Going Offline: An Evaluation of the Offline Phase in Stream Clustering

Philipp Jahn[1,2] (✉), Walid Durani[1,2], Collin Leiber[3], Anna Beer[4], and Thomas Seidl[1,2]

[1] LMU Munich, Munich, Germany. {jahn,durani,seidl}@dbs.ifi.lmu.de
[2] Munich Center for Machine Learning (MCML), Munich, Germany.
[3] Aalto University, Espoo, Finland collin.leiber@aalto.fi
[4] University of Vienna, Vienna, Austria. anna.beer@univie.ac.at

Abstract. Data streams are a challenging and ever more relevant setting for clustering methods as more data arrives faster and faster. Stream clustering strategies either determine the clusters in an online manner directly as the instances appear, or they employ an offline phase where the online summarization structures are processed to obtain a clustering result. A recent analysis found that offline clustering may often be unnecessary or even counterproductive. The methods used in the offline phase are usually fixed for each stream clustering approach and typically stem from only a handful of clustering techniques. In this paper, we perform a broad experimental analysis specifically targeting the offline phase of stream clustering. We analyze several ways of extracting information from the summarization structures, including a novel strategy based on data generation. Ultimately, we showcase that an offline phase is an impactful design choice for stream clustering. We also find that the chosen offline method significantly impacts the clustering performance, with the clustering quality improving drastically for some settings. Our code is available at https://github.com/PhilJahn/SCOPE/.

Keywords: Clustering · Streaming Data · Unsupervised Learning · Offline Phase

1 Introduction

In this modern world, the amount of data is ever-increasing. Many applications continue to produce data infinitely, and as such, methods to handle the ongoing inflow of data are highly demanded [7]. For data streams without a foreseeable end, the amount of data becomes too large to store, which led to the development of stream learning methods such as stream clustering [4, 40]. As with clustering of static data, the goal is to find groupings of data where similar data points are assigned to the same groups and dissimilar data points are kept separate.

A popular strategy for dealing with the constraints of the stream setting is to split the clustering process into an online and an offline phase, where the online phase deals with the ongoing stream, and the offline phase determines

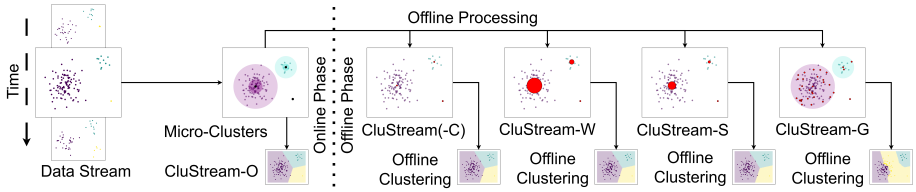

Fig. 1: Data Stream Processing: While the stream data is visualized, it is unavailable once it has been added to the micro-clusters. For micro-clusters, the inner circle represents the deviation radius r, and the outer one the assignment radius r^+. The red points with black borders are reconstructed data points with the size corresponding to the number of data points. The CluStream variants are different processing strategies and will be explained in Section 3.

the actual clustering [2]. Most stream clustering approaches are limited to a single specific offline clustering approach [14]. A recent study [61] has called offline processing into question and has shown that it may be unneeded or even disadvantageous. This study included leaving out the offline processing for two preexisting stream clustering approaches, the partitioning StreamKM++ [1] and the density-based DenStream [13]. Aside from this, they applied k-Means++ [6] and DBSCAN [20], the offline algorithms used in StreamKM++ and DenStream, respectively, to their own approach using micro-clusters, resulting in the pure online summarization outperforming the offline-processed method.

However, evaluating the usefulness of clustering in the offline phase based on the performance of only a few classical clustering methods can be misleading, raising the question of whether the limitations generally lie in the offline processing or merely in the specific selected offline clustering method. Due to the typical lack of strict time constraints [14], the offline phase is also more suitable for optimization than the online phase, which can not be replayed and is more resource-constrained. Thus, in this paper, we investigate the clustering step in the offline phase by incorporating a wide variety of offline clustering methods into a preexisting stream clustering approach. We compare them among each other and against the approach without any offline clustering. As the previous work only examined partitioning and density-based techniques in this context [61], our examination also focuses on these categories. In this study, CluStream [2] serves as a representative, as it is one of the most well-known stream clustering algorithms and the basis [14] built upon by other widely used stream clustering methods like StreamKM++ [1] and DenStream [13]. Figure 1 visualizes our study design. To the best of our knowledge, we are the first to investigate the offline phase of stream clustering with a diverse set of offline clustering algorithms. For the examined datasets, our results have shown that if the offline processing is adjusted, it is possible to boost the performance of the stream clustering method significantly. In one experiment, swapping the offline clustering algorithm for CluStream resulted in almost a doubling of performance compared to the typical configuration of CluStream. Interestingly, using CluStream with the same

underlying clustering as some competitor techniques outperforms them in multiple cases. As such, future evaluations of online summarization techniques should be done with awareness of this factor, as a correctly chosen offline algorithm may dominate the performance of the overall evaluation. Based on CluStream as a leading example, our main contributions are:

- We investigate 12 different offline clustering approaches as processing techniques for the offline phase.
- We introduce a novel way of replicating data from online summaries.
- We investigate and compare multiple ways of processing the summaries.
- We perform an extensive evaluation in the context of other stream clustering approaches and pure online summarization as clustering.

2 Related Work

2.1 Clustering

Clustering methods group similar data objects together while separating dissimilar objects. There are many types of offline clustering algorithms [47] that handle different concepts of data similarity. Here, we put our focus on partitioning and density-based clustering approaches. While there are many more techniques, like hierarchical and grid-based methods, we do not cover them in this work.

Partitioning Clustering Partitioning approaches split the data into distinct partitions. Their most prominent representative is k-Means [37], which finds k clusters by iteratively assigning each point to its closest center and updating the k centroids. It has led to many derived techniques. SubKMeans [41] performs a dimensionality-reduction simultaneously with the clustering. A major focus is the automatic determination of the cluster number k. X-Means [49] does this k-estimation by optimizing information criteria. Projected Dip-Means (P-DipM) [15] applies a test for unimodality on multiple different one-dimensional projections. Aside from these k-Means-based approaches, Spectral Clustering (SC) [38] is a popular approach that transfers the task to a k-dimensional representation based on the similarity graph. Other clustering algorithms, though usually k-Means, can be applied to obtain the final clustering. SpectACl [28] incorporates the average density into the process of SC.

