DeepStream: Autoencoder-Based Stream Temporal Clustering

Shimon Harush, Yair Meidan, Asaf Shabtai

Department of Software and Information Systems Engineering

Ben-Gurion University of the Negev

hshimo@post.bgu.ac.il, yairme@post.bgu.ac.il, shabtaia@bgu.ac.il

ABSTRACT

This paper presents DeepStream, a novel data stream temporal clustering algorithm that dynamically detects sequential and overlapping clusters. DeepStream is tuned to classify contextual information in real time and is capable of coping with a high-dimensional feature space. DeepStream utilizes stacked autoencoders to reduce the dimensionality of unbounded data streams and for cluster representation. This method detects contextual behavior and captures nonlinear relations of the input data, giving it an advantage over existing methods that rely on PCA. We evaluated DeepStream empirically using four sensor and IoT datasets and compared it to five state-of-the-art stream clustering algorithms. Our evaluation shows that DeepStream outperforms all of these algorithms.

CCS CONCEPTS

• Computing methodologies → Machine learning algorithms;

KEYWORDS

Stream clustering; Autoencoder; Dimensionality reduction; Anomaly detection

ACM Reference Format:

Shimon Harush, Yair Meidan, Asaf Shabtai. 2021. DeepStream: Autoencoder-Based Stream Temporal Clustering. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21)*, March 22–26, 2021, Virtual Event, Republic of Korea. ACM, New York, NY, USA, Article 4, 4 pages. <https://doi.org/10.1145/3412841.3442083>

1 INTRODUCTION

Data stream clustering is a special type of clustering that has recently received increased attention from researchers motivated by the challenges posed by applications involving large datasets and streams of data, and the need to learn tasks in real time [1].

Several stream clustering algorithms have been proposed, most of which have two phases: an online phase, which summarizes the data into many micro-clusters, and an offline phase, in which these micro-clusters are re-clustered into a smaller number of final clusters [3]. However, none of these algorithms is suitable for clustering *contextual* sensor streams since these algorithms are designed to produce clusters that are agnostic to the order of arriving records. In contrast, the pcStream algorithm [19] is capable of capturing the temporal importance of the incoming data.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8104-8/21/03.

<https://doi.org/10.1145/3412841.3442083>

We propose a stream clustering method called DeepStream, which is designed to address the above mentioned limitations of existing stream clustering algorithms. DeepStream leverages deep learning techniques to capture the nonlinear relations of input data by using the embedded representation output of a pretrained stacked autoencoder (SAE). By integrating an SAE into the temporal stream clustering algorithm, our method aims to improve the ability to cluster such data streams, thereby improving the quality of the clusters and is able to deal better with high-dimensional data streams. We evaluated our proposed method using four different datasets and compared its performance with that of five existing stream clustering algorithms. The results show a significant improvement in clustering capabilities.

Summary of contributions. **a)** We propose a stream clustering algorithm which is capable of capturing temporal contexts. As a result of integrating SAEs, our method is able to (1) capture nonlinear relations between features, and (2) handle high-dimensional input vectors by using a smaller amount of input features. **b)** We empirically compare our algorithm with state-of-the-art algorithms. Our evaluation shows that our algorithm outperforms the other algorithms in clustering temporal context streams which are present in four datasets.

2 BACKGROUND AND RELATED WORK

2.1 Stream Clustering

Data stream clustering aims to find and maintain a set of valid clusters within a continuous and possibly unbounded stream of records [13]. Handling a huge amount of unbounded data poses a great challenge due to memory and computational constraints [2, 13]. Stream clustering algorithms differ from ordinary clustering algorithms in several aspects: First, since stream clustering algorithms are exposed to the data incrementally, they cannot iterate over all of the data or access different data locations. In addition, most of the data needs to be discarded over time because of memory limitations. Second, the data distribution may change over time and have concept drifts. For this reason, stream clustering models should adapt themselves to new data and have the ability to disregard older, no longer relevant data. Third, incoming data streams accumulate over time, and stream clustering models do not have the time or storage space to handle large amounts of data. Therefore, stream clustering algorithms should only save the minimal amount of data needed to retain their ability to continuously cluster new data efficiently.

