PAPER

Data Augmented Incremental Learning (DAIL) for Unsupervised Data

Sathya MADHUSUDHANAN[†a)] ***and*** **Suresh JAGANATHAN**[†b)]**,** ***Members***

SUMMARY Incremental Learning, a machine learning methodology, trains the continuously arriving input data and extends the model's knowledge. When it comes to unlabeled data streams, incremental learning task becomes more challenging. Our newly proposed incremental learning methodology, Data Augmented Incremental Learning (*DAIL*), learns the ever-increasing real-time streams with reduced memory resources and time. Initially, the unlabeled batches of data streams are clustered using the proposed clustering algorithm, Clustering based on Autoencoder and Gaussian Model (*CLAG*). Later, *DAIL* creates an updated incremental model for the labelled clusters using data augmentation. *DAIL* avoids the retraining of old samples and retains only the most recently updated incremental model holding all old class information. The use of data augmentation in *DAIL* combines the similar clusters generated with different data batches. A series of experiments verified the significant performance of *CLAG* and *DAIL*, producing scalable and efficient incremental model.

key words: *incremental learning, unsupervised data, clustering, data augmentation, extreme learning machine*

1. Introduction

The process of annotating massive data generated in real-time applications is expensive. In these cases, finding the number of clusters becomes a tedious task and such data does not hold any labels, termed as unsupervised or undirected data. Based on the relationship between the data, unsupervised learning technique uncovers the hidden patterns among them. Based on the relationship between the data points, the technique of unsupervised learning uncovers the hidden patterns among them. There are two categories of unsupervised learning – parametric and non-parametric. The Parametric method learns data that follows a probability distribution based on specific parameters. Non-parametric methods does not hold any assumptions on the data distribution.

Clustering [1], [2], an unsupervised learning technique, helps gain insight into the structure of data by organizing objects together into groups based on their similarity. A good cluster is a group of data with minimal internal distance (inter-cluster) and maximal external distance (intra-cluster). Incremental learning or online learning [6]–[8], a machine learning technique, which in contrast to transfer learning [3]–[5], [32], learns the continuously arriving input data and extends the current model information without retraining old samples. It doesn't reuse the pre-trained early layers as in transfer learning. Incremental learning (IL) aims to serve stability-plasticity dilemma [9], by gradually updating the model based on newly arrived data and preserve the knowledge gained from training old data with limited and efficient memory resources. IL serves to address the issue of catastrophic forgetting [9] – the tendency to forget previously learned knowledge.

Incremental learning must also handle the constantly arriving new and unpredictable data changes by creating models with adaptive complexity. Adopting Data augmentation (DA) [10], [11] can help to handle this kind of changes. DA is an oversampling data analysis technique used when there are lesser samples of data. It increases the data samples by diversifying the existing data, either by adding slightly modified copies or synthesizing new data from existing data. DA helps to reduce the problem of overfitting while training a model and acts as a regularizer. In many image classification applications, the data augmentation technique has aided in transferring the knowledge from one task to another related task. The application of random transformations like flips/rotations helps in augmenting the images.

Our contributions are of two-fold: i) design of new clustering algorithm based on Autoencoder and Gaussian model *(CLAG)*, which clusters every batch of unsupervised data, and ii) design and implementation of Data Augmented Incremental Learning *(DAIL)*, which builds an incremental model using data augmentation.

CLAG clusters every incoming batch, and *DAIL* generates a model for each batch. Few samples are generated from each batch using data augmentation. *CLAG* groups the newly generated samples and the subsequent data batch, and *DAIL* creates an updated incremental model, retaining all previously learned information.

Rest of the paper is organized as, Sect. 2 gives an overview of all existing clustering methodologies and lists the application of incremental learning in clustering unsupervised real-time stream data. Section 3 elucidates the proposed clustering algorithm *CLAG* and details the functionality of *DAIL*. Section 4 gives a detailed analysis of the experiments carried out and tabulates the results. Section 5 briefs the conclusion of the paper with future work.

Manuscript received October 8, 2021.
Manuscript revised January 14, 2022.
Manuscript publicized March 14, 2022.

[†]The authors are with Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, Tamilnadu, India.
a) E-mail: sathyam@ssn.edu.in
b) E-mail: sureshj@ssn.edu.in

DOI: 10.1587/transinf.2021EDP7213

2. Literature Survey

Just a hasty glimpse at few articles that seems to be noteworthy from the literature archive on clustering, transfer learning, data augmentation and incremental learning is dealt in this section.

There are several clustering algorithms which some falls in traditional and some in modern. Dongkuan Xu and Yingjie Tain [17] discussed the time complexity, merits and demerits of various clustering algorithms and also few evaluation metrics used for checking the methods. In contrast, Saima Bano and M.N.A. Khan [18] presents a detailed survey of different partitioning and hierarchical clustering algorithms with their pros and cons. It details the *Artificial Bee Colony* and *Firefly algorithms* to optimize the clustering performance, scalability and dimensionality. Preliminaries of deep learning architecture and taxonomy of deep learning clustering techniques, their characteristics are learnt from [19]. [20] proposed *ClusterNet*, which clusters a considerable amount of unlabeled data with few labelled samples. *ClusterNet*, a convolutional autoencoder-based semi-supervised clustering, can learn the data representations to cluster the semi-supervised data.

Chunfeng Song et al. [21] proposed an *autoencoder based clustering* technique, and they have presented a deep autoencoder architecture extracts image features using a robust non-linear mapping, followed by a traditional k-means [29], [30] algorithm for clustering the images. Xifeng Guo et al. [22] proposed the architecture of *Deep Convolutional Embedded Clustering*, where the convolutional autoencoder initially extracts the low dimensional data features minimizing the clustering loss. The k-means algorithm uses the extracted features to cluster the images. For categorical unstructured data clustering, D. Venkatavara Prasad et al. [23] proposed an algorithm named *uCLUST* and it clusters the data on an attribute basis using linked lists.

With the knowledge gained on a few existing clustering algorithms, we also learnt few incremental clustering techniques using the following works. Steven Young et al. [24] proposed a fast and stable incremental algorithm for clustering online data, where samples arrive in iterations. It employs *competitive learning algorithms*, which continuously update the centroids based on input data streams' arrival. Junpeng Bao et al. [25] proposed a boundary profile-based incremental clustering *(BPIC)* algorithm for clustering data streams. It arbitrarily finds clusters' shape by defining boundary profiles and updating them according to the dynamically growing dataset. The boundary-vector-based boundary point detection *(BV-BPD)* algorithm determines the boundary profiles of the clusters. Margareta Ackerman and Sanjoy Dasgupta [26] have studied various incremental clustering methods and initiated a formal analysis on the types of cluster structure the algorithms could detect. The use of weaker boundaries allows additional clusters, which overcomes the limitations of incremental clustering.

Mitchell D. Woodbright et al. [27] have proposed an incremental clustering algorithm named *UIClust*, which clusters each batch of data using traditional k-means algorithm and updates the clustering result when new data arrives using the *DistClust* algorithm. Sanjay Chakraborty et al. [28] have compared the performance of two popular clustering techniques – *incremental K-means* and *incremental DBSCAN*. Experimental results show that *incremental K-means* outperforms in terms of time complexity, while *incremental DBSCAN* excels in handling noisy data. Table 1 tabulates the summary and compares above said works with our proposed method *DAIL*.