Density-based Clustering In density-based clustering, clusters are dense regions with many data points that are separated by areas of lower point density. A common variant is searching for density-*connected* clusters, such as in DBSCAN [20], where clusters are comprised of density-connected points within dense areas. However, DBSCAN cannot find clusters with varying levels of density. Later innovations have attempted to address this issue. HDBSCAN [12] uses hierarchical clustering concepts to do so. RNN-DBSCAN (RNN-DBS) [11] uses reverse nearest neighborhood sizes to define dense points and uses neighborhood relations rather than distances to handle connectivity. In contrast to these density-connectivity-based approaches, there are also methods based on detecting density-peaks, e.g., DPC [52]. Density peaks are characterized by high

local density and large distances to other candidates for density peaks. These peaks serve as centroids for the clusters. SNN-DPC [36] introduces an allocation strategy based on the incorporation of shared nearest neighbors. DBHD [18] uses local density to determine clusters with the help of the k-nearest neighborhood while allowing for some density variation within clusters.

2.2 Stream Clustering

The stream setting adds many additional constraints to tasks like clustering [4]. Stream data may be infinite and arrive continuously in an uncontrollable order. As such, storage is only possible in a limited manner. Additionally, the underlying distribution of the data can change over time [23]. Thus, stream clustering requires both a summarization structure and a window or forgetting mechanism [61]. Summarization structures maintain temporary information about data points but may differ in what information is saved. As with regular clustering, stream clustering offers a diverse set of different strategies and underlying concepts [4, 40, 14, 65]. In this paper, we focus solely on partitioning and density-based methods. A wide array of methods outside these constraints exist, such as hierarchical [51] and grid-based [57] stream clustering techniques.

Most stream clustering methods can be grouped into either online or online-offline methods [40]. Online methods directly return the clusters upon processing the data. In contrast, online-offline methods perform postprocessing, typically with an offline clustering method, either at fixed timesteps or based on user input. Unlike online processing, offline processing is usually not time-critical [14]. The chosen offline clustering method impacts the resulting cluster shapes [4] and is often not specific to a method. Instead, it could also be combined with different online methods [14], though generally, this is not addressed by stream clustering approaches. Instead, most approaches describe one specific offline processing behavior that usually corresponds to one of a few clustering approaches (e.g., k-Means [37], DBSCAN [20]) or a modification thereof. Kranen *et al.* [31] note that the summaries of their method ClusTree could be used with any offline clustering algorithm, including both partitioning and density-based approaches. However, they did not evaluate this aspect in their experiments. In their follow-up approach LiarTree [32] they do not elaborate further on this aspect or specify the offline clustering approach.

CluStream CluStream [2] is a fundamental stream clustering method which had a major influence on subsequent stream clustering approaches [14]. CluStream maintains a set of micro-clusters to summarize regions of interest. The radius r of a micro-cluster is based on the deviation of the assigned points. Here, there are differences depending on the implementation. For MOA [10], the maximum deviation value for any dimension is used, whereas River [43] uses the average across all dimensions. Data point assignment occurs if a data point falls within a maximum boundary factor $mbf \cdot r$ of the centroid. We call this assignment radius r^+. A new micro-cluster is created if a new point falls outside of this range for their closest micro-cluster. To maintain a limited number of micro-clusters, they are merged based on the closest pairs. Alternatively, micro-clusters that

have not received a new data point for too long are aged out. The micro-clusters are then employed in the offline phase to determine the actual clusters using weighted k-Means (Wk-Means) [37]. CluStream has led to the development of multiple related techniques. Spark-CluStream [8] employs a weighted sampling of the micro-cluster centroids. While some variants change the micro-cluster processing procedure, most perform a k-Means-based offline clustering, with some introducing k-estimation techniques [5]. Still, the recent DynamicCluStream [3] instead merges overlapping micro-clusters to form clusters.

Competitors As competitors for this work, we included a collection of both traditional and recent stream clustering approaches. STREAMKmeans [46] is a variant of STREAMLSEARCH [46], where the k-median clustering is replaced with k-Means. It works on the basis of performing clustering chunk-wise and then storing the resulting centers. The final clustering is produced using a weighted clustering on a fixed number of these centers. DenStream [13] incorporates the density-connectivity principles of DBSCAN [20] into an online-offline stream clustering framework. Micro-clusters mimic the typical hierarchical structure of DBSCAN based on the weight and radius of the micro-cluster. The weights of micro-clusters decay over time. Points are assigned based on a distance-parameter ε that is also used for the DBSCAN clustering in the offline phase. DBSTREAM [26] tracks the number of data points that would be assigned to each pair of micro-clusters. These values are then used to merge micro-clusters into full clusters in the offline phase, rather than a traditional offline clustering algorithm. EMCStream [66] is a recent method that works by clustering UMAP [42] embeddings with k-Means [37]. EMCStream includes a rewind function when concept drift is detected, though the rewind is limited to a set amount of input data. MCMSTStream [19] is another recent approach that represents the stream by micro-clusters with a fixed radius. Clusters are determined through minimum spanning trees [24], where micro-clusters within a certain range of another are included in the same cluster. The recent GB-FuzzyStream [62] introduces a two-level architecture of fuzzy summarization structures, called Fuzzy Granular-Balls (FGBs) and Fuzzy Micro-Balls (FMBs). FGBs summarize incoming data, whereas FMBs are formed through the combination of FGBs. In the offline phase, DPC [52] is applied on the centers of the FMBs to determine the clusters.

2.3 Data Generation

Many different types of data generation methods are available that generate new synthetic instances based on given instructions, often in the form of sampling distributions. These distributions either receive their own label or are combined to form more complex clusters [30]. While commonly applied in benchmarking and evaluation, in the case of SMOTE [16], data generation is used to deal with imbalanced data by artificially increasing the number of minority class samples. Especially relevant for this work is SMOClust [17], a stream-based variant of the SMOTE approach. SMOClust makes use of the online summaries of stream clustering approaches to generate new samples for the minority class.