In this paper, we empirically evaluate DeepStream and compare it with the following state-of-the-art stream clustering algorithms: (i) DenStream [6], which is a density-based stream clustering algorithm. It uses the clustering feature (CF) form to determine whether a group of micro-clusters is a legitimate cluster or a collection

of outliers; (ii) D-Stream [8], which also performs density-based stream clustering, but across a grid; (iii) DBStream [11], which captures the shared density between two micro-clusters in order to decide whether or not micro-clusters belong to the same cluster; (iv) CluStream [3], which uses a two-phased approach (dividing the clustering process into online and offline macro-clustering) that provides flexibility to explore the nature of the evolution of the clusters over different time periods; and (v) pcStream [19], which summarizes clusters using the mean and principal components (vectors of highest variance) of the cluster’s last records. Unlike the other methods evaluated, pcStream considers the temporal relation between arriving records when making clustering decisions.

Of them, our algorithm works most similarly to pcStream in the way that decisions are made regarding cluster affiliation and the formation of new clusters.

2.2 Autoencoder-Based Clustering

An autoencoder (AE) is a type of artificial neural network used to learn efficient data encodings in an unsupervised manner. Traditionally, AEs were used for dimensionality reduction or feature learning. AEs take structural input data and try to reconstruct the input after encoding it. Some variants focus on the network architecture, such as the number of layers, latent dimension size, or even the use of CAE (convolutional autoencoders) for image data. Other methods are aimed at preventing overfitting and improving robustness by adding noise to the input [22] or by using dropout [21] to generalize the network. For example, DEC [25] uses a joint deep neural network (DNN) and clustering that simultaneously learns feature representations and cluster assignments with an SAE network. Dimensionality reduction using a deep belief network with nonparametric maximum margin clustering was proposed in [7]. In contrast to some of the above mentioned methods which combine the association of clusters with the neural network architecture, our method does not assume a certain number of clusters.

3 PROPOSED METHOD

Our proposed method, called DeepStream, consists of an offline and online phase. In the offline phase a neural network learns the data in the training set in order to generate a model that produces a compact representation of each record. The online phase continuously generates clusters from incoming data streams using the model trained in the offline phase.

3.1 DeepStream’s Offline Phase

One of the key components of DeepStream consists of a stacked autoencoder (SAE), which is a more advanced architecture than a basic autoencoder. Research has shown that SAEs consistently produce semantically meaningful and well-separated representations on real-world datasets [14, 23]. Thus, the unsupervised representation learned by an SAE represents the nonlinear relationship between the input variables and naturally helps DeepStream better separate between different contexts.

An SAE network is built layer by layer, and each layer is a denoising AE which is trained separately to reconstruct the previous layer’s output [23]. Each denoising AE is made up of two layers.

First, the encoder input x passes through *Dropout()*; dropout is a regularization technique which prevents overfitting by dropping out units during the training [21]. h is the hidden representation result of the $g_1(W_1\tilde{x} + b_1)$ activation function, and W and b are neural network parameters. Hidden layer h adds L2 normalization to the latent space during training; in [5], the authors showed that adding L2 normalization to the latent space improves separability of clusters in a variety of deep autoencoder models. The decoder is built the same way as the encoder but in reverse order. In the training phase, the mean squared error ($MSE = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2$) function is minimized. We use the output of h as the input for training the next autoencoder.

Each internal layer uses ReLU as an activation function, because it is simple and computationally inexpensive, and it does not suffer from the vanishing gradient problem like sigmoid does. However, ReLU suffers from a different problem known as the “dying ReLU” problem [17]. In order to lower the chances of “dying ReLU,” we use SELU as the internal layer activation function in some of the datasets when evaluating DeepStream’s performance. After training each layer separately, we build an AE model from all of the hidden layers (h). Next, the SAE undergoes additional training in its entirety to fine-tune the parameters simultaneously. This results in a trained deep AE with the capability of producing a reduced representation of the input; the output of the encoder (i.e., the SAE’s hidden layer) is then used in the online clustering phase.

