Adapting DeepDPM for a streaming data

DeepDPM - Deep Clustering With an Unknown Number of Clusters

Tomer Habusha and Guy Davidi

Tomer Habusha	**tomerhabusha@gmail.com**	**305229353**
Guy Davidi	**guy.davidix@gmail.com**	**205493448**

1 Abstract

In this paper we present an innovative feature in the field of DeepDPM algorithms. A recent article called "DeepDPM - Deep Clustering With an Unknown Number of Clusters" [3] showed an effective deep-clustering method that does not require knowing the value of K (number of clusters before training) as it infers it during the learning. Using a split/merge framework, a dynamic architecture that adapts to the changing K, and a novel loss, the method outperforms existing nonparametric methods. We had two main research directions, one was to improve the subclastring net architecture and the other was to adapt the system to work with streaming data. As we saw there is no significant improvement with the new subclustring architecture, so we focused on the second direction. We were able to adapt the DeepDPM method to a 1-dimensional time series dataset and get the correct number of classes. But still the accuracy was far from the GT values and we found instability in the results' convergence.

Our code is available at https://github.com/guy-davidi/Deep_Learning_project

More Information (Full results, Presentation and logs): https://drive.google.com/drive/folders/1kcLlJhBipZBA-7BVYSsU1thACHeJeci3?usp=sharing

2 Introduction

Cluster analysis is the process of grouping a set of objects in such a way that objects in the same group, or cluster, are more alike to each other than to those in the other clusters. Mainly, in the realistic setting of cluster analysis class labels, the number of classes (defined as K) and their relative size are unknown. This could be a problem in the analysis of large datasets, and methods used today are not effective enough.

Deep clustering is a new research direction for clustering that combines deep learning and clustering. Nonparametric methods (namely, methods that find K) are those that do not make use of the number of clusters as input. Nonparametric deep clustering methods are best for clustering large and high-dimensional-datasets and do so better and more efficiently than classical (i.e., non-deep) clustering methods. There are only a few nonparametric deep clustering methods used today.

The ability to infer the latent K has practical benefits; using the wrong estimate of K can have a significant negative effect on parametric methods. Also, changing K while training has positive optimization-related implications, and using model-selection methods with different Ks limits the size of the database one can use, and thus is not suitable for deep learning methods.

DeepDPM combines the benefits of deep learning and the Dirichlet Process Mixture Model (DPMM). DPMM is a classical Bayesian Nonparametric (BNP) model which is often used in clustering problems where K is unknown. This combination of methods, DeepDPM, offered in [3], is not capable of using a streaming database, as in data that is generated continuously. Clustering streaming data is challenging for several reasons; storing data continuously to an infinite scale, time spent waiting for a stream to end, achieving a reliable estimate on demand, and results must be updated without revisiting previous data. A streaming data clustering algorithm should be fast, does not need to revisit previously processed data, has the ability to modify the number of clusters as needed, supports non-stationary cluster statistics, no label switching and has efficient memory use.

We aim to create a way for DeepDPM to use streaming datasets that could enable users to cluster real time datasets created by streaming data, allowing efficient clustering. Furthermore this allows us to modify the number of clusters as needed during training.

