-stationary behavior is common. Our results are however only partially relevant to real world applications. As explained in **Section 1.4**, many more factors influence how an algorithm would perform in a real world setting, such as noise and class imbalance.

## 4.9 Impact

The results of this thesis can be easily reproduced, the datasets are freely available online using the referrals in **Table 4.1** and the VAE implementation with scripts to run it is also available[12]. The MOA framework used for evaluating the clustering performance is open source and freely available together with the implementations of the clustering algorithms CluStream, DenStream and ClusTree[13].

[12]https://github.com/au-re/vae
[13]https://github.com/Waikato/moa

# 5 Results

In the following chapter we will present the result of our limited lab experiment. First a comparison of the different clustering algorithms without any dimensionality reduction is presented in **Table 5.1** and plotted in **Figure 5.1**. Then the reconstruction results of the variational autoencoder are display in **Table 5.2** and plotted for a selected number of datasets in **Figure 5.2**. Lastly the results of the clustering algorithms when run on the dim. reduced datasets are displayed in **Table 5.3** and compared to results on the original dataset in **Figure 5.3** and **Figure 5.4**.

## 5.1 Clustering Performance without Dimensionality Reduction

Here are presented the results from our limited lab experiment in regard to RQ1: *"How does the performance of different clustering algorithms differ in the context of non-stationary data streams"*. The clustering algorithms where run with several parameters and the best performing combinations are displayed in **Table 5.1**.

Table 5.1: Comparison of performance for different clustering algorithms. The best result for each dataset, with regard to F1-R, is shown in **bold**.

| | **CluStream** | | | | **DenStream** | | | | **ClusTree** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | F1-R | F1-P | Purity | Clusters | F1-R | F1-P | Purity | Clusters | F1-R | F1-P | Purity | Clusters |
| 1CDT | 0.91 | 0.91 | 1.00 | 2.00 | **0.93** | **0.94** | **0.92** | **1.84** | 0.89 | 0.89 | 1.00 | 2.00 |
| 1CHT | 0.82 | 0.82 | 0.93 | 2.00 | **0.83** | **0.82** | **0.82** | **1.93** | 0.83 | 0.83 | 0.95 | 2.00 |
| 2CHT | 0.54 | 0.49 | 0.60 | 1.99 | **0.66** | **0.69** | **0.54** | **1.06** | 0.52 | 0.51 | 0.58 | 2.00 |
| 4CR | 0.95 | 0.95 | 1.00 | 4.00 | **0.97** | **0.96** | **0.99** | **4.03** | 0.89 | 0.89 | 1.00 | 4.00 |
| 4CRE-V1 | **0.92** | **0.92** | **0.98** | **4.00** | 0.67 | 0.48 | 0.86 | 5.86 | 0.89 | 0.89 | 0.99 | 4.00 |
| 4CRE-V2 | **0.87** | **0.87** | **0.93** | **4.00** | 0.50 | 0.20 | 0.86 | 9.33 | 0.84 | 0.84 | 0.94 | 4.00 |
| 5CVT | 0.79 | 0.79 | 0.91 | 5.00 | 0.34 | 0.36 | 0.74 | 7.09 | **0.82** | **0.82** | **0.90** | **5.00** |
| 1CSurr | 0.82 | 0.82 | 0.89 | 1.98 | 0.65 | 0.76 | 0.63 | 1.05 | **0.83** | **0.83** | **0.90** | **2.00** |
| 4CE1CF | 0.86 | 0.86 | 0.99 | 5.00 | 0.62 | 0.48 | 0.83 | 5.68 | **0.87** | **0.87** | **0.99** | **5.00** |
| FG-2C-2D | 0.62 | 0.59 | 0.79 | 1.99 | 0.63 | 0.85 | 0.75 | 1.00 | **0.65** | **0.62** | **0.77** | **2.00** |
| UG-2C-2D | **0.87** | **0.87** | **0.96** | **2.00** | 0.68 | 0.69 | 0.54 | 1.07 | 0.86 | 0.86 | 0.96 | 2.00 |
| UG-2C-3D | **0.79** | **0.79** | **0.94** | **2.00** | 0.71 | 0.71 | 0.59 | 1.22 | 0.75 | 0.75 | 0.95 | 2.00 |
| UG-2C-5D | 0.56 | 0.56 | 0.94 | 2.00 | **0.66** | **0.66** | **0.54** | **1.11** | 0.48 | 0.48 | 0.94 | 2.00 |
| MG-2C-2D | **0.74** | **0.75** | **0.84** | **2.00** | 0.67 | 0.59 | 0.64 | 1.56 | 0.74 | 0.74 | 0.85 | 2.00 |
| GEARS-2C-2D | **0.80** | **0.80** | **0.92** | **2.00** | 0.66 | 0.66 | 0.52 | 1.04 | 0.80 | 0.80 | 0.92 | 2.00 |
| HyperP | **0.01** | **0.02** | **0.97** | **4.19** | 0.01 | 0.01 | 0.72 | 12.56 | 0.01 | 0.01 | 0.93 | 4.40 |
| Keystroke | 0.30 | 0.48 | 0.63 | 3.00 | 0.37 | 0.46 | 0.31 | 1.00 | **0.40** | **0.62** | **0.90** | **4.00** |
| Gestures | 0.01 | 0.03 | 0.14 | 2.76 | **0.38** | **0.64** | **0.58** | **1.00** | 0.00 | 0.01 | 0.10 | 3.34 |
| GasSensors | 0.00 | 0.02 | 0.19 | 4.42 | **0.13** | **0.25** | **0.90** | **17.31** | 0.00 | 0.01 | 0.09 | 5.20 |
| EyeState | 0.14 | 0.15 | 0.67 | 1.07 | **0.53** | **0.38** | **0.99** | **34.89** | 0.20 | 0.21 | 0.86 | 1.13 |

To better visualize and compare the performance of the different clustering algorithms we have plotted their F1-R score over time in **Figure 5.1**. This also helps to visualize how the algorithms perform when confronted with sudden concept-drifts.