3.2 DeepStream’s Online Clustering Phase

DeepStream’s online clustering phase is based on its predecessor, the pcStream algorithm [19], except for the way in which the input is manipulated. While pcStream relies on the soft independent modelling by class analogy method (SIMCA) [24] for similarity score calculation, DeepStream uses an SAE to encode high-dimensional input to lower dimensions in order to obtain semantically meaningful context separation by leveraging the nonlinear relationship between the input features.

DeepStream’s clustering phase (Algorithm 1) has six different hyperparameters: threshold (t) represents the level of sensitivity of the algorithm; drift buffer size (d) sets the minimum size of each context; memory size (m) limits the maximum size of each context; (*encoder*) stands for a pretrained encoder that encodes each input from stream X ; and context limit (*lim*) limits the size of the algorithm’s contexts.

In lines 1-2 of Algorithm 1, a list of contexts C is initialized with a new context that contains the first d instances from stream X . The operation is performed by calling the function *updateContext*($X[0..d]$). It is assumed that the first d records belong to the same cluster to provide the algorithm a starting point. In lines 5-16, every record (x) in the data stream is processed. First, the record is encoded to a lower-dimensional representation by the *encode*(x , *encoder*) function.

In line 7, the Mahalanobis distance is measured on $\tilde{x}$ from each context in list C . We chose to use the Mahalanobis distance over the Euclidean distance, since we can’t assume that the clusters have identical co-variances. Clusters with elliptical shape co-variances

Algorithm 1: DeepStream online clustering algorithm.

```

input : threshold ( $t$ ), drift buffer size ( $d$ ), memory size ( $m$ ), trained encoder
        ( $encoder$ ), data stream ( $X$ ), context limit ( $lim$ )
output: Context list ( $C$ )

1  $C \leftarrow []$ 
2  $append(updateContext(X[0..d]), C)$ 
3  $Buffer \leftarrow []$ 
4  $X \leftarrow X[d..]$ 
5 for  $x \in X$  do
6    $\tilde{x} \leftarrow encode(x, encoder)$ 
7    $bestContext \leftarrow \underset{c \in C}{argmin} dist(\tilde{x}, c)$ 
8   if  $dist(\tilde{x}, bestContext) < t$  then
9      $updateContext(C[bestContext], Buffer, m)$ 
10     $Buffer \leftarrow []$ 
11     $updateContext(C[bestContext], \tilde{x}, m)$ 
12  else
13     $append(\tilde{x}, Buffer)$ 
14    if  $len(Buffer) = d$  then
15       $append(updateContext(Buffer), C)$ 
16      if  $len(C) = lim$  then
17         $C \leftarrow merge(C)$ 
18      end
19    end
20  end
21 end

```

---

are modeled much better using the Mahalanobis distance measure. In rare cases when features have zero correlations, the Euclidean and Mahalanobis distance are the same, except that the Euclidean distance performs faster [9]. After measuring all of the distances between $\tilde{x}$ and the cluster centers, the closest cluster index is chosen (line 7). In cases in which the closest cluster distance is less than threshold (t): (1) the attempt to create a new cluster breaks down, and therefore if the drift buffer (denoted as $Buffer$) is not empty, the records are dumped into the $C[bestContext]$ cluster in function $updateContext()$; and (2) the current instance $\tilde{x}$ joins the $C[bestContext]$ cluster. In order to save memory, function $updateContext(C[bestContext], \tilde{x}, m)$ updates the cluster $C[bestContext]$ with the m newest incoming records. In addition, the use of the FIFO policy allows clusters to adapt themselves to changes over time (i.e., concept drift). Then, the inverse co-variance matrix is calculated for the Mahalanobis distance measurement. In cases in which the closest context distance is greater than threshold (t) (lines 13-16), $\tilde{x}$ instance is appended to the drift buffer ($Buffer$), and if the maximum number (d) of instances in the drift buffer has been reached, a new context is created with those instances.