Table 1 Summary of the related works

S.No	Paper Title	Clustering Algorithm used	Supports Incremental Learning	Data Type supported / Learning Method
1.	*Semi-Supervised Clustering with Neural Networks*	Convolutional autoencoder + constrained K-Means	X	Image / Semi-supervised
2.	*Auto-encoder Based Data Clustering*	Deep Autoencoder + K-Means	X	Image / Unsupervised
3.	*Deep Clustering with Convolutional Autoencoders*	Convolutional autoencoder + K-Means	X	Image / Unsupervised
4.	*uCLUST - a new algorithm for clustering unstructured data*	Linked lists clustering	X	Categorical / Unsupervised
5.	*A Fast and Stable Incremental Clustering Algorithm*	Competitive learning clustering algorithms	✓	Numerical / Unsupervised
6.	*An incremental clustering method based on the boundary profile*	BPIC + BV-BPD(DBSCAN)	✓	Numerical / Unsupervised
7.	*A Novel Incremental Clustering Technique with Concept Drift Detection*	UIClust – K-Means + DistClust	✓	Numerical / Unsupervised
8.	*Performance comparison of incremental k-means and incremental DBSCAN algorithms*	Incremental K-Means and Incremental DBSCAN	✓	Numerical / Unsupervised
9.	***Proposed Work***	**CLAG + DAIL**	✓	**Image, Numerical / Unsupervised**

3. Methodology

Our proposed work adopts data augmentation to implement incremental learning [33], which handles unsupervised data with varying classes at varying times. Incremental learning technique creates a model for each batch separately gathering the new knowledge, and in addition, it remembers the old information. The newly proposed clustering algorithm, *Clustering using the Auto-Gaussian Mixture Model (CLAG)*, clusters every batch of unsupervised data. *CLAG* embeds GMM [14] clustering into the Stacked Autoencoder (SAE), where SAE [12], [13] reduces the dimensionality of input features and clusters them simultaneously using GMM. Few samples from each cluster are collected, and augmented to generate new samples. Newly generated samples are combined with the next batch and fed as input for clustering. After clustering, each cluster is assigned a label and fed into *DAIL*, which uses the Extreme Learning Machine [15], [16] classifier to create a model. The use of data augmentation on each batch serves two purposes:

- It combines the similar clusters produced with the current and next batch of data. Refer *Sect. 5.3* for an illustration of data augmentation.
- Helps in creating incremental classifier model, serving the stability-plasticity dilemma and overcoming the problem of catastrophic forgetting.

3.1 DAIL: Data Augumented Incremental Learning

Figure 1 shows the architecture of *DAIL* supporting the

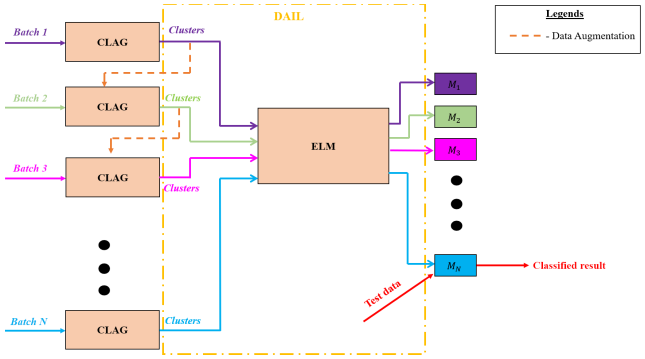


Fig. 1 DAIL architecture

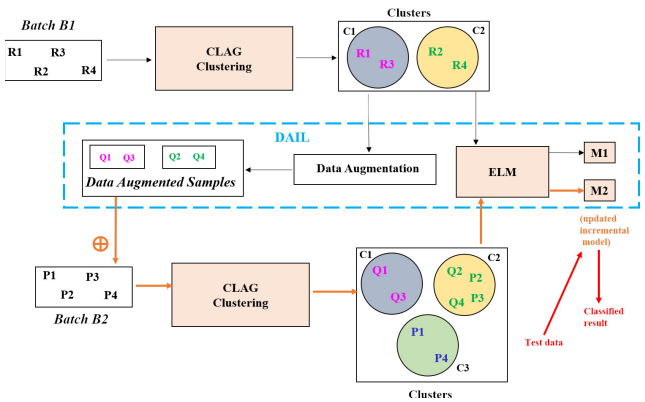


Fig. 2 Illustration of DAIL

incremental learning of unsupervised real-time stream data, which arrive in batches at regular time intervals.

The newly proposed *CLAG* algorithm clusters every data batch separately, which embeds GMM clustering into Stacked Auto Encoder. After labelling the data according to the clusters assigned, the well-known classifier Extreme Learning Machine (ELM) trains the input and creates a model. On the other hand, *DAIL* helps to augment the clustered data to create new samples, which serves as an additional input for clustering the next batch of data. The data augmentation process, i) carries forward all the learned information about existing cluster labels through new samples and ii) combines the similar clusters on current and next batch of data, thus avoiding unnecessary clusters. After data augmentation, the old samples can be deleted and need not be stored, saving the memory space. The next batch of data is then clustered only with new samples, avoiding any re-clustering of old samples, thus saving the processing time.

Figure 2 depicts the purpose of data augmentation, let us consider, there are two batches $B1$ and $B2$, records $R1$, $R2$, $R3$, $R4$, are available in batch $B1$, similarly records $P1$, $P2$, $P3$, $P4$, are available in batch $B2$. Initially, *CLAG* forms two clusters $C1$ and $C2$, where records $R1, R3 \in C1$ and $R2, R4 \in C2$ and model $M1$ is generated. Samples are augmented using the data available in two clusters $C1$ and $C2$, let us say records $Q1$, $Q2$, $Q3$, $Q4$ are augmented data, when clustered with batch $B2$, *CLAG* forms three clusters ($C1$, $C2$ and $C3$) using the above said records, and $Q1, Q3 \in C1$, $Q2, Q4, P2, P4 \in C2$ and $P1, P4 \in C3$. While augmenting records $Q1$, $Q3$ are generated from $R1$, $R3$ and $Q2$, $Q4$ are generated from $R2$, $R4$, which falls in two clusters $C1$ and $C2$, retaining the previously learned information about the clusters (detailed explanation is available in Sect. 3.3).

Later, *DAIL* trains and classifies the newly labelled batch and augmented data using ELM, creating a new updated model and test data from old and new batches are given to the updated model to get the classified result. Figure 2 shows that model $M1$ has learned old labels/classes, and model $M2$ has learned both old and new labels/classes. It also clearly indicates that *DAIL* discards the old model $M1$ and the test data is classified only using the new model $M2$.