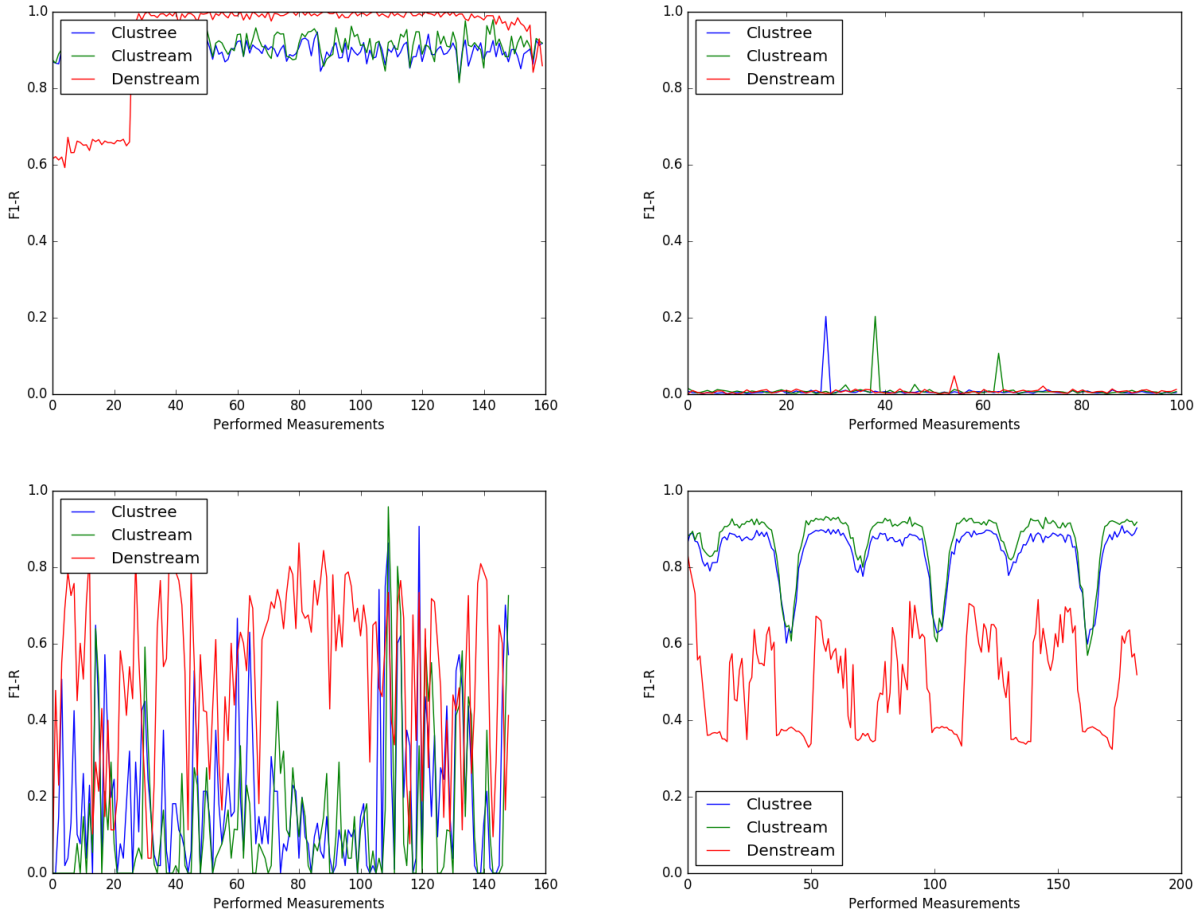



Figure 5.1: *The top left plot presents the F1-R score over time on the dataset **1CDT**, all 3 clustering algorithms perform rather well on low dimensional datasets. As can be seen in the top right plot however, their performance is very different on dataset **hyperP**, on this 10 dimensional dataset the F1-R score stays around 0.0. On the lower left the F1-R score over time on the dataset **EyeState** is presented. With 15 dimensions, the performance of the different clustering algorithms, although very erratic, is higher than on hyperP. This shows how additional factors besides the dimensionality play a role in the clustering accuracy. One such factor is clearly visible in the lower right plot. In dataset **4CRE-V2** the classes overlap periodically and during these moments of overlap the clustering accuracy drops.*

## 5.2 Clustering Performance with Dimensionality Reduction

After having established a baseline for the performance of ClusTree, CluStream and DenStream on the datasets without any dimensionality reduction, we can now present the clustering performance when combined with the VAE. We first investigate how well the VAE can reconstruct the original data and then compare how the clustering algorithms perform on the dimensionality reduced datasets.