To prevent an explosion of contexts, parameter lim limits the number of possible contexts by merging the two closest contexts in the $merge(C)$ function. Merging contexts is carried out by taking the newest m instances from the two merging contexts. This requires that each instance of DeepStream is saved with its arrival timestamp, in order to enable an accurate merge operation.

4 EXPERIMENTS

4.1 Evaluation Setup

Dataset. In order to evaluate the performance and generality of our proposed algorithm, we tested DeepStream on four real-world contextual sensor datasets, each of which captures a different context domain and constitute extensive coverage of number of input features and different level of contexts. In Table 1, we present a

summary of the datasets that we used to empirically evaluate and compare the stream clustering algorithms.

Benchmarking. In our experiments, we compared DeepStream with five state-of-the-art stream clustering algorithms: pcStream [19], DBstream [11], DenStream [6], D-Stream [8] and CluStream [3]. To enable a fair comparison of the algorithms, each of them was trained on exactly the same training set and was later applied to the same test set. For each algorithm evaluated, we performed a grid search for hyperparameter tuning in order to optimize the algorithm’s performance.

In order to evaluate D-Stream, DenStream, DBstream, and CluStream we used an R package called streamMOA [12]. In streamMOA, the algorithms classify records as noise if they do not fit into any cluster. In order to perform a fair comparison with pcStream and DeepStream, we only took into account results with less than 5% of noisy records. We demonstrate the effectiveness of DeepStream mainly by comparing it with pcStream, as these two algorithms are designed to find temporal contexts in data streams but use different information extraction methods.

Parameter setting. In DeepStream, the parameter tuning process is divided into two parts: (1) the offline training phase in which the AE is trained to encode the input data, and (2) the online phase in which the DeepStream model is trained to cluster the stream data in the training set. For the offline phase, we trained SAEs in order for the encoder to use them later for dimensionality reduction of the input stream. For each of the HAR, HearO, and IoT datasets, we trained an SAE whose architecture (specifically, the size of each layer) was derived from the input size of each dataset. All of the internal layers are activated by the ReLU nonlinearity function [10] except for the IoT dataset which uses SELU (scaled exponential linear unit) [16] as the internal activation function. For the online phase, we used the encoder trained in the previous offline phase and executed DeepStream with grid search on two parameters: threshold (t) and drift buffer size (d). The memory size (m) was set to 1,000 as well as the context limit.

Evaluation metric. In many stream clustering experiments the adjusted Rand index (ARI) and sum of squares (SSQ) are the most common evaluation metrics. However, the SSQ metric assumes that clusters do not overlap, and therefore it is not suitable for our clustering problem. All of the clustering methods were evaluated using the ARI metric [15] which is widely used to compare the quality of different partitions given the ground truth. The ARI is a measure of similarity between two data clustering assignments regardless of their spatial qualities.

4.2 Results

The best results are presented in Table 2, where it is evident that DeepStream outperforms all of the other methods when applied to any dataset. Since we only report the results with less than 5% of noisy records, D-Stream lacks results on the HAR, IoT, and PAMAP2 datasets. All algorithms obtained low ARI scores on the HearO-Low-High and PAMAP2 datasets (since these two datasets consist of real-life data which makes them noisy and hard to cluster).

On the HearO dataset, the results (see Table 2) show that the higher the granularity of the labels, the more sensitive the algorithm needs to be. Accordingly, the values for the selected drift size (d)

Table 1: Description of the datasets used in our study.

Dataset	# of features	Examples of features	# of rows in the test set	Context groups	# of labels	Examples of labels
HAR [4]	561	Accl. (x,y,z), FFT...	349	Human activity recognition	6	Sitting, walking, going upstairs
HearO	5	Accl cor. (xy,xz,yz), temperature, battery level	1257	High-level	4	At home, at work, on a break
				Low-level	9	Hungry, interested, shopping
				Battery plugged in	2	Yes, no
IoT [18]	237	Packet size, bytes, http time..	1024	IoT security camera models	7	Withings, Foscam
PAMAP2 [20]	51	IMU hand, IMU chest, IMU ankle	2442	Human activity monitoring	12	Sitting, standing, rope jumping, descending stairs...