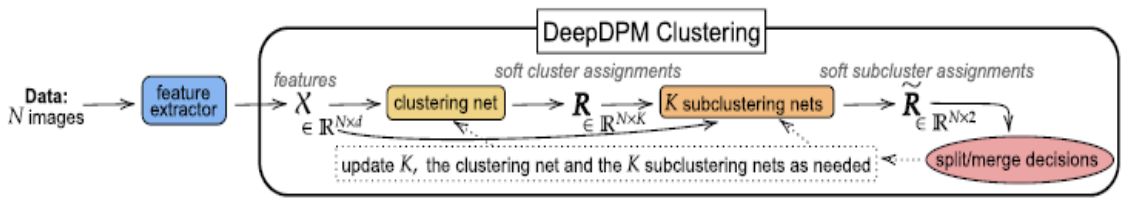
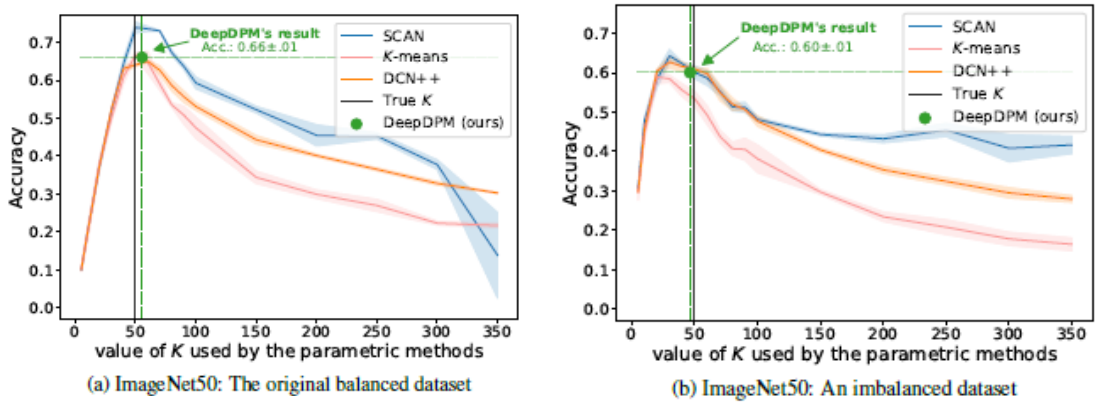


Figure 1. DeepDPM's pipeline: given features X, the clustering net outputs cluster assignments, R, while the subclustering nets generate subcluster assignments, eR. Upon the acceptance of split/merge proposals, all those nets are updated during the learning.

3 Related work

Parametric Deep Clustering methods. One kind of parametric deep clustering consists of a two-step process. Such as in Mc-Conville et al., K-means ran on the embeddings, transformed by UMAP, of a pretrained Autoencoder (AE). This method achieves competitive results when it is applicable, but it is not scalable. SCAN for instance, uses unsupervised pretrained feature extractors, but being a parametric method, it relies on a given relatively accurate estimate of K. Also, SCAN assumes the same weights for the classes, which is often unrealistic.

Nonparametric Classical Clustering. In [3] the authors describe a new method called DeepDPM. This method, also referred to as a DPM interface algorithm, enables the number of clusters to be inferred and changed during training. It consists of three main parts as described in *figure 1* : (1) feature extractor that can be divided into two types. The one is a two-step approach, in which clustering is performed on features extracted in a pretext task. The second approach is an end-to-end method to jointly learn features and clustering, possibly by AE alternations. (2) A clustering net that generates soft cluster assignments for each input data point, and (3) Subclustering nets that take the previously generated soft cluster assignments as inputs and generates soft subcluster assignments, which will later be used to support split and merge decisions to dynamically adapt to and change the number of clusters. This removes the need to predefine the number of clusters in clustering tasks and can infer it instead. In a split step each cluster is divided into two subclusters, this split proposal could be accepted or rejected, and K is increased accordingly. In a merge step, two clusters could merge, if accepted, new subclustering nets are created, and K changed accordingly. DeepDPM was found to be more efficient when compared to classical parametric (e.g. K-means and GMM), classical nonparametric (e.g. DBSCAN and moVB), and deep nonparametric (e.g. DCC and AdapVAE) methods on widely used image and text datasets at varying scales as described in *figure 2* and *table 1*.



(a) ImageNet50: The original balanced dataset

(b) ImageNet50: An imbalanced dataset

Figure 2. Mean clustering accuracy of 3 runs (± std. dev.) on ImageNet50. The Ground Truth K is 50. Parametric methods such asK-means, DCN++ (an improved variant of [71]) and SCAN require knowing K