### 5.2.1 Reconstruction Accuracy

In this section we present the reconstruction accuracy of the VAE on the different datasets. The loss over time and the average loss gives insight over the reconstruction accuracy of the VAE and how usable it is for dimensionality reduction. The VAE was run with several parameters on all datasets and the runs with the lowest reconstruction loss were kept, these are presented in **Table 5.2**.

Table 5.2: These are the best performing VAE configurations for each dataset and their *average loss* as well as the *average execution time per element*. The lower the loss and the avg. execution time, the better. The structure is defined as the number of nodes per layer and the number of layers. It is displayed as for example 10-2-10, 10 being the number of nodes in the first hidden layer and so on. The input and output layers are not presented as their size depends on the dataset to be processed.

| | **Configuration** | | **Results** | |
|---|---|---|---|---|
| **Dataset** | Structure | Learning Rate | avg. Loss | avg. Time (ms) |
| 1CDT | 50-50-2-50-50 | 0.01 | 1.292 | 3.75 |
| 1CHT | 50-50-2-50-50 | 0.01 | 1.337 | 3.71 |
| 2CHT | 10-10-2-10-10 | 0.01 | 1.324 | 2.36 |
| 4CR | 10-2-10 | 0.01 | 1.388 | 1.85 |
| 4CRE-V1 | 10-2-10 | 0.01 | 1.388 | 2.08 |
| 4CRE-V2 | 10-10-2-10-10 | 0.01 | 1.386 | 2.40 |
| 5CVT | 50-50-2-50-50 | 0.01 | 1.335 | 3.95 |
| 1CSurr | 10-2-10 | 0.01 | 1.374 | 2.28 |
| 4CE1CF | 10-2-10 | 0.01 | 1.388 | 2.18 |
| FG-2C-2D | 10-2-10 | 0.01 | 1.384 | 2.21 |
| UG-2C-2D | 10-2-10 | 0.01 | 1.383 | 2.00 |
| UG-2C-3D | 10-2-10 | 0.01 | 3.383 | 2.00 |
| UG-2C-5D | 10-10-2-10-10 | 0.01 | 1.973 | 3.00 |
| MG-2C-2D | 10-2-10 | 0.01 | 1.379 | 2.00 |
| GEARS-2C-2D | 10-2-10 | 0.01 | 1.388 | 2.00 |
| HyperP | 500-500-2-500-500 | 0.01 | 6.933 | 5.36 |
| Keystroke | 100-100-2-100-100 | 0.01 | 3.427 | 3.00 |
| Gestures | 500-500-2-500-500 | 0.01 | 17.767 | 7.00 |
| GasSensors | 50-50-2-50-50 | 0.01 | 40.419 | 5.18 |
| EyeState | 100-4-100 | 0.01 | 8.669 | 3.97 |

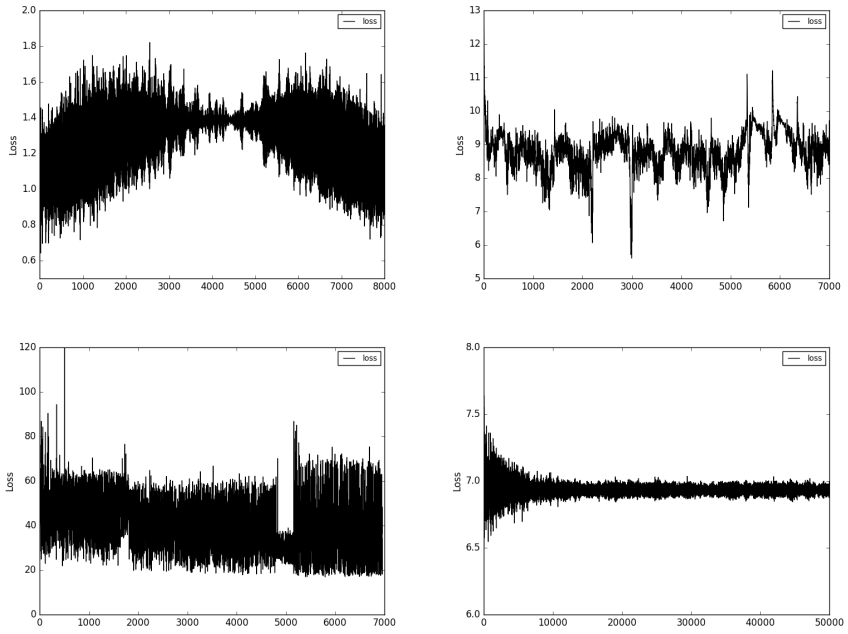



Figure 5.2: *The plots show the reconstruction loss over time of the VAE on different datasets. The top left image shows how the reconstruction loss spreads when the 2 classes of dataset* ***1CDT*** *come close to each other and partially overlap. The top right image shows the reconstruction for the dataset* ***EyeState****, the bottom left for dataset* ***GasSensor*** *and the bottom right for dataset* ***HyperP****.*

### 5.2.2 Clustering Results

In order to answer **RQ2**: *"How do the previously assessed clustering algorithms perform on the same datasets after feature extraction by a deep learning model"*, we run CluStream, DenStream and ClusTree on the dim. reduced datasets with different parameters. The runs with the highest scores are displayed in **Table 5.3**. To illustrate the results we plot first some runs that benefited from the dimensionality reduction in **Figure 5.3** and some that didn't in **Figure 5.4**.