3.2 CLAG: Clustering Using Auto-GMM

The proposed *CLAG* clustering algorithm embeds GMM in a stacked autoencoder. SAE is a multi-layered neural network consisting of several sparse autoencoders. Each autoencoder consists of i) an encoder, which compresses the input features to produce a code, and ii) a decoder, which reconstructs the input back from the generated code. An encoder in SAE non-linearly maps any hidden layer h_i from its previous hidden/input layer x_i, as in Eq. (1).

$$h_i = f(l_i) = \sigma(W.x_i + b) \tag{1}$$

Any hidden/input layer $x_i^{'}$ is reconstructed from its previous hidden layer using a decoder as in Eq. (2).

$$x_i^{'} = f(h_i) = \sigma^{'}(W^{'}.h_i + b) \tag{2}$$

Equation (3) calculates the reconstruction loss as follows:

$$L(x, x^{'}) = min(\frac{1}{M}\sum_{i=1}^{M}||x_i - x_i^{'}||^2) \tag{3}$$

where M represents the number of input data samples.

In contrast to K-means clustering, GMM performs the clustering via density estimation, which uses Gaussian distribution to model each cluster. GMM addresses the clusters which are non-circular in shape and does a soft clustering by assigning the data to a cluster with some probability. The probability distribution function describing each multivariate Gaussian cluster is given by Eq. (4).

$$N(x;\mu,\lambda) = \frac{1}{(2\pi)^{\frac{d}{2}}}|\lambda|^{-\frac{1}{2}}\exp\{-\frac{1}{2}(x-\mu)^T\lambda^{-1}(x-\mu)\} \tag{4}$$

where μ represents the mean vector, λ represents the $d \times d$ co-variance matrix, d represents the dimensions, and π represents the cluster's weights.

The general evaluation of the parameters μ, λ is done as in Eqs. (5) and (6).

$$\mu = \frac{1}{m}\sum_{i} x^{(i)} \tag{5}$$

Algorithm 1: Clustering using CLAG

Require: Input Samples
Initialize each cluster with a random μ_c, π_c and λ_c.
Let $t = 1$
repeat
Run the t^{th} iteration of the stacked autoencoder to compute $f^t(x_i)$.
E-Step:
For the extracted feature representation $f^t(x_i)$
Compute

$$r_{ic} = \frac{\pi_c N(f^t(x_i); \mu_c, \lambda_c)}{\sum_{c'} \pi_{c'} N(f^t(x_i); \mu_{c'}, \lambda_{c'})} \tag{7}$$

Assign feature representation $f^t(x_i)$ to the cluster c whose r_{ic} value is higher.
M-Step:
Update the parameters, m_c, pi_c, μ_c and λ_c
Compute

$$\mu_c = \sum r_{ic} \tag{8}$$

Compute

$$\pi_c = \frac{m_c}{m} \tag{9}$$

Compute

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} f^t(x^{(i)}) \tag{10}$$

Compute

$$\lambda_c = \frac{1}{m_c} \sum_i r_{ic} (f^t(x^{(i)}) - \mu_c)^T (f^t(x^{(i)}) - \mu_c) \tag{11}$$

Compute log-likelihood

$$log_t p(f(X)) = \sum_i log[\sum_i \pi_c N(f^t(x_i); \mu_c, \lambda_c)] \tag{12}$$

Compute objective function as $A - B$, where

$$A = min[\frac{1}{M} \sum_{i=1}^{M} \|f^t(x_i) - f^t(x_i')\|^2] \tag{13}$$

$$B = min[log_t p(f^t(X)) - log_{t-1} p(f^{t-1}(X))] \tag{14}$$

Compute $t = t + 1$
until objective function is minimized and converged
NOTE: The minimization of the objective function ensures both the minimal loss in reconstructing the input and the assignment of samples to the optimal cluster.

and

$$\lambda = \frac{1}{m} \sum_i (x^{(i)} - \hat{\mu})^T (x^{(i)} - \hat{\mu}) \tag{6}$$

where m represents the number of samples in a cluster.

CLAG produces an efficient non-linear mapping of the input features to a compact code representation, which undergoes the GMM clustering steps to create clusters. GMM clustering algorithm repeats the steps Expectation-Maximum (EM) Likelihood until the convergence reaches. In EM algorithm, E-step computes the probability of a sample belonging to a cluster and the M-step updates the model parameters. Algorithm 1 details the steps involved in the *CLAG*.

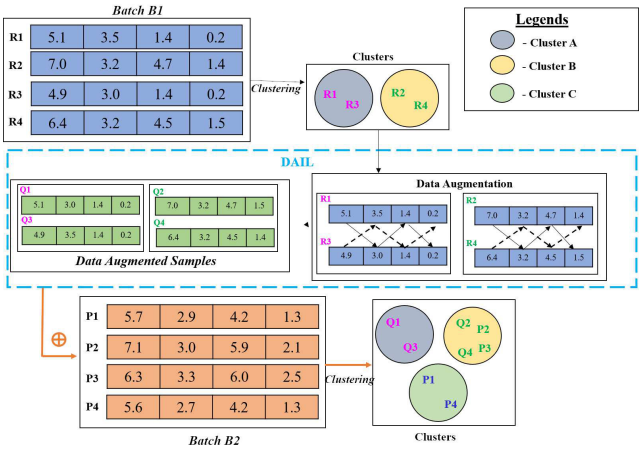


Fig. 3 Data augmentation: an illustration

3.3 Data Augmentation

Data Augmentation is the technique for generating a new samples used in *DAIL* to support incremental learning. After clustering of each batch of data, the ELM classifier creates a model for the same. On the other hand, *DAIL* augments the clustered data to combine it with the next batch of unlabeled data. *DAIL* carries out the data augmentation process by alternating the column values between any two input samples. This procedure ensures that the generated samples belong to the same cluster as the selected inputs. The data augmentation serves two purposes in *DAIL*.

1. Helps in combining similar clusters generated with two different batches of data.
   From Fig. 3, *Batch B1* produces only two clusters – *Cluster A* and *Cluster B*. Data Augmented samples *Q*2 and *Q*4, are generated from *R*2 and *R*4 of *Batch B1*, falls into the same cluster as samples *P*2 and *P*3 of *Batch B2*. Thus, it combines the similar *Cluster B* generated with two different sets of data.
2. Supports incremental learning – any previously learned cluster information is preserved and carry forward to the next batch of data. For instance, if the sample data from new batches doesn't falls in the available clusters, then data augmentation helps to retain the missing cluster information.
   From *Batch B1* two clusters are formed, i.e. *Cluster A* and *Cluster B*, from *Batch B2* also produces two clusters *Cluster B* and *Cluster C*, where *Cluster A* is completely unknown for the *Batch B2*. With the help of data augmentation done using *Batch B1*, and by combining the data with *Batch B2*, *Cluster A* can be produced (for pictorial representation refer Fig. 3.

Figure 3 illustrates the purpose of Data Augmentation in *DAIL*. Table 2 summarizes the clustered records, and algorithm 2 details how the data augmentation process generates new samples.