	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
	MNIST [18]			USPS [35]			Fashion-MNIST [69]		
K-means^p	.90± .02	.84± .05	.85±.06	.86±.01	.79±.05	.80±.06	.67±.01	.50±.03	.60±.04
GMM^p	**.94±.00**	**.95±.00**	**.98±.00**	.86±.02	.79±.05	.81±.06	.66±.01	.49±.02	.58±.03
DBSCAN	.92±0	.86±0	.89±0	.72±0	.46±0	.57±0	.63±0	-.32±0	.39±0
DPM Sampler	.92±.01	.91±.04	.93±.05	.87±.01	.82±.02	.83±.03	.67±.01	.49± .02	.59±.03
moVB	.93±.00	.94±.00	.97±.00	.87±.02	**.86±.04**	**.90±.04**	.66±.02	.47±.03	.55±.03
DeepDPM (Ours)	**.94±.00**	**.95±.00**	**.98±.00**	**.88±.00**	**.86±.01**	.89±.2	**.68±.01**	**.51±.02**	**.62±.03**
	MNIST^{imb}			USPS^{imb}			Fashion-MNIST^{imb}		
K-means^p	.89± .03	.84± .06	.83±.06	.82±.02	.71±.05	.71±.05	.62±.01	.46±.02	.56±.03
GMM^p	.94±.02	.95±.03	.96±.04	.83±.01	.74±.05	.76±.05	.62±.01	.46±.02	.57±.03
DBSCAN	.93±0	.92±0	.94±0	.84±0	.79±0	.80±0	.62±0	.35±0	.46±0
DPM Sampler	.93±.01	.94±.02	.96±.02	.89±.02	.89±.06	.91±.04	**.66±.01**	**.50± .01**	**.61±.01**
moVB	.94±.00	.95±.00	.96±.00	.88±.01	.89±.02	.91±.02	.63±.01	.44±.02	.53±.02
DeepDPM (Ours)	**.95±.01**	**.97±.01**	**.98±.01**	**.90±.00**	**.92±.00**	**.94±.00**	.65±.00	**.50±.00**	**.61±.00**

Table 1. Comparing the mean results (±std. dev.) of DeepDPM with classical clustering methods. The results are the mean of 10 independent runs. Methods marked with p are parametric (require K). Datasets marked with imb are imbalanced ones.

Deep Clustering streaming data. Dinari and Freifeld [6] adapted both the DPMM and a known DPMM sampling-based non-streaming inference method for streaming-data clustering, to create a new BNP method called ScStream. In the papers "Deep learning for time series classification: a review" [4] and "On the Performance of Deep Learning Models for Time Series Classification in Streaming" [5] perform approaches to process the streaming datasets. This with the use of the same databases as we will describe later.

4 Research Directions

As we described in the Introduction section, we focused on two different research directions. First, improving network architecture, especially the sub-clustering network. The second and the main direction was adapting DeepDPM to streaming Data. In the next sections we will describe for each research direction which data we used, the method we developed, the experiments we performed and their results.

5 Improving Network Architecture

As described in the paper, the main step in DeepDpm algorithm is the Sub - clustering network. The sub-clustering network includes input layer with K neurons, an hidden layer with hidden_dim multiple K neurons while detaching different sub-clustering networks, an output layer with 2*K neurons (2 neurons for each current class) and two fully connected layers.

DeepDPM code implementation [9] shows that the sub-clustering network is limited to one hidden layer, and hard-coded value 50 for the

"hidden_dim" parameter. In addition, only one algorithm "2-means" was tried as an alternative method as you can see in *table 2*.

	ACC	
	K_{init}=3	K_{init}=10
No splits/merges	.29±.01	.59±.03
No splits	.29±.01	.59±.02
No merges	.46±.00	.58±.01
2-means instead of f_{sub}	.61±.00	.59±.02
No priors in the M step	.58±.01	.57±.02
Isotropic loss instead of $\mathcal{L}_{\text{cl}}$	.58±.00	.58±.00
DeepDPM (full method)	**.62±.03**	**.61±.00**

Table 2. DeepDPM's performance under different ablations.

In contrast to the other parts of DeepDpm algorithm, we saw that no changes in the sub-clustering hyperparameters were tried. This was the main lead for us for improving sub-clustering network architecture.

5.1 Data

We evaluate our method experiments on the same image datasets used in the original paper: MNIST, Fashion-MNIST. Their properties are described in *table 3.*

Dataset	Train Samples	Val Samples	Data Dimension	GT K
MNIST	60,000	10,000	28X28	10
Fashion-MNIST	60,000	10,000	28X28	10

Table 3. Descriptive properties of the datasets used for evaluation.

5.2 Method

We examined a few changes in sub-clustering architecture: changing number of neurons, changing number of hidden layers, changing activation function from relu to tanh and combination of all.