Table 5.3: Comparison of performance for different clustering algorithms, after applying feature extraction of the datasets using the configurations presented in **Table 5.2**. The best result for each dataset, with regard to F1-R, is shown in **bold**.

| | **CluStream** | | | | **DenStream** | | | | **ClusTree** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | F1-R | F1-P | Purity | Clusters | F1-R | F1-P | Purity | Clusters | F1-R | F1-P | Purity | Clusters |
| 1CDT | 0.72 | 0.72 | 0.82 | 2.00 | 0.66 | 0.69 | 0.54 | 1.33 | **0.74** | **0.74** | **0.82** | **2.00** |
| 1CHT | 0.69 | 0.69 | 0.76 | 2.00 | 0.66 | 0.69 | 0.54 | 1.28 | **0.71** | **0.69** | **0.76** | **1.98** |
| 2CHT | 0.62 | 0.61 | 0.65 | 2.00 | **0.66** | **0.69** | **0.54** | **1.22** | 0.61 | 0.61 | 0.65 | 2.00 |
| 4CR | **0.54** | **0.54** | **0.60** | **4.00** | 0.40 | 0.40 | 0.28 | 1.51 | 0.53 | 0.53 | 0.60 | 4.00 |
| 4CRE-V1 | **0.48** | **0.48** | **0.54** | **4.00** | 0.40 | 0.41 | 0.28 | 1.43 | 0.47 | 0.47 | 0.54 | 4.00 |
| 4CRE-V2 | **0.41** | **0.41** | **0.27** | **4.00** | 0.40 | 0.40 | 0.25 | 1.00 | 0.36 | 0.32 | 0.37 | 3.29 |
| 5CVT | 0.33 | 0.35 | 0.44 | 3.13 | 0.33 | 0.50 | 0.33 | 1.17 | **0.33** | **0.36** | **0.41** | **3.35** |
| 1CSurr | 0.64 | 0.72 | 0.65 | 1.97 | **0.65** | **0.78** | **0.63** | **1.06** | 0.58 | 0.57 | 0.67 | 2.00 |
| 4CE1CF | **0.47** | **0.47** | **0.55** | **4.99** | 0.33 | 0.36 | 0.22 | 1.00 | 0.47 | 0.47 | 0.55 | 4.99 |
| FG-2C-2D | 0.63 | 0.85 | 0.75 | 2.00 | **0.63** | **0.86** | **0.75** | **1.00** | 0.53 | 0.52 | 0.74 | 1.96 |
| UG-2C-2D | 0.66 | 0.63 | 0.54 | 2.00 | **0.67** | **0.67** | **0.50** | **1.00** | 0.61 | 0.60 | 0.68 | 2.00 |
| UG-2C-3D | **0.67** | **0.61** | **0.58** | **2.00** | 0.67 | 0.67 | 0.51 | 1.00 | 0.62 | 0.61 | 0.68 | 2.00 |
| UG-2C-5D | 0.66 | 0.66 | 0.51 | 2.00 | **0.67** | **0.67** | **0.51** | **1.00** | 0.57 | 0.49 | 0.59 | 2.00 |
| MG-2C-2D | **0.66** | **0.67** | **0.51** | **2.00** | 0.66 | 0.67 | 0.51 | 1.00 | 0.59 | 0.58 | 0.65 | 2.00 |
| GEARS-2C-2D | 0.64 | 0.64 | 0.72 | 2.00 | **0.66** | **0.65** | **0.52** | **1.53** | 0.63 | 0.63 | 0.71 | 2.00 |
| HyperP | 0.35 | 0.51 | 0.39 | 4.41 | **0.36** | **0.55** | **0.38** | **1.00** | 0.31 | 0.26 | 0.42 | 3.54 |
| Keystroke | 0.37 | 0.31 | 0.39 | 3.94 | **0.40** | **0.39** | **0.25** | **2.00** | 0.38 | 0.33 | 0.37 | 3.44 |
| Gestures | **0.44** | **0.50** | **0.65** | **3.15** | 0.43 | 0.68 | 0.54 | 1.00 | 0.44 | 0.49 | 0.65 | 3.58 |
| GasSensors | 0.36 | 0.40 | 0.50 | 4.44 | 0.29 | 0.42 | 0.30 | 1.15 | **0.37** | **0.38** | **0.50** | **5.64** |
| EyeState | 0.77 | 0.78 | 0.96 | 1.13 | **0.92** | **0.94** | **0.96** | **2.31** | 0.74 | 0.76 | 0.97 | 1.12 |