Table 2 Summary of clustered records

Sample ID	Batch ID	Data Augmented Sample?	Cluster label	Cluster by Data Augmentation?
R1	1	X	A	X
R2	1	X	B	X
R3	1	X	A	X
R4	1	X	B	X
P1	2	X	C	X
P2	2	X	B	X
P3	2	X	B	X
P4	2	X	C	X
Q1	2	✓	A	✓
Q2	2	✓	B	X
Q3	2	✓	A	✓
Q4	2	✓	B	X

Algorithm 2: Data Augmentation

Input: samples S_i $(m - dimensional)$ from current batch B_p along with their cluster label C_k

Output: new samples Q_i serving as additional input for the next batch B_{p+1}

1. For each cluster $k = 1, 2, \ldots, K$ created in batch B_p,
   - Take even number of samples $(S_1, S_2, \ldots, S_N)$ from each cluster C_k
   - For every two successive records, do the following:
   for $(i = 1, 3, \ldots, N - 1)$ **do**
   if *(m is odd)* **then**
   $Q_i = concat(S_{(i,j)}, S_{(i+1,j+1)}, S_{(i,j+2)}, S_{(i+1,j+3)}, \ldots, S_{(i,m)})$
   $Q_{i+1} = concat(S_{(i+1,j)}, S_{(i,j+1)}, S_{(i+1,j+2)}, S_{(i,j+3)}, \ldots, S_{(i+1,m)})$
   else
   $Q_i = concat(S_{(i,j)}, S_{(i+1,j+1)}, S_{(i,j+2)}, S_{(i+1,j+3)}, \ldots, S_{(i+1,m)})$
   $Q_{i+1} = concat(S_{(i+1,j)}, S_{(i,j+1)}, S_{(i+1,j+2)}, S_{(i,j+3)}, \ldots, S_{(i,m)})$
   end
   end
2. Add the newly generated samples along with the next batch of data B_{p+1}.
3. Do the clustering for batch B_{p+1}.

3.4 Incremental Learning

Where does the incremental learning happen in *DAIL*? Each batch of clustered data undergoes data augmentation to i) preserve previously learned cluster information and ii) combine similar clusters generated on different data batches. Following the clustering process, ELM trains and creates a model representing the generated clusters. Every clustering result will hold information about all the previous clusters with the help of data augmentation. ELM creates a new incremental model containing information about all the formerly generated clusters by training the recently produced clusters. Every newly generated model discards all the previously learned data samples and created models. The most recently updated model classifies the test data. The advantages of discarding all the old samples and models are, i) saves memory space – avoids unnecessary storage of old samples and models, ii) avoids retraining of old samples and saves training time.

Table 3 Standard datasets

S.No	Dataset Type	Datasets	Number of samples	Number of classes	Number of attributes
1.	Numerical	Letter Recognition	20000	20	16
2.		Cardiotocography	2126	10	23
3.		Broad institute cancer	14000	14	5
4.		GHz outdoor channel measurement	7840	3	5
5.		AReM tasks	42240	6	6
6.		PIMA diabetes	768	8	2
7.		Buddy Move	249	3	6
8.	Image	MNIST	60000	10	784
9.		USPS	4649	10	256
10.		YaleB	5850	10	1200
11.		CIFAR100	60000	100	1024

Table 4 Accuracy and NMI comparison between clustering methods

S.No	Datatype	Dataset	Performance Comparison					
			Accuracy			NMI		
			SAE-K-Means	GMM	CLAG	SAE-K-Means	GMM	CLAG
1.	Numerical	Letter Recognition	0.71	0.73	0.78	0.65	0.70	0.77
2.		Cardiotocography	0.845	0.872	0.89	0.889	0.89	0.92
3.		Broad institute cancer	0.832	0.853	0.885	0.856	0.883	0.90
4.		GHz outdoor channel measurement	0.872	0.877	0.90	0.911	0.915	0.94
5.		AReM tasks	0.84	0.868	0.876	0.88	0.897	0.912
6.		PIMA diabetes	0.874	0.916	0.94	0.90	0.939	0.963
7.		Buddy Move	0.576	0.60	0.640	0.534	0.585	0.620
8.	Image	MNIST	0.760	0.783	0.815	0.669	0.688	0.703
9.		USPS	0.715	0.72	0.735	0.651	0.659	0.66
10.		YaleB	0.902	0.916	0.93	0.923	0.934	0.956

4. Experimental Results and Analysis

4.1 Experimental Setup

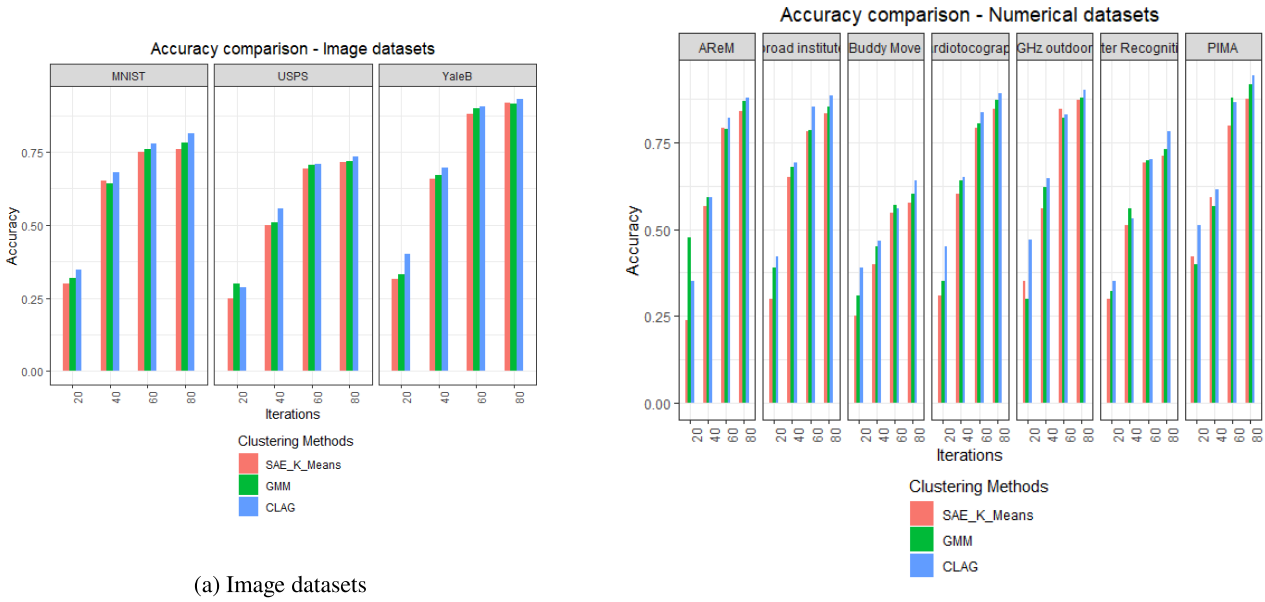
The proposed work was implemented in R programming and tested using standard UCI datasets. Every standard dataset is split into two batches to test the incremental learning scenario. The input features were dimensionally reduced and clustered using the proposed *CLAG* clustering algorithm. Experiments compare the performance of *CLAG* with the existing clustering algorithm, *SAE-K-Means*. Whenever a new batch of data arrives, *DAIL* augments the most previous data batch. In *DAIL*, ELM incrementally trains every data cluster and creates a separate model for every data batch. The most recent model tests and classifies the new unlabelled data, which holds information about all the previously learned clusters. Experiments compare *DAIL's* performance with the existing *ELM++* algorithm.