For supporting more than one hidden layer we add few changes to code implementation: Updating sub-clustering class, updating merge\split functions, and changing the updating net functions. The other parameters we examine are configurable.

5.3 Experiments and Results

As we described, we run DeepDpm script with our additional changes on varied values of the following parameters: dropout, number of hidden layers,

number of neurons in hidden layer and activation function. The results described in *table 4*. As we can see, there is no improvement compared to the paper's [3] result.

Dataset	Dropout	Number of hidden Layers	Number of neurons	Activation function	Accuracy	Final K
MNIST	**0**	**1**	**50**	**relu**	**0.98+0.00 (paper)**	**10**
MNIST	0.5	1	50	relu	0.9787	10
MNIST	0.5	1	75	relu	0.97873	10
MNIST	0	2	50	relu	0.78364	8
MNIST	0	2	25	relu	0.97871	10
MNIST	0.3	2	25	relu	0.9787	10
MNIST	0	2	15	relu	0.9787	10
MNIST	0	2	50	tanh	0.9787	10

Table 4. Comparing the results of DeepDPM with the new changes to the sub-clustering network. The first row is the Paper's result.

6 Adapting DeepDPM to Streaming Data

Our main research was adapting the DeepDPM method to work with streaming datasets. In order to understand more about deep learning classification models for streaming data pipelines, we examine the papers ""On the performance of deep learning models for time series classification in streaming" [5]. The paper evaluates models such as MLP, CNN, LSTM and Temporal Convolutional Network (TCN) over several time-series datasets that are simulated as streams. The framework implemented in the paper is called ADLStream framework as described in the *figure 3*.

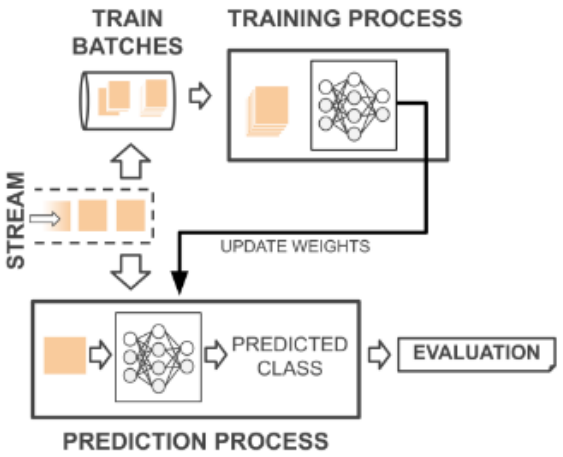


Figure 3. Asynchronous dual-pipeline deep learning framework

We also examine the paper "Deep learning for time series classification: a review" [4]. In our project we used the same datasets from UCRAArchive and evaluated our results compared to their results on the MAP classifier, which is the classifier that is used in the clustering network in DeepDPM method.
In addition, we used preprocessing code implementation [10, 11] for streaming datasets.

6.1 Data

In our project we used datasets from UCRAArchive_2018 [12]
UCRAArchive_2018 includes one-dimensional time series datasets with different characteristics and different domains as described in *table 5.*

#	Dataset	Instances	Length	Classes	Type
1	TwoPatterns	5000	128	4	SIMULATED
2	CinCECGtorso	1420	1639	4	ECG
3	TwoLeadECG	1162	82	2	ECG
4	Wafer	7164	152	2	SENSOR
5	Pendigits	10992	16	10	MOTION
6	FacesUCR	2250	131	14	IMAGE
7	Mallat	2400	1024	8	SIMULATED
8	FaceAll	2250	131	14	IMAGE
9	Symbols	1020	398	6	IMAGE
10	ItalyPowerDemand	1096	24	2	SENSOR
11	ECG5000	5000	140	5	ECG
12	MoteStrain	1272	84	2	SENSOR
13	NonInvasiveFetalECGThorax1	3765	750	42	ECG
14	NonInvasiveFetalECGThorax2	3765	750	42	ECG
15	SwedishLeaf	1125	128	15	IMAGE
16	FordA	4921	500	2	SENSOR
17	Yoga	3300	426	2	IMAGE
18	UWaveGestureLibraryX	4478	315	8	MOTION
19	FordB	4446	500	2	SENSOR
20	ElectricDevices	16637	96	7	DEVICE
21	UWaveGestureLibraryY	4478	315	8	MOTION
22	UWaveGestureLibraryZ	4478	315	8	MOTION
23	HandOutlines	1370	2709	2	IMAGE
24	InsectWingbeatSound	2200	256	11	SENSOR
25	ShapesAll	1200	512	60	IMAGE
26	MedicalImages	1141	99	10	IMAGE
27	PhalangesOutlinesCorrect	2658	80	2	IMAGE
28	ChlorineConcentration	4307	166	3	SIMULATED
29	Phoneme	2110	1024	39	SENSOR

Table 5. CRAArchive_2018's datasets.