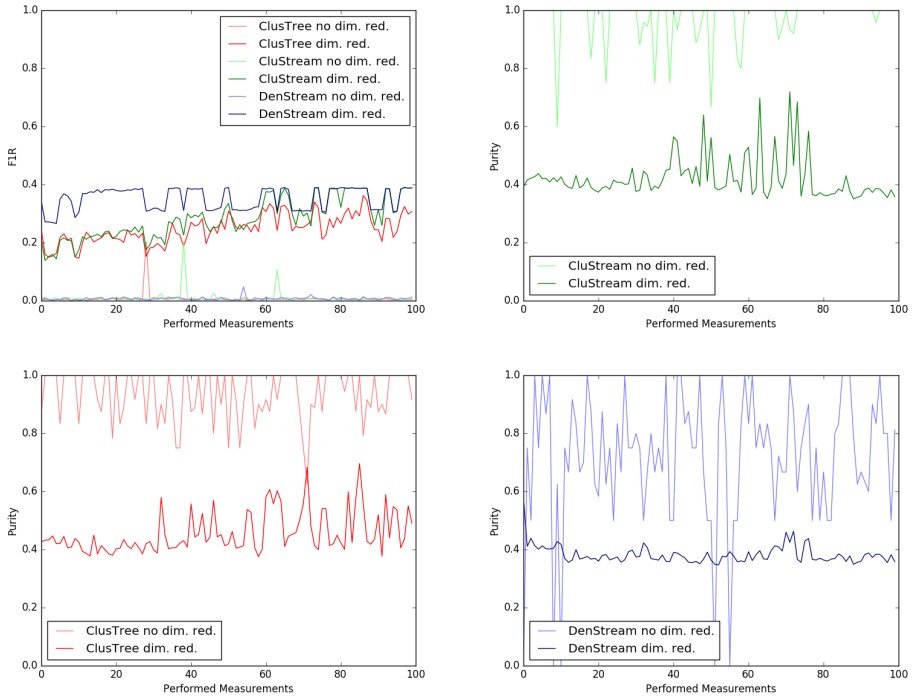



Figure 5.3: *On these plots we visualize the performance difference on the dataset* ***hyperP*** *before and after performing the dim. reduction. On the top left we can see how the different clustering algorithms have improved after the dimensionality reduction. The other 3 plots show how the average Purity of the different clustering algorithms has decreased as a result of the dimensionality reduction. It should however also be noted how the Purity over time becomes much less erratic.*

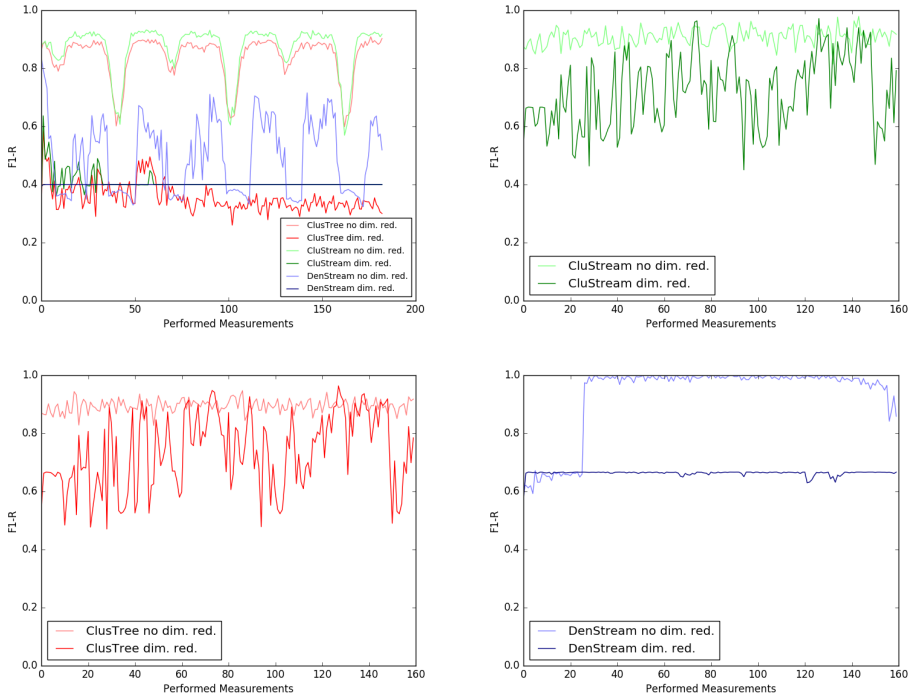



Figure 5.4: *The following plots show the decrease in F1-R score on low dim. datasets. On the top left we can see the worst performance after reconstruction among all datasets. Dataset* ***4CRE-V2*** *seems to be reconstructed with a high amount of noise causing both Denstreams and CluStreams to stagnate at 0.4. The other 3 plots show how the performance decrease on dataset* ***1CDT*** *for the different algorithms.*

# 6 Discussion

In the following sections we will discuss the results of our limited lab experiment in regard to the hypotheses described in **Section 1.3.2**. We first compare the performance of the different clustering algorithms in order to answer **RQ1** before we investigate the performance change caused by the dimensionality reduction, **RQ2**.

## 6.1 Performance Comparison of the Different Clustering Algorithms

From **Table 5.1** we can see which clustering algorithms perform best across the different datasets and different metrics. We extract and summarize that information in **Table 6.1** and run a one way anova test to verify the performance difference among clustering algorithms for statistical significance. The results are as follow: on average across all datasets **CluStream** performs better, followed by first **ClusTree** and finally **DenStream**. This holds for all 3 metrics: F1-R, F1-P and Purity.

With the result presented in **Table 6.2** we cannot reject the null hypotheses **H0 F1R**, **H0 F1P** and **H0 Pur**: there is no statistically relevant difference in the average F1-P, F1-R and Purity between the different clustering algorithms tested.