4.2 Datasets

Experiments are carried out using the standard datasets considering, i) numerical and ii) image. Table 3 shows the details about the dataset. Experiments are carried out in three parts. First, we compared the *CLAG* against *SAE-K-Means* and *GMM*. Secondly, we compared the *DAIL* against *ELM++*, and then, we checked the efficiency of *DAIL* over a few incremental learning methods like *GR* and *BI-R*.

4.3 Clustering Results

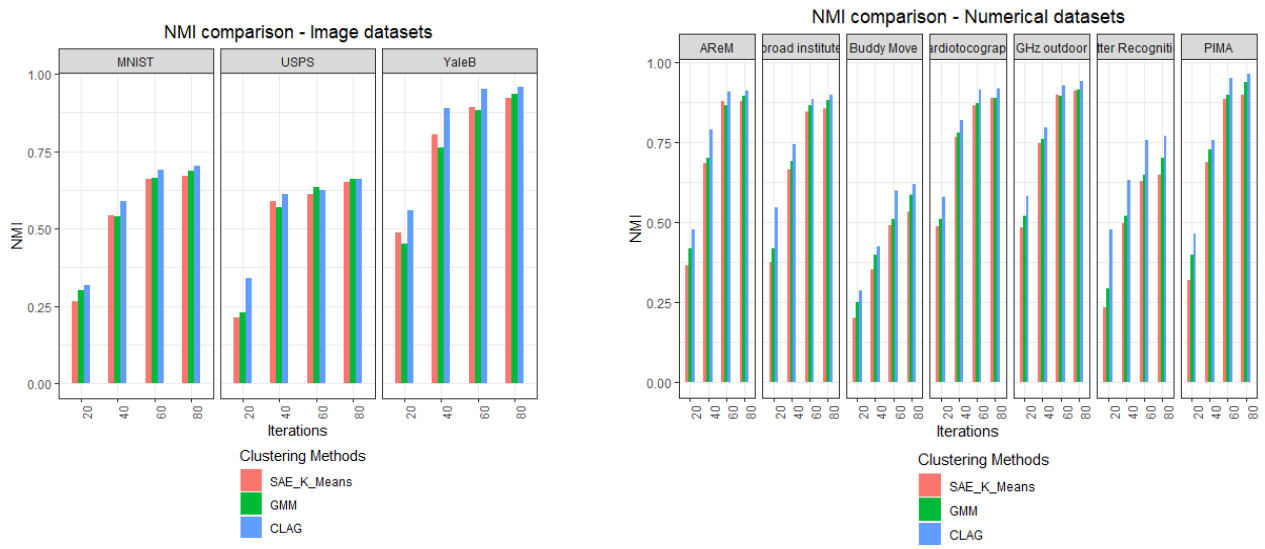
CLAG applies dimensionality reduction technique to reduce the size of dataset and two evaluation metrics are considered, i) Accuracy and ii) Normalized Mutual Information



(a) Image datasets

(b) Numerical datasets

Fig. 4 Accuracy comparison between clustering methods: (a) image datasets, (b) numeric datasets



(a) Image datasets

(b) Numerical datasets

Fig. 5 NMI comparison between clustering methods: (a) image datasets, (b) numeric datasets

(NMI). *CLAG* is compared with existing algorithm *SAE-KMeans* [21] which adopts autoencoder for dimensionality reduction as *CLAG* does.

(i) Accuracy: Accuracy is the fraction of samples correctly classified by any model. Equation (15) calculates the accuracy of a model.

$$accuracy = \frac{number\ of\ correctly\ classified\ samples}{Total\ number\ of\ samples} \tag{15}$$

(ii) Normalized Mutual Information: Normalized Mutual Information (NMI) is the normalized score of the mutual information. Mutual information gives the agreement between two splits – splits according to clusters and splits according to class labels. The score lies between 0 and 1.

Let Q be the actual cluster labels, and R be the labels obtained from clustering. Equation (16) computes the normalized mutual information.

$$NMI = MI(Q,R)/max(H(Q),H(R)) \tag{16}$$

Where $H(Q)$, $H(R)$ are the entropies of Q, R, and

$MI(Q,R)$ is the mutual information of Q and R, which are calculated as in Eqs. (17) and (18) respectively.

$$H(Q) = -\sum_{i=1}^{Q} P_i log(P_i) \tag{17}$$

Where P_i denotes the probability of a cluster label.

$$MI(Q,R) = H(Q) - H(Q|R) \tag{18}$$

Where $H(Q|R)$ is the conditional entropy.

Table 4 tabulates the *accuracy* and *NMI* comparison between *CLAG* and existing *Gaussian Mixture Models (GMM)*, *SAE-K-Means* clustering methods. The evaluation metrics of the clustering results show that *CLAG* outperforms both *GMM* and *SAE-K-Means*. The *CLAG* supports dimensionality reduction and achieves better performance in fewer iterations, as in Figs. 4 and 5, which shows the *accuracy* and *NMI* comparison between the three methods. Figures 4 (a) and 5 (a) shows the results for *image* datasets, whereas Figures 4 (b) and 5 (b) illustrates *numerical* datasets.

Table 5 Performance comparison between DAIL and ELM++ on numerical and image datasets

S.No	Datatype	Dataset	Batch	Number of clusters in a batch	Overlapping cluster cases involved?	Accuracy		
						Indi. Testing	Incre. Testing	
							ELM++	DAIL
1.	Numerical	Letter Recognition	1	10(1-10)	✓	0.678	0.72	0.756
			2	12(9-20)		0.708		
2.		Cardiotocography	1	5(1-5)	X	0.804	0.842	0.88
			2	5(6-10)		0.856		
3.		Broad institute cancer	1	7(1-7)	X	0.796	0.85	0.871
			2	7(8-14)		0.848		
4.		GHz outdoor channel measurement	1	2(1-2)	✓	0.86	0.89	0.92
			2	2(2-3)		0.91		
5.		AReM tasks	1	3(1-3)	✓	0.82	0.87	0.895
			2	4(3-6)		0.86		
6.		PIMA diabetes	1	2(1-2)	✓	0.916	0.945	0.97
			2	2(1-2)		0.956		
7.		Buddy Move	1	2(1-2)	✓	0.577	0.624	0.66
			2	3(1-3)		0.64		
8.	Image	MNIST	1	5(1-5)	✓	0.79	0.823	0.854
			2	7(4-10)		0.80		
9.		USPS	1	5(1-5)	✓	0.715	0.755	0.78
			2	5(6-10)		0.748		
10.		YaleB	1	5(1-5)	✓	0.86	0.925	0.93
			2	7(4-10)		0.94		

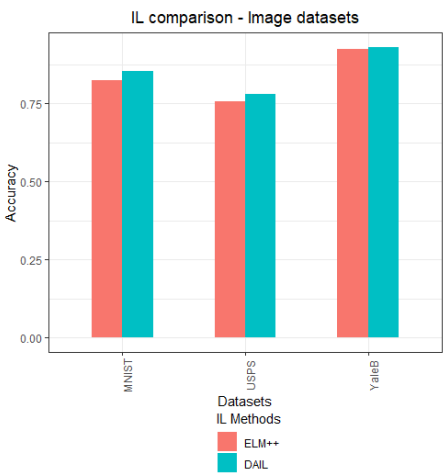
4.4 Incremental Learning Results

The clustering process labels each batch of data and trains using an ELM neural network to create a model. *DAIL* achieves incremental learning through data augmentation, and the most recent model retains the information of all classes learned so far. *DAIL* compares its classification accuracy with an existing incremental learning methodology *ELM++*, and Table 5 tabulates the results for all the above-mentioned standard datasets. Figure 6 (a) and 6 (b) shows the comparison results between *ELM++* and *DAIL* for image and numerical datasets respectively.