Each dataset has the following format: TSV file with one time series exemplar per row, and the first value in the row is the class label. We focused on two different datasets in our experiments:

1. **MedicalImages** [13]: Histograms of pixel intensity of medical images. The classes are different human body regions. This dataset contains a train of 381 points, 760 test points and 10 classes. .
2. **PedestrianCountingSystem** [14]: The City of Melbourne, Australia has developed an automated pedestrian counting system to better understand pedestrian activity within the municipality, such as how people use different city locations at different times of the day. The data analysis can

facility decision making and urban planning for the future. MelbournePedestrian Data are pedestrian counts for 12 months of the year 2017. This dataset contains a train of 1194 points, 2439 test points and 10 classes.

6.2 Method

Adapting DeepDPM to streaming dataset include 3 main parts:
(1) pre-processing to streaming dataset, (2) paradigm replacing from time to feature dimensional and (3) hyperparameters tuning.

The **pre-processing** part includes several additions and changes: converting datasets format to .pt format, deal with "empty values", normalizing data, using some of the built-in functionality for custom data, changing DeepDPM main script to deal with time series files and finally changing functions get_train_data and get_test_data.

In contrast to all experiments in the previous sections, now it was necessary to **move from time dimensional to feature dimensional**.
we move from a two-step approach implemented in DeepDPM.py to an end-to-end approach implemented in DeepDPM_alternations.py.
In addition, we changed DeepDPM_alternations to deal with time series files and adjusted the script to run Auto-Encoder alternations for feature extraction.

The last part was examining **hyperparameters' influence** on our results. We found that the main hyperparameters are the following:

- **init_k:** the initial guess for K;
- **beta:** coefficient of the regularization term on "clustering"**;**
- **lr:** learning rate;
- **train_cluster_net:** Number of epochs to pretrain the cluster net;
- **lambda**:coefficient of the reconstruction loss;
- **number_of_ae_alternations:** When performing feature learning and clustering in alternation, we need to choose the number of times we perform the alternations, one alternation includes training the AE followed by the DeepDPM training;
- **latent_dim:** the AE's learned embeddings dimension.

6.3 Experiments and Results

In the experiments we examine two main results**:** convergence to the dataset's number of classes K (in both cases GT K is 10), and accuracy.

6.3.1 Experiments and Results - MedicalImages dataset

The first experiments were on the MedicalImages dataset [13].
Table 6 summarizes the top 5 results after hyperparameters tuning.
As we can see, the top result (k=7, accuracy=0.26842) is far away from Paper's result (accuracy = 0.722). Our main conclusions from this experiment were

calibrating hyperparameters magnitude, and a dataset with more samples is needed.

Dataset	init_k	beta	lr	train_cluster_net	lambda	number_of_ae_alternation	latent_dim	accuracy	Final K
Medical Images	1	0.001	0.01	350	0.5	2	5	**0.26842**	**7**
Medical Images	1	0.005	0.01	300	0.5	2	5	0.26316	7
Medical Images	1	0.005	0.01	300	0.5	2	5	0.31184	6
Medical Images	1	0.001	0.01	320	0.5	2	5	0.38158	5
Medical Images	1	0.005	0.01	330	0.5	2	5	0.29868	5
Medical Images	Paper's result dor MLP classifier (k is known)							0.722	10

Table 6. Top 5 results MedicalImages results

6.3.2 Experiments and Results - MelbournePedestrian dataset

MelbournePedestrain dataset [14] is bigger than MedicalImages dataset, but with the same number of classes. *Table 7* summarizes the top 5 results after hyperparameters tuning. As we can see, the top result (k=10, accuracy=0.42025) converges to GT K!, the accuracy has increased but still far away from Paper's result (accuracy = 0.8708).