Table 6.1: Average performance of the different clustering algorithms across all datasets, across all low dimensional datasets (<4D) and across all high dimensional datasets. As we can see all algorithms perform far better on average on the lower dimensional datasets than on the higher dimensional ones.

| | **CluStream** | | | **DenStream** | | | **ClusTree** | | | **All Algorithms** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | F1-R | F1-P | Purity | F1-R | F1-P | Purity | F1-R | F1-P | Purity | F1-R | F1-P | Purity |
| avg. all datasets | **0.611** | **0.624** | **0.818** | 0.566 | 0.576 | 0.707 | 0.597 | 0.607 | 0.798 | 0.591 | 0.602 | 0.774 |
| avg. all low dim. | **0.811** | **0.807** | **0.909** | 0.679 | 0.659 | 0.717 | 0.801 | 0.798 | 0.908 | 0.764 | 0.754 | 0.845 |
| avg. all high dim. | 0.189 | 0.231 | 0.621 | **0.324** | **0.400** | **0.686** | 0.160 | 0.197 | 0.561 | 0.222 | 0.276 | 0.623 |

Table 6.2: To verify that there is a statistical significance in the performance difference between ClusTree, CluStream and DenStream, we perform a one-way ANOVA statistical test on the results of *all datasets*. A confidence interval of 0.95 was selected. No significant results were found.

| **Result** | F1-R | F1-P | Purity |
|---|---|---|---|
| *F-value* | 0.089 | 0.15 | 1.33 |
| *PR F* | 0.915 | 0.861 | 0.273 |

In addition to the average performance across all used datasets it is also interesting to note that the datasets used have a varying number of dimensions. We can therefore further compare the performance of the different algorithms on datasets with high dimensionality as well as on datasets with low dimensionality, this is also presented in **Table 6.1**. On the higher dimensional datasets **DenStream** has a higher F1-R score, F1-P score and Purity score than the two other algorithms. It also has the best performance on 8 of the 20 datasets, more than CluStream or ClusTree. This however comes at the cost of a large number of parameter tweaks involving human oversight and decided upon retrospectively. In the scenario of analyzing stream data this is not always an option and often lower results might be obtained. Additionally in lower dimensional datasets it is **CluStream** that performs best in average.

## 6.2 Performance Change with Dimensionality Reduction

To answer **RQ2** and see if the clustering performance can be improved with the help of a VAE we need to compare the results from **Table 5.1** and **Table 5.3** and see if any significant difference exists. **Table 6.3** displays the change in performance of the algorithms after performing a dimensionality reduction. As expected the performance decreases for datasets that already have a low number of dimensions, this is due to the

reconstruction not being accurate or not providing any benefit for clustering. Reducing dimensions makes obviously more sense in high dimensional space and so, on datasets with a high number of features, we can see clear improvements with a higher average F1-R score by 0.26. There is an improvement of the F1-R and F1-P score across all clustering algorithms, although it seems to benefit ClusTree and CluStream the most as these performed very poorly on high dimensional data. The Purity of the clusters however drops in some cases dramatically. There is also an average reduction in the Purity across the board but it is not too pronounced (-0.09 in total in average) and DenStream is the most affected by it. The lower Purity might be problematic in some scenarios where a high number of false positives are not desired. However it seems to highly depend upon the dataset measured, in some cases the Purity actually improves.

Table 6.3: The change in clustering performance after performing a dimensionality reduction on the datasets. As expected the performance drops on low dimensional datasets and increases on high dimensional datasets. It is not clear to us why the clustering performance on dataset 2CHT doesn't suffer from the reconstruction.

| | **Dimensions** | | **CluStream** | | | **DenStream** | | | **ClusTree** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | Original | Reduced | F1-R | F1-P | Purity | F1-R | F1-P | Purity | F1-R | F1-P | Purity |
| 1CDT | 2 | 2 | -0.19 | -0.19 | -0.18 | -0.27 | -0.25 | -0.38 | -0.15 | -0.15 | -0.18 |
| 1CHT | 2 | 2 | -0.13 | -0.13 | -0.17 | -0.17 | -0.13 | -0.28 | -0.12 | -0.14 | -0.19 |
| 2CHT | 2 | 2 | **0.08** | **0.12** | **0.05** | 0.00 | 0.00 | 0.00 | **0.09** | **0.10** | **0.07** |
| 4CR | 2 | 2 | -0.41 | -0.41 | -0.40 | -0.57 | -0.56 | -0.71 | -0.36 | -0.36 | -0.40 |
| 4CRE-V1 | 2 | 2 | -0.44 | -0.44 | -0.44 | -0.27 | -0.07 | -0.58 | -0.42 | -0.42 | -0.45 |
| 4CRE-V2 | 2 | 2 | -0.46 | -0.46 | -0.66 | -0.10 | **0.20** | -0.61 | -0.48 | -0.52 | -0.57 |
| 5CVT | 2 | 2 | -0.46 | -0.44 | -0.47 | -0.01 | **0.14** | -0.41 | -0.49 | -0.46 | -0.49 |
| 1CSurr | 2 | 2 | -0.18 | -0.10 | -0.24 | 0.00 | **0.02** | 0.00 | -0.25 | -0.26 | -0.23 |
| 4CE1CF | 2 | 2 | -0.39 | -0.39 | -0.44 | -0.29 | -0.12 | -0.61 | -0.40 | -0.40 | -0.44 |
| FG-2C-2D | 2 | 2 | **0.01** | **0.26** | -0.04 | 0.00 | **0.01** | 0.00 | -0.12 | -0.10 | -0.03 |
| UG-2C-2D | 2 | 2 | -0.21 | -0.24 | -0.42 | -0.01 | -0.02 | -0.04 | -0.25 | -0.26 | -0.28 |
| UG-2C-3D | 3 | 2 | -0.12 | -0.18 | -0.36 | -0.04 | -0.04 | -0.08 | -0.13 | -0.14 | -0.27 |
| UG-2C-5D | 5 | 2 | **0.10** | **0.10** | -0.43 | **0.01** | **0.01** | -0.03 | **0.09** | **0.01** | -0.35 |
| MG-2C-2D | 2 | 2 | -0.08 | -0.08 | -0.33 | -0.01 | **0.08** | -0.13 | -0.15 | -0.16 | -0.20 |
| GEARS-2C-2D | 2 | 2 | -0.16 | -0.16 | -0.20 | 0.00 | -0.01 | 0.00 | -0.17 | -0.17 | -0.21 |
| HyperP | 10 | 2 | **0.34** | **0.49** | -0.58 | **0.35** | **0.54** | -0.34 | **0.30** | **0.25** | -0.51 |
| Keystroke | 10 | 2 | **0.07** | -0.17 | -0.24 | **0.03** | -0.07 | -0.06 | -0.02 | -0.29 | -0.53 |
| Gestures | 32 | 2 | **0.43** | **0.47** | **0.51** | **0.05** | 0.04 | -0.04 | **0.44** | **0.48** | **0.55** |
| GasSensors | 129 | 2 | **0.36** | **0.38** | **0.31** | **0.16** | **0.17** | -0.60 | **0.37** | **0.37** | **0.41** |
| EyeState | 15 | 4 | **0.63** | **0.63** | **0.29** | **0.39** | **0.56** | -0.03 | **0.54** | **0.55** | **0.11** |