3 Offline Handling for CluStream

Offline Processing The offline phase of CluStream works by taking the centroids of the micro-clusters and using them as input for a traditional clustering algorithm. The reported cluster label for a specific centroid then serves as the label for the whole micro-cluster. Regular CluStream uses Wk-Means with the micro-cluster weights as weights. Since we want to evaluate the impact of various offline clustering algorithms, we substitute Wk-Means with 12 different clustering algorithms selected from both traditional methods and more recent techniques. These are put into contrast with **CluStream-O**, a CluStream variant that directly uses the micro-clusters determined in the online phase for the cluster assignment rather than performing an offline phase. When incorporating other offline clustering algorithms into the base CluStream, the process is similar to Wk-Means. The centroids of the micro-clusters are used as prototypes to reconstruct the real data points when training the offline clustering algorithms. We refer to the form of CluStream, which directly replaces Wk-Means and uses only the centroids as inputs, as **CluStream-C** (Centroid CluStream). Unlike k-Means, most offline clustering approaches do not offer a weighted variant and thus need other ways to introduce weights to fully exploit the online summaries.
Weighted Data Reconstruction As with Spark-CluStream [8], a fake weighting scheme can be introduced independently of the choice of offline clustering method on the side of the data reconstruction technique. A naive approach is to multiply the centroids by the weight of the micro-cluster. This would correspond to a true weighted clustering. We refer to this approach as **CluStream-W** (Weighted CluStream). Furthermore, we include **CluStream-S** (Sampled CluStream), which is similar to Spark-CluStream, where we sample a limited number of points from the centroids based on their relative weight ratios rather than the absolute values. All centroids are present in the reconstructed data at least once, so no micro-cluster is lost due to the sampling. The label assignment uses the clustering label determined for the closest centroid.
Generative Processing Additionally, we introduce a generative approach to produce suitable input data for offline clustering algorithms. As the micro-clusters offer both a centroid and a radius, we can use them as definitions for distributions for synthetic data generation. Since the real distribution of the data points within a micro-cluster is no longer available, we instead use a uniform distribution over the hypersphere defined by the micro-cluster (see, e.g., [30]). As with River [43], we define r as the average standard deviation across all dimensions. Since points assigned to a micro-cluster may lie outside of this micro-cluster radius r, the assignment radius $mbf \cdot r = r^+$ is instead used as the radius of the generating hypersphere. This approach is denoted **CluStream-G** (Generative CluStream) and generates a limited number of data points. The centroid of every micro-cluster is also included, meaning that, as with CluStream-S, all micro-clusters are represented. The mapping of data points to clusters in CluStream happens on a micro-cluster basis. Every evaluated point is simply assigned to the closest micro-cluster. The label of that micro-cluster is then used as the label for the data point. However, the micro-clusters may not be homogenous,

Table 1: Properties of used datasets

Name	Key	Type	Shuffled?	# Dim.	# Obj.	k	One-off?
Complex-9 [9]	Comp-9	Synthetic	Yes	2	3031	9	No
DENSIRED-2 [30]	DEN-2	Synthetic	Yes	2	5000	11	Yes
DENSIRED-10 [30]	DEN-10	Synthetic	Yes	10	5000	11	No
DENSIRED-100 [30]	DEN-100	Synthetic	Yes	100	5000	11	Yes
RBF-3 40000[5]	RBF-3	Synthetic	No	2	40000	8	No
Fertility-vs-Income[6] [59]	FvI	Real-World	No	2	4014	2	Yes
Electricty [54]	Elec	Real-World	No	8	45312	2	Yes
KDDCUP99 [54]	KDD99	Real-World	No	41	494021	23	No
Gas Sensor Array [54]	Gas	Real-World	No	128	13910	6	No
Star Light Curves [54]	SLC	Real-World	No	1024	9236	3	Yes

adding some level of granularity to this. When using generative approaches, the points are not just focused on the centroid of each micro-cluster, but are instead more spread out. As a result, the clustering approaches can produce different labels for different reconstruction points even if they are created from the same micro-cluster. These points are then used to label the data rather than just the centroid. The assigned labels for CluStream-G are based on the labels of the nearest neighbor within the generated dataset rather than just considering the centroids as done in regular CluStream. Thus, the final step is comparable to the nearest neighbor classifier [21]. A more formalized description of the CluStream variants can be found in the supplementary file.

4 Experiments

4.1 Experiment Setup

We present the properties of all used datasets in Table 1, though some were only one-off datasets used for either Section 4.5 or Section 4.6. Some datasets (Complex-9 and DENSIRED variants) were shuffled for the experiments as they were presorted based on the ground truth labels. All other datasets were not shuffled and retain any present concept drift [23]. We examine the effects of this drift using the FvI dataset in Section 4.5. We used the River stream learning library[7] [43] as a base for our evaluation. We used their implementations of DenStream, DBStream, and STREAMKmeans, as well as a modified version of their CluStream code. EMCStream[8], MCMSTStream[9] and GB-

[5] `https://github.com/CIG-UFSCar/DS_Datasets/tree/master`, last accessed 25.02.2025, based on data generation from MOA [10]

[6] Dataset created from data from the Gapminder data repository `https://www.gapminder.org/data/`, last accessed: 05.06.2025

[7] `https://github.com/online-ml/river`, last accessed: Feb 20th, 2025

[8] `https://gitlab.com/alaettinzubaroglu/emcstream`, last accessed: Feb 20th, 2025

[9] `https://github.com/senolali/MCMSTStream`, last accessed: Feb 20th, 2025

FuzzyStream[10] were adapted from the linked repositories. k-Means(++), SC, DBSCAN and HDBSCAN were from scikit-learn[11] [48]. SubKMeans, X-Means and Projected Dip-Means were from ClustPy[12] [34]. SpectACl[13], SNN-DPC[14] and DPC[15] were taken from the linked repositories. Our implementations of RNN-DBS and DBHD are available in our repository. There was a 7-day time limit for experiments, which was exceeded by some approaches.

4.2 Parameter Selection

We used the AutoML approach SMAC3[16] [35] to look for the best parameters for both the CluStream variants and the competitors. We apply the same parameters across the full stream and optimize based on batches, circumventing the problems arising from using traditional AutoML for stream clustering methods [14]. We downsample large datasets for AutoML, as recommended (e.g., in [64]). As in prior research on AutoML for clustering [53], we give a fixed time limit for all approaches. All experiments were executed on the same hardware (Intel(R) Xeon(R) CPU E5-4640 v4 @ 2.10GHz). For CluStream variants, we split the time budget so that 20% of the time was spent on the online phase and 80% on the offline phase. Most CluStream variants use the parameters obtained from the same online phase optimizations for comparability and use 100 micro-clusters. CluStream-O variants with different micro-cluster numbers are optimized separately. CluStream-S and CluStream-G generate around 1000 points for each evaluation batch. We use the ground-truth cluster number for methods requiring an input cluster number. For more details, see the supplementary file.