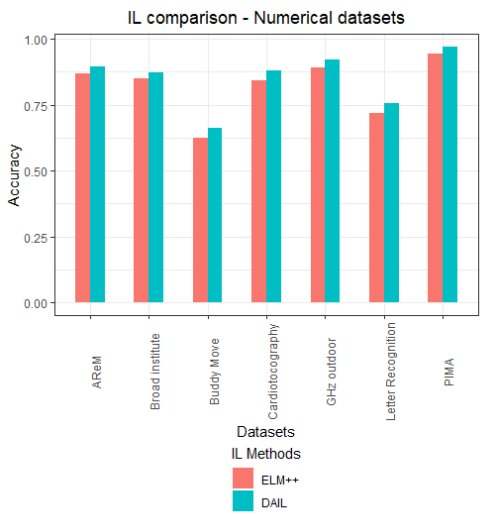
ELM++ [31] is an incremental learning methodology, where ELM trains and creates an individual model for every incoming data batch. Every generated model contains information only about the arrived batch classes, so storing all the previously generated models and the currently developed model is necessary. One or more stored models classifies the test data by comparing the class and sample mean. If a specific class C comes in two different batches, $B1$ and $B2$, both the models $M1$ and $M2$ will contain information about the class. So, if the mean of the test data matches class C, both these models are used to classify the data. Later, the classified results from models $M1$ and $M2$ are combined using a combiner to output the class of the test data.

Experiments are conducted in such a way that, readers get the glimpse of how incremental learning happens, for this purpose we split the datasets into two batches. Table 5 tabulates the information how datasets are split. Our experiments are carried out to address all three possible scenarios: i) Batch $B2$ data contains new clusters, ii) Batch $B2$ data does not include any/few of the old clusters and iii) Batch $B1$ and $B2$ data have overlapping clusters.

Every batch in a dataset holds a certain number of classes. Table 5 tabulates the classification accuracy

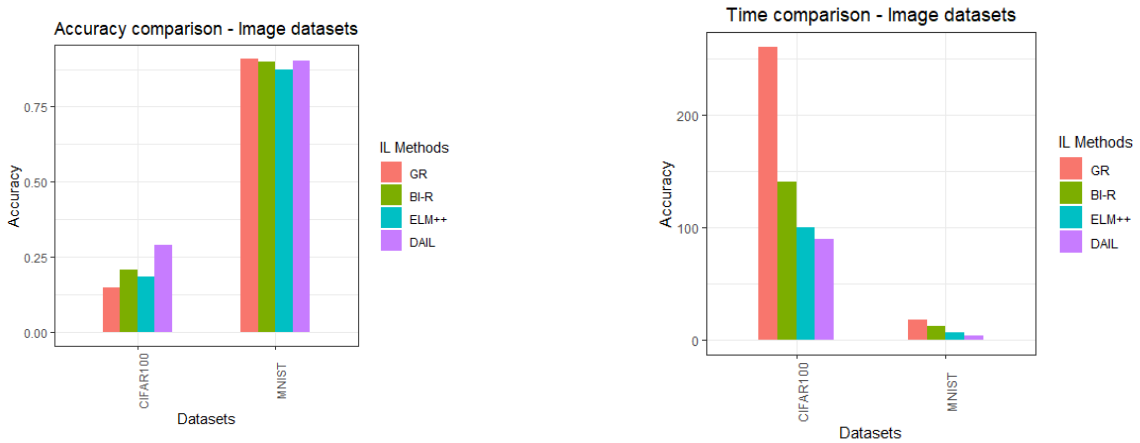


(a) Image datasets



(b) Numerical datasets

Fig. 6 Performance comparison between ELM++ and DAIL: (a) image datasets, (b) numeric datasets



(a) Accuracy

(b) Time

Fig. 7 Performance comparison between Incremental methods on image datasets: (a) accuracy, (b) time

Table 6 Performance comparison between various incremental methods on Image datasets

S.No	Dataset	Accuracy Performance				Time (minutes)			
		GR	BI-R	ELM++	DAIL	GR	BI-R	ELM++	DAIL
1.	**MNIST**	0.91	0.90	0.875	0.902	18	12	7	4
2.	**CIFAR100**	0.15	0.208	0.185	0.29	260	140	100	90

achieved on individual training of each set using both *ELM++* and *DAIL*. For example, in the *Letter recognition dataset*, batch $B1$ is trained for ten classes ($C1 - C10$), and batch $B2$ trains twelve classes ($C9 - C20$). The individual testing results show how accurately the test data is classified using each separate ELM model created, where the test data constitute samples from all 20 classes ($C1 - C20$). The incremental testing shows how accurately the test data is classified using the incremental methodologies *ELM++* and *DAIL*, where *ELM++* classifies by combining the created models. *DAIL* uses the most recently generated model.

4.5 Efficiency of DAIL

Most of the existing incremental algorithms [34] are implemented only for image datasets. So, to further experiment *DAIL* on its classification accuracy and performance time, we compared *DAIL* rigorously against a few existing incremental methods such as *Generative Replay (GR)*, *Brain-inspired replay (BI-R)* and *ELM++* with two image datasets (MNIST and CIFAR-100). To perform the comparison, we split the MNIST dataset into five tasks/episodes, where each episode learns two classes. Similarly, we spilt the CIFAR-100 dataset into ten episodes/tasks, where each task learns ten classes. Table 6 summarizes the results, whereas Figs. 7 (a) and 7 (b) displays the plots for accuracy and time comparison, respectively.

From Table 6, we found that the methods *GR*, *BI-R* and *DAIL* gives almost similar accuracy results for smaller datasets like MNIST. However, when a large dataset like CIFAR100 is tested, *DAIL* shows a slightly better performance than *GR* and *BI-R*. The performance of *ELM++* degrades with an increasing number of classes. So, the methods *GR*, *BI-R* and *ELM++* perform well with image datasets holding fewer classes, whereas *DAIL* consistently performs with every dataset irrespective of the number of classes.