Table 6.4: Average *performance change* of the different clustering algorithms across all datasets after using the VAE for dimensionality reduction. Note that while DenStream has benefited the least from the dimensionality reduction it still is the best performing algorithm on high dimensional data, even after the reduction.

| | **CluStream** | | | **DenStream** | | | **ClusTree** | | | **All Algorithms** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | F1-R | F1-P | Purity | F1-R | F1-P | Purity | F1-R | F1-P | Purity | F1-R | F1-P | Purity |
| avg. all datasets | -0.06 | -0.05 | -0.22 | -0.04 | 0.03 | -0.25 | -0.08 | -0.10 | -0.21 | -0.06 | -0.04 | -0.23 |
| avg. all high dim. | 0.32 | 0.32 | -0.02 | 0.17 | 0.21 | -0.18 | 0.29 | 0.23 | -0.05 | 0.26 | 0.25 | -0.09 |

Table 6.5: To verify that there is a statistical significance in the performance gain on high dimensional data, we perform a paired t-test. A confidence interval of 0.95 was selected. The significant results are presented in **bold**.

| | **CluStream** | | | **DenStream** | | | **ClusTree** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Result** | F1-R | F1-P | Purity | F1-R | F1-P | Purity | F1-R | F1-P | Purity |
| *p*-value | **0.02135** | **0.02961** | 0.8943 | 0.2616 | 0.142 | 0.2386 | **0.02742** | 0.1167 | 0.7898 |
| *t*-value | **-2.7492** | **-2.5869** | 0.1371 | -1.1899 | -1.599 | 1.2533 | **-2.617** | -1.7434 | 0.2764 |

With the results of the *t*-test (**Table 6.5**), we can partially reject the null hypotheses **H0 VAE-F1R** and

**H0 VAE-F1P** for high dimensional datasets. The improvements of the dimensionality reduction for F1-R and F1-P for DenStream are not significant, but they are for CluStream and in the case of F1-R also for ClusTree. **H0 VAE-Pur** however cannot be rejected as there is no significant improvement in the Purity of the clustering when using a VAE for dimensionality reduction for any algorithm.

While there is a considerable improvement in the performance on high dimensional datasets like *HyperP*, the clustering results are still far from the same as on the low dimensional datasets, it is possible though that with better adjusted parameters we could improve the results of the VAE and the clustering performance.

## 6.3 Threats to Validity

The threats to the validity of our findings are as follows: first, while the selected datasets all exhibit concept drift and are either manufactured to be or are actually recorded from data stream, it is not clear if and to what extent dimensionality reduction is meaningful on them beforehand. There are only six different high dimensional datasets selected, this might be too few to be significant. Additionally the high dimensional datasets all are (except for *HyperP*) real world datasets and could therefore be more challenging for clustering, regardless of their dimensions. Second, the search for good parameters for the VAE was not as extensive as it could have been, as it was too computationally expensive. This means that the performance of the dimensionality reduction could be further improved.

# 7 Conclusion

From this limited lab experiment, our contributions are twofold: we present a side by side comparison of several clustering algorithms, CluStream, DenStream and ClusTree on non-stationary data streams. We cannot, from this comparison, validate the hypothesis that there is a significant difference between the performance of the algorithms in general. However our results indicate that DenStream is the most performant on high-dimensional data but CluStream is more performant on average. Our second contribution is the comparison of the clustering algorithms after adding a VAE for dimensionality reduction. In this case we demonstrate that not all clustering algorithms benefit from the dimensionality reduction in the same way. While there is a significant improvement for CluStream and ClusTree this is not the case for DenStream. Additionally we show that regardless of the clustering algorithm, there is no relevant improvement in the Purity of the clusters after the dimensionality reduction.