4.3 Metrics

For evaluating the clustering results, we used adjusted rand index (ARI) [29] and adjusted mutual information (AMI) [45]. Both metrics have a maximum value of 1, where higher is better. We use the sum of both to optimize the results of the parameter selection. For fairness, the metrics are calculated in batches of 1000, and the mean value over 5 seeds is reported for the best-performing parameters.

Many definitions for the similarity between datasets have been proposed [56]. For the paper, we focus on strategies that do not require ground-truth labels. Bag of prototypes (BoP)[17] [58] is a method of representing datasets by determining patches within the dataset. The Jensen-Shannon divergence [22] between the histogram for the patches in the original dataset and the histogram produced by the assignment of the points of the projected compared dataset is

[10] `https://github.com/xjnine/GBFuzzyStream`, last accessed: Feb 20th, 2025
[11] `https://scikit-learn.org`, last accessed: Feb 20th, 2025
[12] `https://github.com/collinleiber/ClustPy`, last accessed: Feb 20th, 2025
[13] `https://bitbucket.org/Sibylse/spectacl`, last accessed: Feb 20th, 2025
[14] `https://github.com/liurui39660/SNNDPC`, last accessed: Feb 20th, 2025
[15] `https://github.com/colinwke/dpca`, last accessed: Feb 20th, 2025
[16] `https://github.com/automl/SMAC3`, last accessed 30.01.2025
[17] `https://github.com/Klaus-Tu/Bag-of-Prototypes`, last accessed: 02.12.2024

then used as a similarity score. Furthermore, the Maximum Mean Discrepancy (MMD) is also a common method for comparing distributions [25, 50]. We used the PyTorch-based implementation from [60][18] to compute the MMD, which, by default, examines the distributions using five different kernel functions. Aside from distributions, Classifier Two-Sample Tests [63, 50] are also used to assess the similarity between datasets. Here, a binary nearest-neighbor classifier [21] is trained to discriminate between the data of both datasets. We distinguish between the leave-one-out accuracy using the real data points (CA_r) and the offline points (CA_o) based on which points the accuracy is calculated for. We also report the average distance of the real data points to the closest offline data point (NNd) and the average distance of an offline data point to its closest real data point. The scores are calculated in batches of 1000, and the weighted average score is reported. Smaller values denote a closer resemblance of data. The purity of a clustering [39] is the accuracy if each cluster is assigned its majority ground-truth label. We treat all unique offline data points as potential cluster labels and report the purity for assigning the real data points to their closest offline data point. This value represents a hard boundary for the performance of the stream clustering algorithm, as the CluStream-based approaches can not distinguish clusters beyond this. To maintain the logic of smaller values denoting a better score, we instead track the Impurity (Imp.), defined as $1-$Purity. Purity has also been used as the primary metric for evaluating the offline phase in a prior work [61]. However, we opted for ARI and AMI, as the usage of Purity requires a restriction to a similar number of clusters to allow for a meaningful comparison, and we aimed for multiple methods without a specified number of clusters. Still, we report the Purity performance in our supplementary file.
For evaluating the cluster evolution, we use the Temporal Silhouette index (TS)[19] [59] and the Cluster Mapping Measure (CMM) [33]. TS is an unsupervised cluster validation index designed to handle concept drift, whereas CMM is a supervised evaluation metric designed for stream data. We use the objects within the smallest hypersphere that covers all objects with the same label as the ground truth clusters. We include our implementation of the CMM in our repository. While TS is calculated for the full clustering with the most suitable clusters per batch based on the overlap mapped to the ground truth, CMM is calculated in batches of 1000, and the weighted average is reported.

4.4 Performance Impact of the Offline Phase

The metrics for best-performing parameters when directly replacing Wk-Means in CluStream (CluStream-C) and all competitors are given in Table 2. Colored cells denote the relationship in performance between CluStream-O and other approaches, where red/blue means it is worse/better than the best variant of CluStream-O, with saturation indicating the relative performance. The results show that the chosen offline strategy has a strong impact on CluStream's performance. This can be seen best for Comp-9, where the best-performing offline

[18] `https://github.com/jindongwang/transferlearning`, last accessed: 08.03.2025
[19] `https://github.com/CN-TU/py-temporal-silhouette`, last accessed: 05.06.2025

Table 2: Mean metric scores over 5 seeds for evaluated datasets for best-performing parameters according to the sum of ARI and AMI (×100). The best scores are marked as **bold**, and the second-best scores are underlined.

Name	Comp-9		DEN-10		RBF-3		KDD99		Gas	
	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI
STREAMKmeans	41.6	61.6	28.8	52.6	68.4	74.6	**94.0**	**87.0**	11.6	17.0
DenStream	50.0	70.5	60.0	73.2	63.3	69.9	79.2	75.9	35.3	53.0
DBSTREAM	56.9	69.3	67.8	74.9	69.6	76.1	92.5	85.2	26.3	50.1
EMCStream	57.3	80.4	60.3	73.9	53.6	66.3	81.6	76.8	35.1	41.7
MCMSTStream	65.3	74.5	**73.7**	**84.7**	74.8	78.1	90.3	82.7	16.4	38.4
GB-FuzzyStream	20.0	57.5	19.8	43.4	31.0	51.4	-	-	5.8	20.4
CluStream-O - var. k	48.7	66.9	53.2	70.7	65.2	73.2	89.5	83.5	27.4	50.1
CluStream-O - fixed k	41.6	64.7	16.0	34.3	60.2	71.2	87.1	80.6	25.5	37.8
CluStream-O - k=100	9.5	53.0	49.7	69.8	41.9	60.3	80.3	67.5	24.2	50.5
CluStream - Wk-Means	36.8	62.8	54.5	67.9	75.3	78.4	89.7	77.5	32.0	45.2
CluStream-C - k-Means	37.6	63.4	14.4	37.2	73.5	77.8	90.4	79.6	24.7	39.4
CluStream-C - SubKMeans	35.7	61.3	35.9	53.9	73.5	77.5	90.4	79.6	24.2	38.7
CluStream-C - X-Means	49.9	66.7	26.6	47.3	76.3	79.3	90.2	79.6	29.8	52.4
CluStream-C - P-Dip-M	0.0	0.0	0.0	0.0	18.3	24.4	89.6	79.0	12.7	20.3
CluStream-C - SC	47.7	72.8	49.6	64.7	**76.8**	79.0	91.2	81.4	31.6	45.3
CluStream-C - SpectACl	60.8	79.0	55.9	74.5	66.5	76.4	89.6	80.0	26.4	39.0
CluStream-C - DBSCAN	**73.4**	**86.5**	52.8	70.3	63.7	77.1	90.6	78.0	26.5	50.9
CluStream-C - HDBSCAN	71.9	85.7	58.3	73.2	72.0	**79.6**	90.7	80.5	34.7	51.4
CluStream-C - RNN-DBS	37.0	69.2	9.5	22.1	65.1	71.9	87.6	79.8	32.4	49.0
CluStream-C - DPC	45.0	75.7	56.5	70.2	69.3	76.7	92.1	83.7	31.5	52.2
CluStream-C - SNN-DPC	46.3	68.0	25.6	49.4	59.0	69.2	86.1	77.6	29.6	47.0
CluStream-C - DBHD	52.3	75.9	57.7	69.9	73.4	78.5	88.4	79.4	**35.6**	**54.0**