Table 7 Comparison of various incremental learning methods

Parameter	Incremental Methods			
	GR	BI-R	ELM++	DAIL
Supported Datasets	Image	Image	Image, Numeric	Image, Numeric
Supports Streaming Data	No	No	Yes	Yes
Retraining of old data	Yes – regenerates the trained samples	Yes – regenerates the trained samples	No – retains all created models	No – generates new samples
Data Augmentation	No	No	No	Yes
Stored trained samples	No	No	No	No
Adopted architecture	VAE+GMM	VAE+GMM	ELM	SAE+GMM for clustering, ELM for classification
Stability / Plasticity dilemma	No	No	Yes	Yes
Category Proliferation	Yes	Yes	Yes	Yes
Accuracy	Low	High	Medium	High
Time	Very High	High	Medium	Low

The experiments carried out to evaluate the execution time prove the advantage of *DAIL* against other incremental methods. *GR* method, a base version of the *BI-R* method, consists of two variational autoencoder models – one model trains the data, whereas the other model regenerates the old data. As a modified structure, *BI-R* merges the two VAE models into one. So, the execution time of *BI-R* is always better than *GR* for all datasets.

DAIL outperforms both these methods for the following reasons, i) *DAIL* augments new samples by mutating samples of old trained classes, ii) *DAIL* uses *ELM* for training purposes, which involves no iterations. From the obtained results for datasets, YaleB and USPS give more or less equal classification accuracy. There is a negligible difference in execution time as in MNIST since the number of classes is fewer in number. Table 7 provides a summary of all the incremental methods.

5. Discussion

5.1 Brain-Inspired Replay vs DAIL

The incremental method *Brain-Inspired replay (BI-R)* used

the Variational Auto-Encoder to generate samples learned previously. Variational Auto-Encoder generates the old samples and serves the purpose of data augmentation. *Gaussian Mixture Model (GMM)* generates specific desired classes while regenerating samples. The following points in the *Brain-inspired replay* method for continual learning are noteworthy:

- The method tries to reconstruct an already trained sample with a different image quality, which is a kind of retraining.
- The model regenerates an old image only on its final task. The model needs to be updated at the end of each task in the case of streaming data. In such cases, the *Brain-inspired replay (BI-R)* method would fail because if the same sample gets reconstructed for five or more tasks, the quality of the image gets even worse when we use the hidden representations. It may also lead to a change in the prediction of the class label.
- The method also involves the implementation of a gating network, depending on the task or classes to be trained.

In contrast to all the earlier points, *DAIL* handles streaming data and updates the model every time a new batch arrives. It generates new samples for every new batch, which avoids retraining old samples. It also uses an effortless technique of alternating the feature values between two samples belonging to the same class, ensuring the assignment of a new sample to the same class. Furthermore, our work uses the *ELM* neural network, which trains the data quickly.

5.2 Importance of Dimensionality Reduction in CLAG

CLAG supports dimensionality reduction of inputs, which helps achieve clustering results in fewer iterations. The illustration of the importance of dimensionality reduction is as follows. An *MNIST* image is of size 28*X*28, whose features, when given directly to a GMM clustering algorithm, takes a long time (around 1000 – 1500 iterations) to converge, forming consistent clusters. In *CLAG*, we maintain an SAE architecture with three hidden layers stacked with 784 – 500 – 350 – 200 neurons for the *MNIST* dataset. As a result, SAE reduces each image from 784 features to 200 features, which undergoes the clustering process with the embedded GMM and achieves convergence in less than 100 iterations.

5.3 Importance of Data Augmentation

As mentioned in the above sections, in *DAIL*, data augmentation plays a vital role in supporting incremental learning of real-time unsupervised data. What is the need for data augmentation even when it generates a surplus of extra samples for classification? The below two scenarios explain the necessity of data augmentation in supporting incremental learning.

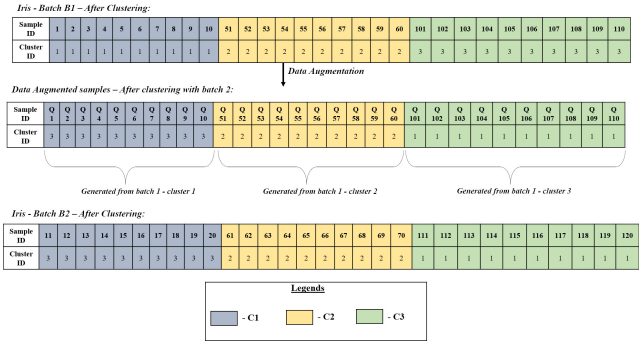


Fig. 8 Combining similar clusters using data augmentation

Scenario 1: When the same clusters occur on every batch of data

Consider the *iris* dataset[†] with 150 samples and 3 classes. Each class in the *iris* dataset has 50 representative samples. In the dataset, samples 1 – 50 belong to *cluster 1 – Setosa (C1)*, samples 51 – 100 belong to *cluster 2 – Versicolor (C2)*, and samples 101 – 150 belong to *cluster 3 – Virginica (C3)*. To illustrate the scenario, let's separate the dataset into two batches – *batch B1* will hold samples from all the three clusters, 1 – 10, 51 – 60 and 101 – 110; *batch B2* will also keep samples from the same previously learned three clusters, 11 – 20, 61 – 70 and 111 – 120. Both the batches have different samples from the same clusters. Neither we add nor delete any new cluster. Is data augmentation needed to support incremental learning in such a scenario? Yes, Fig. 8 explains the importance of data augmentation in such a scenario, combining similar clusters. The cluster that data augments the extra samples and the cluster in which they fall are identical, as illustrated in Fig. 8.

Figure 8 shows that samples 1 – 10 from *batch B1* fall in *cluster 1* and labelled as 1, while samples 11 – 20 from *batch B2* fall in the same cluster but labelled as 3. But, both these set of samples must have the same label. Data Augmentation helps in the relabelling process. The figure shows that data augmented samples $Q1-Q10$ generated from samples 1 – 10 in *batch B1*, groups with samples 11 – 20 in *batch B2*, thus relabelling the similar clusters.

Scenario 2: Advantage of DAIL over ELM++ The advantages of *DAIL* over *ELM++* in achieving incremental learning are the following: i) ELM++ stores all the generated models, whereas *DAIL* stores only the most recently created model, holding information on all the learned classes; ii) in some cases, *ELM++* classifies the test data using more than one model, whereas *DAIL* uses only one model to classify any test data; iii) *ELM++* requires the need of a combiner when a test data is classified using more than one model, whereas *DAIL* does not require any combiner.

5.4 DAIL Handling Variety of Data

DAIL uses a simple data augmentation approach for both

[†]Data Augmentation technique works for all datasets. Iris dataset is just shown for a simple, clear-cut illustration

images and numerical data to support incremental learning. For images, more complex data augmentation techniques like mirroring, flipping, random cropping etc are available. Genetic algorithms may also be used to augment data, but these methods does not guarantee whether the new samples fall into the same cluster as its parent. But, *DAIL* follows a simple column-alternating approach which guarantees the same cluster samples for all types of data.