Dataset	init_k	beta	lr	train_cluster_net	lambda	number_of_ae_alternation	latent_dim	accuracy	Final K
Melbourne Pedestrian	1	0.0008	0.008	250	0.5	3	12	**0.42025**	**10**
Melbourne Pedestrian	1	0.0008	0.008	250	0.5	3	24	0.41451	10
Melbourne Pedestrian	1	0.001	0.01	250	0.5	3	12	0.51907	9
Melbourne Pedestrian	1	0.002	0.02	250	0.5	3	12	0.50841	9
Melbourne Pedestrian	1	0.001	0.01	250	0.5	3	12	0.4551	9
Melbourne Pedestrian	Paper's result dor MLP classifier (k is known)							**0.8708**	**10**

Table 7. Top 5 results - MelbournePedestrain dataset

In order to examine the stability of the method, we choose the 3 best configurations and run the script ~6 times on each configuration. The results described in *table 8*. This experiment proved to us that the method is not stable enough.

Dataset	init_k	beta	lr	train_cluster_net	lambda	number_of_ae_alternation	latent_dim	avg acc (over ~6 runnings)	avg K (over ~6 runnings)
MelbournePedestrian	1	0.0008	0.008	250	0.5	3	24	0.455	8
MelbournePedestrian	1	0.001	0.01	250	0.5	3	12	0.456	7.83
MelbournePedestrian	1	0.0008	0.008	250	0.5	3	12	0.455	7.71
MelbournePedestrian	Paper's result dor MLP classifier (k is known)							0.8708	10

Table 8. Average results - MelbournePedestrain dataset

7 Conclusion

During this research we deal with a relatively large integration between several methods and ways to implement the existing algorithm and adapt it to time-dependent information. In this way, we learned about a number of innovative and interesting methods that we did not know before, especially at a time when big data is growing. Together with these methods we got to know and learn about innovative architectures from the last few years that we were not exposed to before.

Our main conclusions from our research are: we were able to adapt the DeepDPM method to a 1-dimensional time series dataset and get the correct number of classes. But still the accuracy was far from the paper's results and we found instability in the results' convergence.

A possible option for innovation is mainly adjusting additional factors and performing Mathematical changes on DeepDPM algorithm for time series datasets. In addition, working with multidimensional time series datasets

8 Appendix

8.1 Our Code and results

[1] Our code is available at
https://github.com/guy-davidi/Deep_Learning_project

[2] More Information (Full results, Presentation and logs):
https://docs.google.com/spreadsheets/d/1YrwLQl-BepSNlq82RTl1ukWeZpodT44b09bcqFuVaDY/edit?usp=sharing

8.2 Reference Papers

[3] Meitar Ronen, Shahaf Finder, and Oren Freifeld. "DeepDPM: Deep Clustering With an Unknown Number of Clusters", CVPR 2022

[4] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller. “Deep learning for time series classification: a review”

[5] Lara-Benítez, Pedro & Carranza-García, Manuel & Martínez-Álvarez, Francisco & Riquelme, José. (2021). On the Performance of Deep Learning Models for Time Series Classification in Streaming.

[6] *Or Dinari, Oren Freifeld.* “Sampling in Dirichlet Process Mixture Models for Clustering Streaming Data”,

[7] Huang, X., Hu, Z. & Lin, L. Deep clustering based on embedded auto-encoder. *Soft Comput* (2021).

[8] C. Avgerinos, V. Solachidis, N. Vretos and P. Daras, "Non-Parametric Clustering Using Deep Neural Networks," in *IEEE Access*,

8.3 Reference code

[9] https://github.com/BGU-CS-VIL/DeepDPM - Paper [1]

[10] https://github.com/hfawaz/dl-4-tsc - Paper [2]

[11] https://github.com/pedrolarben/ADLStream - Paper [3]

8.4 Additional Research data

[12] https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ - UCR Time Series Classification Archive

[13] http://www.timeseriesclassification.com/description.php?Dataset=MedicalImages - MedicalImages dataset

[14]http://www.timeseriesclassification.com/description.php?Dataset=MelbournePedestrian - MelbornePedestrian dataset