clustering technique has almost double the ARI score of the default case. CluStream-C, with a properly chosen offline phase method, often beats the competing strategies. Still, it is noticeable that the best offline strategy depends on the dataset. This lines up with the statement that the offline phase method determines the final cluster shapes [4]. Specifically, density-based approaches work best for datasets with arbitrarily shaped clusters (like Comp-9). DEN-10 has multiple density levels, leading to more difficulty for DBSCAN. In contrast, centroid-based techniques perform well for RBF-3, where the dataset consists of multiple moving distributions. Despite CluStream originally working with Wk-Means, a density-based offline clustering often results in a better performance.

We also compared the results against CluStream-O with different settings for cluster count k. Clustream-O var. k allows for a flexible choice of clusters beyond the ground-truth cluster number, which was used for CluStream-O fixed k and all other approaches requiring the cluster number. CluStream-O with k=100 uses the same micro-clusters as CluStream-C. In the results we see that CluStream-O is a viable strategy that can outperform CluStream when using an offline phase,

especially for the default setup of CluStream. This reinforces the observations made by Wang *et al.* [61]. For CluStream, the micro-clusters already roughly adhere to the same ideas as k-Means, where points are assigned to centroids that adjust based on the data. The online phase also has the advantage of accessing real data points, whereas the offline phase can only use summaries. Despite this, an appropriate choice of the offline clustering algorithm still often leads to CluStream outperforming CluStream-O. For all examined datasets, there is at least one configuration that manages to perform better than the best CluStream-O variant, though these are not necessarily the same variants on all datasets.

When using the same underlying clustering methods as CluStream-C, some competitors can have a worse performance. Specifically, DenStream is similar to CluStream, but introduces concepts of DBSCAN in its summaries. Despite its less specialized micro-clusters, CluStream-C outperforms DenStream when using DBSCAN in the offline phase for three of the five datasets in Table 2. Still, on one of these three, DenStream outperforms the default CluStream variant. Thus, stream clustering methods should also be compared using the same offline methods, as the final performance is impacted by the chosen offline method.

4.5 Cluster Evolution

Concept drift [23] and subsequent cluster evolution [55, 33] are important factors for the real-world application of stream clustering [44]. To examine the cluster evolution, we drew on prior research [59]. Like them, we use the FvI dataset to showcase this and aimed to use the same parameters they did. They provide the code they used for evaluating the real datasets in their repository[20]. We set the parameters for TS [59] to the default parameters described in their paper, which also correspond to the ones they used in their code. These are a window-size w of 100, the number of neighbors k to 1000, and ς to 1. CMM [33] requires a neighborhood size k, which was set to 5. In the description of CMM, this parameter is noted to only have a minor impact when varied between 1 and 10.

The clustering results for several stream clustering algorithms are presented in Figure 2. Here, we selected the best run among five runs with default parameters and five with optimized parameters, based on the sum of ARI and AMI. The mapping to the ground truth clusters was done based on the largest overlap within each batch of size 1000. The chosen offline algorithm has a significant impact on the ability to handle cluster evolution. A major factor appears to be the ability of methods to determine the cluster number on their own. Here, these estimations often lead to overestimation, which negatively impacts the TS index. Still, oversegmenting appears to have the advantage with CMM. This behavior can be attributed to CMM performing its own matching and, as such, handling the additional cluster labels without an additional malus. As the oversegmented clusters may be comparatively pure, this leads to high performance with CMM. This again highlights the importance of aligning metrics and goals during evaluation and of selecting offline clustering methods that suit the problem setting.

[20] `https://github.com/CN-TU/py-temporal-silhouette`, last accessed: 05.06.2025

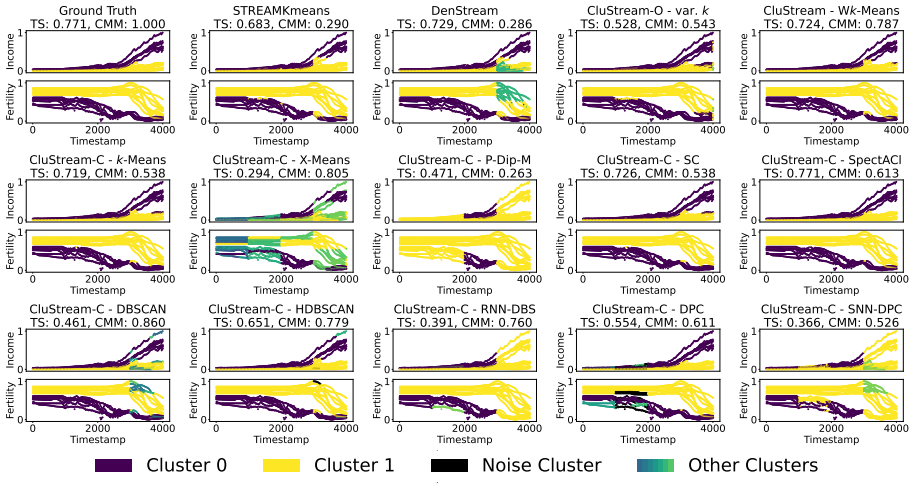

Fig. 2: Clustering behavior of the best-performing run according to ARI and AMI for selected stream clustering algorithms on the FvI dataset

4.6 Offline Replication Quality

In Table 3, some values for centroid-based offline approaches (CluStream-C (CS-C), CluStream-W (CS-W), and CluStream-S (CS-S)) are the same, and thus reported only once. These are the metrics that cover the assignment of real data points to offline data points. Here, CluStream-G always outperforms the other approaches. This is inherently the case as CluStream-G includes all centroids. Nonetheless, the Impurity score showcases a significantly higher ceiling for the clustering performance. However, with increasing dimensionalities, the gap between CluStream-G and the other approaches in terms of the nearest neighbor-

Table 3: Dataset replication scores for CluStream offline clustering inputs for default parameters (×100). The best scores are marked in **bold**.