Table 2: ARI values for the datasets evaluated and stream clustering algorithms; the HearO dataset was tested several times with different levels of label granularity.

	DeepStream	pcStream	CluStream	DBstream	DenStream	D-Stream
HAR	0.867	0.622	0.434	0.560	0.460	-
HearO - High-level	0.283	0.250	0.137	0.089	0.089	0.180
HearO - Low-level	0.389	0.138	0.174	0.089	0.089	0.180
HearO - Plugged in	0.510	0.115	0.104	0.039	0.184	0.003
IoT - Security camera	0.726	0.551	0.537	0.497	0.572	-
PAMAP2	0.420	0.362	0.177	0.179	0.215	-

and threshold (t) parameters were lower for the ‘low-level’ labels compared to the ‘high-level’ labels.

Finally, our evaluation supports the empirical evidence reported by Mirsky *et al.* [19], as pcStream outperforms most of the state-of-the-art stream clustering methods evaluated (except DeepStream) on a wide range of temporal context datasets.

5 CONCLUSIONS

In this paper, we presented DeepStream, a stream clustering algorithm that uses deep learning to detect temporal contexts on unbounded temporal data streams. Among its strengths, we note that DeepStream (and particularly the AEs it relies on) does not require any labeled data for training. In addition, DeepStream leverages the capability of AEs to model complex nonlinear functions, compress a high-dimensional feature space into a low-dimensional latent space, and improve stream clustering performance on high-dimensional IoT datasets. Our empirical evaluation showed that DeepStream performs better than state-of-the-art stream clustering algorithms, particularly on high-dimensional sensor and IoT temporal datasets.

Acknowledgments

This research was supported by the CONCORDIA project that has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement number 830927.

REFERENCES

- [1] Charu C Aggarwal. 2007. *Data streams: models and algorithms*. Vol. 31. Springer Science & Business Media.
- [2] Charu C Aggarwal. 2018. A survey of stream clustering algorithms. In *Data Clustering*. Chapman and Hall/CRC, 231–258.
- [3] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 81–92.
- [4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *Esann*.
- [5] Caglar Aytekin, Xingyang Ni, Francesco Cricri, and Emre Aksu. 2018. Clustering and unsupervised anomaly detection with 1 2 normalized deep auto-encoder representations. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.
- [6] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 328–339.
- [7] Gang Chen. 2015. Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084* (2015).
- [8] Yixin Chen and Li Tu. 2007. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.
- [9] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems* 50, 1 (2000), 1–18.
- [10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.
- [11] Michael Hahsler and Matthew Bolaños. 2016. Clustering Data Streams Based on Shared Density Between Micro-Clusters. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (June 2016), 1449–1461. <https://doi.org/10.1109/TKDE.2016.2522412>
- [12] Michael Hahsler, Matthew Bolanos, John Forrest, et al. 2017. Introduction to stream: An extensible framework for data stream clustering research with r. *Journal of Statistical Software* 76, 14 (2017), 1–50.
- [13] Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. 1998. Computing on data streams. *External memory algorithms* 50 (1998), 107–118.
- [14] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.
- [15] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.
- [16] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems*. 971–980.
- [17] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. 3.
- [18] Yair Meidan, Michael Bohadana, Asaf Shabtai, Martin Ochoa, Nils Ole Tippenhauer, Juan Davis Guarnizo, and Yuval Elovici. 2017. Detection of unauthorized IoT devices using machine learning techniques. *arXiv preprint arXiv:1709.04647* (2017).
- [19] Yisroel Mirsky, Bracha Shapira, Lior Rokach, and Yuval Elovici. 2015. pcstream: A stream clustering algorithm for dynamically detecting and managing temporal contexts. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 119–133.
- [20] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*. IEEE, 108–109.
- [21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
- [22] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*. 1096–1103.
- [23] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, Dec (2010), 3371–3408.
- [24] Svante Wold and Michael Sjostrom. 1977. SIMCA: a method for analyzing chemical data in terms of similarity and analogy. *Chemometrics: theory and application* 52 (1977), 243–282.
- [25] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*. 478–487.