# References


[1] C. Aggarwal et al. "A Framework for Clustering Evolving Data Streams". *Proceedings of the 29th VLDB Conference*. Berlin, Germany, 2003, pp. 81–92.

[2] V. Basili, R. Selby, and D. Hutchens. "Experimentation in Software Engineering". *IEEE Transactions on Software Engineering* **12**.7 (July 1986), pp. 733–743.

[3] A. Bifet et al. "MOA: Massive Online Analysis". *Journal of Machine Learning Research* **11** (Aug. 2010), pp. 1601–1604.

[4] R. Calandra et al. "Learning Deep Belief Networks from Non-Stationary Streams". *Proceedings of the 22nd ICANN International Conference on Artificial Neural Networks and Machine Learning*. Lausanne, Switzerland, 2012, pp. 379–386.

[5] F. Cao et al. "Density-Based Clustering over an Evolving Data Stream with Noise". *Proceedings of the 2006 SIAM International Conference on Data Mining*. Bethesda, Maryland, USA, 2006, pp. 328–339.

[6] X.-W. Chen and X. Lin. "Big Data Deep Learning: Challenges and Perspectives". *IEEE Access* **2** (May 2014), pp. 514–525.

[7] Chervinskii. *Schematic picture of an autoencoder architecture*. 2015. URL: `https://commons.wikimedia.org/w/index.php?curid=45555552`.

[8] I. Färber et al. "On Using Class-Labels in Evaluation of Clusterings". *Proceedings of the 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings, held at SIGKDD'10*. Washington DC, USA, 2010.

[9] A. Hinneburg and D. A. Keim. "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering". *Proceedings of the 25th VLDB Conference*. Edinburgh, Scotland, UK, 1999, pp. 506–517.

[10] T. R. Hoens, R. Polikar, and N. V. Chawla. "Learning from streaming data with concept drift and imbalance: an overview". *Progress in Artificial Intelligence* **1**.1 (Jan. 2012), pp. 89–101.

[11] D. Kingma and M. Welling. "Auto-Encoding Variational Bayes". *Proceedings of the 2nd International Conference on Learning Representations*. Banff, Canada, 2014.

[12] P. Kranen et al. "The ClusTree: Indexing Micro-clusters for Anytime Stream Mining". *Knowledge and Information Systems* **29**.2 (Nov. 2011), pp. 249–272.

[13] G. Krempl et al. "Open Challenges for Data Stream Mining Research". *ACM SIGKDD Explorations Newsletter* **16**.1 (Sept. 2014), pp. 1–10.

[14] Y. LeCun, Y. Bengio, and G. Hinton. "Deep Learning". *Nature* **521** (May 2015), pp. 436–444.

[15] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[16] R. C. B. Madeo, C. A. M. Lima, and S. M. Peres. "Gesture Unit Segmentation Using Support Vector Machines: Segmenting Gestures from Rest Positions". *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. Coimbra, Portugal, 2013, pp. 46–52.

[17] M. M. Masud et al. "Addressing Concept-Evolution in Concept-Drifting Data Streams". *Proceedings of the 10th IEEE International Conference on Data Mining*. Sydney, Australia, 2010, pp. 929–934.

[18] G. Moise, J. Sander, and M. Ester. "P3C: A Robust Projected Clustering Algorithm". *Proceedings of the 6th IEEE International Conference on Data Mining*. Hong Kong, China, 2006, pp. 414–425.

[19] J. Read, F. Perez-Cruz, and A. Bifet. "Deep Learning in Partially-labeled Data Streams". *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. Salamanca, Spain, 2015, pp. 954–959.

[20] I. Rodriguez-Lujan et al. "On the calibration of sensor arrays for pattern recognition using the minimal number of experiments". *Chemometrics and Intelligent Laboratory Systems* **130** (Jan. 2014), pp. 123–134.

[21] J. Schmidhuber. "Deep Learning in Neural Networks: An Overview". *Neural Networks* **61** (Jan. 2015), pp. 85–117.

[22] V. M. A. Souza et al. "Data Stream Classification Guided by Clustering on Nonstationary Environments and Extreme Verification Latency". *Proceedings of the 2015 SIAM International Conference on Data Mining*. Vancouver, Canada, 2015, pp. 873–881.

[23] S. B. Tatar et al. "Benchmarking Stream Clustering for Churn Detection in Dynamic Networks". *Proceedings of the 18th International Conference on Discovery Science*. Banff, Canada, 2015, pp. 284–298.

[24] A. Tsymbal. *The problem of concept drift: definitions and related work*. Tech. rep. Trinity College Dublin, Apr. 2004.

[25] A. Vergara et al. “Chemical gas sensor drift compensation using classifier ensembles”. *Sensors and Actuators B: Chemical* **166–167** (May 2012), pp. 320–329.
[26] A. Vieira and N. Barradas. “A training algorithm for classification of high-dimensional data”. *Neurocomputing* **50** (Jan. 2003), pp. 461–472.
[27] X. Wu et al. “Data Mining with Big Data”. *IEEE Transactions on Knowledge and Data Engineering* **26**.1 (Jan. 2014), pp. 97–107.
[28] X. Zhu. *Stream Data Mining Repository*. 2010. URL: `http://www.cse.fau.edu/~xqzhu/stream.html`.