Key	CS-C	CS-W	CS-S	Centroid			CluStream-G			
	iNNd	iNNd	iNNd	NNd	CA_r	Imp.	iNNd	NNd	CA_r	Imp.
Comp-9	**0.88**	0.97	0.96	2.62	86.0	0.10	1.07	**1.19**	**46.5**	**0.00**
DEN-2	0.82	**0.58**	0.59	1.85	86.4	4.98	0.80	**0.84**	**57.3**	**0.60**
DEN-10	7.52	**4.03**	4.12	7.90	86.2	9.28	5.89	**6.54**	**84.8**	**0.80**
DEN-100	**0.27**	11.25	10.72	27.14	89.1	37.56	13.05	**26.37**	**88.9**	**13.54**
RBF-3	0.79	**0.49**	0.51	1.86	86.5	4.60	0.58	**0.75**	**47.2**	**2.30**
Elec	**2.82**	3.06	3.05	6.91	85.3	18.39	3.86	**5.94**	**79.3**	**9.19**
KDD99	4.22	**1.93**	2.03	3.21	69.2	0.37	2.12	**3.14**	**68.9**	**0.12**
Gas	4.11	**3.82**	3.84	7.99	78.5	6.11	3.96	**7.90**	**78.0**	**1.63**
SLC	**26.50**	45.42	45.09	100.09	80.1	13.71	45.50	**99.89**	**80.0**	**6.93**

Table 4: Additional dataset replication scores for CluStream offline clustering inputs for default parameters (×100). The best scores are marked as **bold**.

Name	CluStream-C			CluStream-W			CluStream-S			CluStream-G		
	CA_o	BoP	MMD	CA_o	BoP	MMD	CA_o	BoP	MMD	CA_o	BoP	MMD
Comp-9	**0.6**	7.2	**0.6**	98.8	9.5	0.7	99.6	9.0	**0.6**	53.4	**2.7**	**0.6**
DEN-2	**0.0**	18.3	37.0	97.0	14.5	**3.4**	98.5	14.1	3.5	60.1	**5.1**	3.6
DEN-10	**0.2**	29.3	109.1	92.6	11.7	**5.6**	97.7	11.5	5.7	82.8	**8.4**	5.8
DEN-100	**0.0**	39.0	214.4	87.5	29.7	9.2	98.0	**29.0**	6.4	98.0	**29.0**	**6.0**
RBF-3	**0.0**	20.2	15.1	97.1	13.9	1.0	98.3	13.6	**0.9**	53.4	**3.1**	**0.9**
Elec	**0.1**	8.4	2.6	99.6	9.4	1.3	99.8	9.0	**1.2**	89.2	**6.4**	**1.2**
KDD99	**1.1**	34.3	124.8	92.9	11.4	**23.0**	98.9	11.3	23.6	97.0	**10.8**	23.6
Gas	**0.3**	20.2	42.6	97.2	13.7	**1.4**	98.8	13.5	1.5	98.8	**13.0**	1.5
SLC	**0.6**	**22.9**	36.1	91.4	31.4	7.4	97.1	30.6	**7.1**	97.1	30.4	**7.1**

hood diminishes. This behavior is expected due to the Curse of Dimensionality as the generated points are typically close to the hypersphere boundaries [27].

For the iNNd score in Table 3, CluStream-G consistently performs worse than other approaches as it spreads its offline points further. Especially for higher dimensionalities, this can lead to some points being far apart from real data points. For CA_o in Table 4, using CluStream centroids without weighting returns the best results. Since there is only one replicated point per micro-cluster, it only requires a real data point to be closer to a centroid than another micro-cluster. When evaluating the distributions, which are covered by BoP and MMD in Table 4, CluStream-G again performs well, though it does not outperform other techniques for all examined datasets. However, even when performing worse than other approaches, the scores are still fairly close to the best-performing ones. Depending on the dataset, the weighting can be both advantageous and disadvantageous. Although weighting allows for better mimicking of the real data regarding point presence, the lack of weight reduction over time can lead to deviations from the current real distributions during later timesteps.

4.7 Performance Impact of the Offline Replication

Another aspect to evaluate is the impact of the offline replication technique on the clustering performance. An overview of the results for the best-performing parameters can be seen in Figure 3. As CluStream does not include decay of the micro-cluster weights, they accumulate for streams where micro-clusters survive for a long time and summarize many data points. This could lead to large sizes of the generated data for CluStream-W, which can result in scalability issues and does not align with the constraints of stream clustering, as the weights could grow infinitely. Despite most clearly approximating a weighted clustering, it is often outperformed by the other weighted approaches or CluStream-C.

Although CluStream-G generally results in a better representation of the real data according to the dataset similarity metric, this does not necessarily result

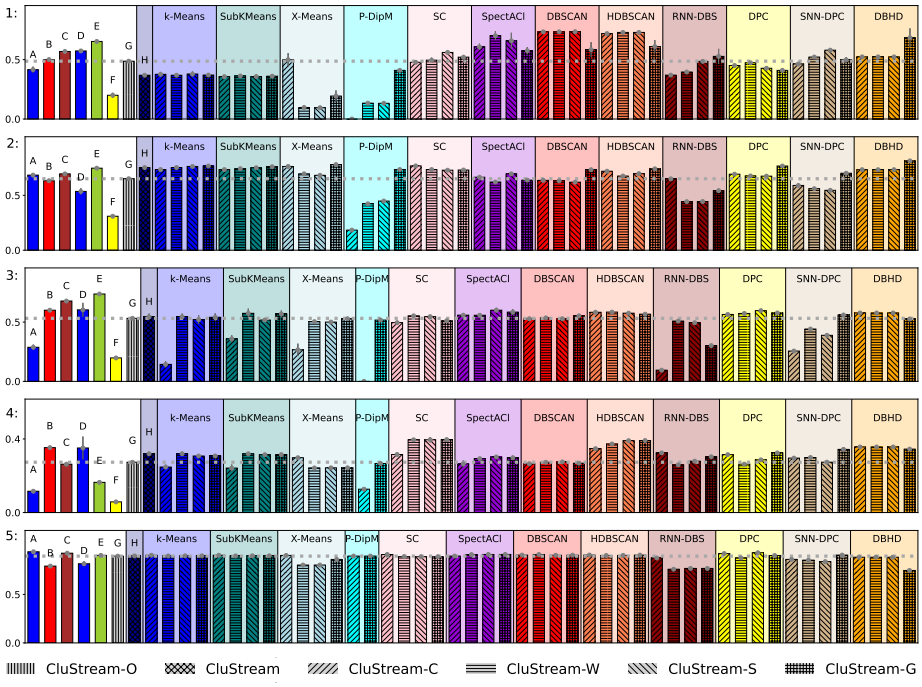

Fig. 3: Mean ARI results over 5 seeds for the datasets Complex-9 (1), RBF-3 40000 (2), DENSIRED-10 (3), Gas Sensor Array (4), and KDDCUP99 (5) for the best-performing runs of STREAMKmeans (A), DenStream (B), DBSTREAM (C), EMCStream (D), MCMSTStream (E), GB-FuzzyStream (F), CluStream-O (G) (also marked by the horizontal line), as well as the CluStream variants (denoted by hatch) for the respective offline clustering algorithms, including the default case of Wk-Means (H). The color indicates the offline clustering. The standard deviation over the different seeds is denoted by gray bars at the top.

in a better clustering performance. The introduction of weights appears to be advantageous for many methods, though it can lead to mixed results for some approaches, which is particularly noticeable in RNN-DBS and X-Means. For the case of density-connectivity approaches, the introduction of weights translates to a shift from connecting singleton micro-clusters to connecting primarily within the micro-clusters themselves for higher-weight micro-clusters for CluStream-S. This can be interpreted as a shift between a global representation for singleton micro-clusters, where the broader structure is made clearer, and a more local representation when weights are introduced, leading to some trade-offs. For CluStream-G, the higher spread can lead to individual generated points becoming their own clusters while the overall distances between micro-clusters are reduced, making them harder to separate for density-connectivity approaches. Still, it can also lead to connections between micro-clusters with a large radius

that belong to the same ground-truth cluster. For density-peak and partitioning methods, the introduction of weights helps differentiate between micro-clusters in terms of relevance. Generating the data points also allows for a better mimicking of the distributions. This is most noticeable for P-DipM, which improves significantly due to a better representation of the distributions used in the unimodality test. In contrast, P-DipM consistently failed during the parameter optimization for approaches that only multiply the centroids for higher-dimensional data.

5 Discussion and Conclusion

We performed a broad evaluation of the offline phase on the example of CluStream, investigating different offline clustering strategies from different categories. We also investigated different approaches to reconstruct the data from summaries, which support this offline clustering. We tackled the observation made by Wang *et al.* [61] that the offline phase may be superfluous in many cases. With suitable parameters and offline clustering approaches, it is possible to outperform the purely online CluStream as well as many competitors. However, it is also important to note that there is no universal solution that always performs best. Although a generative reconstruction offers the best representation based on our evaluation, it does not necessarily lead to the best clustering performance in all cases. It is sensible to treat both the offline reconstruction and the actual offline clustering algorithm as additional constraints for evaluation. Rather than using a single fixed offline approach specific to each stream clustering algorithm, it instead is better to treat the offline phase as something flexible where the same offline approach is either used for all strategies or the performance is measured across a broader spectrum of offline strategies. In the future, we want to expand the evaluation to more online summarization principles, especially those that include weight decay, an aspect that CluStream lacks.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: StreamKM++: A clustering algorithm for data streams. ACM J. Exp. Algorithmics **17**(1) (2012)
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB. pp. 81–92. Morgan Kaufmann (2003)
3. Ahsani, S., Yousef Sanati, M., Mansoorizadeh, M.: DynamicCluStream: An algorithm based on clustream to improve clustering quality. International Journal of Web Research **6**(2), 77–87 (2023)
4. de Andrade Silva, J., Faria, E.R., Barros, R.C., Hruschka, E.R., de Leon Ferreira de Carvalho, A.C.P., Gama, J.: Data stream clustering: A survey. ACM Comput. Surv. **46**(1), 13:1–13:31 (2013)

5. de Andrade Silva, J., Hruschka, E.R.: A support system for clustering data streams with a variable number of clusters. ACM Trans. Auton. Adapt. Syst. **11**(2), 11:1–11:26 (2016)
6. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: SODA. pp. 1027–1035. SIAM (2007)
7. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: PODS. pp. 1–16. ACM (2002)
8. Backhoff, O., Ntoutsi, E.: Scalable online-offline stream clustering in apache spark. In: ICDM Workshops. pp. 37–44. IEEE Computer Society (2016)
9. Barton, T.: Clustering-benchmark repository. `https://github.com/deric/clustering-benchmark` (2015)
10. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
11. Bryant, A., Cios, K.J.: RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. IEEE Trans. Knowl. Data Eng. **30**(6), 1109–1121 (2018)
12. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: PAKDD (2). Lecture Notes in Computer Science, vol. 7819, pp. 160–172. Springer (2013)
13. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SDM. pp. 328–339. SIAM (2006)
14. Carnein, M., Trautmann, H.: Optimizing data stream representation: An extensive survey on stream clustering algorithms. Bus. Inf. Syst. Eng. **61**(3), 277–297 (2019)
15. Chamalis, T., Likas, A.: The projected dip-means clustering algorithm. In: SETN. pp. 14:1–14:7. ACM (2018)
16. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
17. Chiu, C.W., Minku, L.L.: Smoclust: synthetic minority oversampling based on stream clustering for evolving data streams. Mach. Learn. **113**(7), 4671–4721 (2024)
18. Durani, W., Mautz, D., Plant, C., Böhm, C.: DBHD: density-based clustering for highly varying density. In: ICDM. pp. 921–926. IEEE (2022)
19. Erdinç, B., Kaya, M., Senol, A.: MCMSTStream: applying minimum spanning tree to kd-tree-based micro-clusters to define arbitrary-shaped clusters in streaming data. Neural Comput. Appl. **36**(13), 7025–7042 (2024)
20. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. pp. 226–231. AAAI Press (1996)
21. Fix, E.: Discriminatory analysis: nonparametric discrimination, consistency properties, vol. 1. USAF school of Aviation Medicine (1985)
22. Fuglede, B., Topsoe, F.: Jensen-shannon divergence and hilbert space embedding. In: ISIT 2004. Proceedings. p. 31. IEEE (2004)
23. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 44:1–44:37 (2014)
24. Graham, R.L., Hell, P.: On the history of the minimum spanning tree problem. IEEE Ann. Hist. Comput. **7**(1), 43–57 (1985)
25. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.J.: A kernel two-sample test. J. Mach. Learn. Res. **13**, 723–773 (2012)
26. Hahsler, M., Bolaños, M.: Clustering data streams based on shared density between micro-clusters. IEEE Trans. Knowl. Data Eng. **28**(6), 1449–1461 (2016)

27. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition. Springer Series in Statistics, Springer (2009)
28. Hess, S., Duivesteijn, W., Honysz, P., Morik, K.: The SpectACl of nonconvex clustering: A spectral approach to density-based clustering. In: AAAI. pp. 3788–3795. AAAI Press (2019)
29. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**, 193–218 (1985)
30. Jahn, P., Frey, C.M.M., Beer, A., Leiber, C., Seidl, T.: Data with density-based clusters: A generator for systematic evaluation of clustering algorithms. In: ECML/PKDD (7). Lecture Notes in Computer Science, vol. 14947, pp. 3–21. Springer (2024)
31. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: The ClusTree: indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. **29**(2), 249–272 (2011)
32. Kranen, P., Reidl, F., Villaamil, F.S., Seidl, T.: Hierarchical clustering for real-time stream data with noise. In: SSDBM. Lecture Notes in Computer Science, vol. 6809, pp. 405–413. Springer (2011)
33. Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., Pfahringer, B.: An effective evaluation measure for clustering on evolving data streams. In: KDD. pp. 868–876. ACM (2011)
34. Leiber, C., Miklautz, L., Plant, C., Böhm, C.: Benchmarking deep clustering algorithms with clustpy. In: ICDM (Workshops). pp. 625–632. IEEE (2023)
35. Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., Hutter, F.: SMAC3: A versatile bayesian optimization package for hyperparameter optimization. J. Mach. Learn. Res. **23**, 54:1–54:9 (2022)
36. Liu, R., Wang, H., Yu, X.: Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Inf. Sci. **450**, 200–226 (2018)
37. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–136 (1982)
38. von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
39. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press (2008)
40. Mansalis, S., Ntoutsi, E., Pelekis, N., Theodoridis, Y.: An evaluation of data stream clustering algorithms. Stat. Anal. Data Min. **11**(4), 167–187 (2018)
41. Mautz, D., Ye, W., Plant, C., Böhm, C.: Towards an optimal subspace for k-means. In: KDD. pp. 365–373. ACM (2017)
42. McInnes, L., Healy, J.: UMAP: uniform manifold approximation and projection for dimension reduction. CoRR **abs/1802.03426** (2018)
43. Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T., Bifet, A.: River: machine learning for streaming data in python. J. Mach. Learn. Res. **22**, 110:1–110:8 (2021)
44. Namitha, K., Kumar, G.S.: Concept drift detection in data stream clustering and its application on weather data. Int. J. Agric. Environ. Inf. Syst. **11**(1), 67–85 (2020)
45. Nguyen, X.V., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. J. Mach. Learn. Res. **11**, 2837–2854 (2010)
46. O'Callaghan, L., Meyerson, A., Motwani, R., Mishra, N., Guha, S.: Streaming-data algorithms for high-quality clustering. In: ICDE. pp. 685–694. IEEE Computer Society (2002)

47. Oyewole, G.J., Thopil, G.A.: Data clustering: application and trends. Artif. Intell. Rev. **56**(7), 6439–6475 (2023)
48. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011)
49. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: ICML. pp. 727–734. Morgan Kaufmann (2000)
50. Plesovskaya, E., Ivanov, S.: An empirical analysis of KDE-based generative models on small datasets. Procedia Computer Science **193**, 442–452 (2021)
51. Rajagopalan, A., Vitale, F., Vainstein, D., Citovsky, G., Procopiuc, C.M., Gentile, C.: Hierarchical clustering of data streams: Scalable algorithms and approximation guarantees. In: ICML. Proceedings of Machine Learning Research, vol. 139, pp. 8799–8809. PMLR (2021)
52. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492–1496 (2014)
53. da Silva, M.C., Licari, B., Tavares, G.M., Junior, S.B.: Benchmarking automl clustering frameworks. In: AutoML Conference 2024 (ABCD Track) (2024)
54. de Souza, V.M.A., dos Reis, D.M., Maletzke, A.G., Batista, G.E.A.P.A.: Challenges in benchmarking stream learning algorithms with real-world data. Data Min. Knowl. Discov. **34**(6), 1805–1858 (2020)
55. Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., Schult, R.: MONIC: modeling and monitoring cluster transitions. In: KDD. pp. 706–711. ACM (2006)
56. Stolte, M., Kappenberg, F., Rahnenführer, J., Bommert, A.: Methods for quantifying dataset similarity: a review, taxonomy and comparison. Statistics Surveys **18**(none) (Jan 2024). https://doi.org/10.1214/24-ss149
57. Tareq, M., Sundararajan, E.A., Harwood, A., Bakar, A.A.: A systematic review of density grid-based clustering for data streams. Ieee Access **10**, 579–596 (2021)
58. Tu, W., Deng, W., Gedeon, T., Zheng, L.: A bag-of-prototypes representation for dataset-level applications. In: CVPR. pp. 2881–2892. IEEE (2023)
59. Vázquez, F.I., Zseby, T.: Temporal silhouette: validation of stream clustering robust to concept drift. Mach. Learn. **113**(4), 2067–2091 (2024)
60. Wang, J., et al.: Everything about transfer learning and domain adapation. `http://transferlearning.xyz`
61. Wang, X., Wang, Z., Wu, Z., Zhang, S., Shi, X., Lu, L.: Data stream clustering: An in-depth empirical study. Proc. ACM Manag. Data **1**(2), 162:1–162:26 (2023)
62. Xie, J., Dai, M., Xia, S., Zhang, J., Wang, G., Gao, X.: An efficient fuzzy stream clustering method based on granular-ball structure. In: ICDE. pp. 901–913. IEEE (2024)
63. Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., Weinberger, K.Q.: An empirical study on evaluation metrics of generative adversarial networks. CoRR **abs/1806.07755** (2018)
64. Zogaj, F., Cambronero, J.P., Rinard, M.C., Cito, J.: Doing more with less: Characterizing dataset downsampling for AutoML. Proc. VLDB Endow. **14**(11), 2059–2072 (2021)
65. Zubaroglu, A., Atalay, V.: Data stream clustering: a review. Artif. Intell. Rev. **54**(2), 1201–1236 (2021)
66. Zubaroglu, A., Atalay, V.: Online embedding and clustering of evolving data streams. Stat. Anal. Data Min. **16**(1), 29–44 (2023)