6. Conclusion

Incremental Learning serves as an added advantage in scenarios like credit card fraud detection or any e-commerce applications, where data flow in every second. This paper presents an incremental learning method, Data Augmented Incremental Learning (DAIL), which consists of two levels, to build an incremental model. As *DAIL* concentrates on unlabeled data, *CLAG* clusters the data using autoencoder and gaussian model. To handle unpredictable data changes and to avoid the reuse of already learned data, augmentation is performed and classification is done using ELM. *CLAG* successfully clusters the data (10 benchmark datasets taken from UCI of two types numerical and image) and also incorporates dimensionality reduction to form the clusters in fewer iterations. *CLAG* performance is compared with SAE-K-Means and GMM, results state that performance of *CLAG* is reliable in-terms of accuracy for both type of datasets. *DAIL* saves time by avoiding the retraining of old samples and reduces memory resources by discarding all old models. *DAIL* performance was experimented and tested using 10 benchmark datasets, whose results shows the reliability in-terms of accuracy and model generation. DAIL can deal with credit card or insurance applications involving numeric data and be applicable in health care domains. Although the results are promising, some improvements has to be done in these areas, i) focusing on handling concept drift for real-time streaming data and ii) able to handle and build incremental models for sentiment classification of feeds.

References

[1] D. Patel, R. Modi, and K. Sarvakar, "A comparative study of clustering data mining: Techniques and research challenges," International Journal of Latest Technology in Engineering, Management & Applied Sciences, III(IX), pp.67–70, 2014.

[2] S. Saraswathi and M.I. Sheela, "A comparative study of various clustering algorithms in data mining," International Journal of Computer Science and Mobile Computing, vol.3, no.11, pp.422–428, 2014.

[3] K. Weiss, T.M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," Journal of Big Data, vol.3, no.9, pp.1–40, 2016.

[4] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," Artificial Neural Networks and Machine Learning – ICANN 2018, Springer International Publishing, pp.270–279, 2018, ISBN:978-3-030-01424-7.

[5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," Advances in Neural Information Processing Systems, vol.27, 2014.

[6] Y. Luo, L. Yin, W. Bai, and K. Mao, "An Appraisal of Incremental Learning Methods," Entropy, vol.22, no.11, 1190, 2020, DOI:10.3390/e22111190.

[7] S.S. Sarwar, A. Ankit, and K. Roy, "Incremental Learning in Deep Convolutional Neural Networks Using Partial Network Sharing," IEEE Access, vol.8, pp.4615–4628, 2020, DOI:10.1109/access.2019.2963056.

[8] F.M. Castro, Manuel J. Marín-Jiménez, N. Guil, C. Schmid, and A. Karteek, "End-to-End Incremental Learning," Computer Vision – ECCV 2018, pp.241–257, 2018, ISBN:978-3-030-01258-8.

[9] H. Bouchachia, B. Gabrys, and Z. Sahel, "Overview of Some Incremental Learning Algorithms," Proc. 2007 IEEE International Conference on Fuzzy Systems, pp.1–7, 2007, DOI:10.1109/FUZZY.2007.4295640.

[10] C. Shorten and T. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," Journal of Big Data, vol.6, no.60, pp.1–48, 2019.

[11] M. Milicevic, K. Zubrinic, I. Obradovic, and T. Sjekavicar, "Data augmentation and transfer learning for limited dataset ship classification," WSEAS Transactions on Systems and Control, vol.13, pp.460–465, 2018.

[12] G. Liu, H. Bao, and B. Han, "A Stacked Autoencoder-Based Deep Neural Network for Achieving Gearbox Fault Diagnosis," Mathematical Problems in Engineering, vol.2018, pp.1–10, 2018, DOI:10.1155/2018/5105709.

[13] D. Bhowick, D.K. Gupta, S. Maiti, and U. Shankar, "Stacked autoencoders based machine learning for noise reduction and signal reconstruction in geophysical data," arXiv:1907.03278, 2019.

[14] R. Douglas, "Gaussian Mixture Models," Encyclopedia of Biometrics, pp.827–832, 2015, DOI:10.1007/978-1-4899-7488-4_196.

[15] S. Ding, X. Xu, and R. Nie, "Extreme learning machine and its applications," Neural Computing and Applications, vol.25, pp.549–556, 2013.

[16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing, vol.70, no.1-3, pp.489–501, 2006, DOI:10.1016/j.neucom.2005.12.126.

[17] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," Annals of Data Science, vol.2, no.2, pp.165–193, 2015, DOI: 10.1007/s40745-015-0040-1.

[18] S. Bano and N. Khan, "A Survey of Data Clustering Methods," International Journal of Advanced Science and Technology, vol.113, pp.133–142, 2018, DOI: 10.14257/ijast.2018.113.14.

[19] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture," IEEE Access, vol.6, pp.39501–39514, 2018, DOI:10.1109/ACCESS.2018.2855437.

[20] A. Shukla, G.S. Cheema, and S. Anand, "Semi-Supervised Clustering with Neural Networks," IEEE Sixth International Conference on Multimedia Big Data (BigMM), India, pp.152–161, 2020, DOI: 10.1109/BigMM50055.2020.00030.

[21] C. Song, F. Liu, Y. Huang, L. Wang, and T. Ta, "Auto-encoder Based Data Clustering," CIARP, Lecture Notes in Computer Science, vol.8258, pp.117–124, 2013, DOI: 10.1007/978-3-642-41822-8_15.

[22] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep Clustering with Convolutional Autoencoders," Neural Information Processing, pp.373–382, 2017, DOI:10.1007/978-3-319-70096-0_39.

[23] D.V. Prasad, S. Madhusudanan, and S. Jaganathan, "uCLUST-a new algorithm for clustering unstructured data," ARPN Journal of Engineering and Applied Sciences, vol.10, pp.2108–2117, 2015.

[24] S. Young, I. Arel, T. Karnowski, and D. Rose, "A Fast and Stable Incremental Clustering Algorithm," Seventh International Conference on Information Technology: New Generations, pp.204–209, 2010, DOI:10.1109/ITNG.2010.148.

[25] J. Bao, W. Wang, T. Yang, and G. Wu, "An incremental clustering method based on the boundary profile," PLOS ONE, Public Library of Science, vol.13, no.4, pp.1–19, 2018, DOI:10.1371/journal.pone.0196108.

[26] M. Ackerman and S. Dasgupta, "Incremental clustering: The case for extra clusters," Advances in Neural Information Processing Systems, vol.27, 2014.

[27] M.D. Woodbright, M.A. Rahman, and M.Z. Islam, "A Novel Incremental Clustering Technique with Concept Drift Detection," arXiv: 2003.13225, 2020.
[28] S. Chakraborty, N.K. Nagwani, and L. Dey, "Performance comparison of incremental kmeans and incremental DBSCAN algorithms," International Journal of Computer Applications, vol.27, no.11, pp.14–18, 2011, DOI:10.5120/3346-4611.
[29] O.J. Oyelade, O.O. Oladipupo, and I.C. Obagbuwa, "Application of k means clustering algorithm for prediction of Students academic performance," International Journal of Computer Science and Information Security, vol.7, no.1, pp.292–295, eprint:1002.2425, 2010.
[30] S. Na, L. Xumin, and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," Third International Symposium on Intelligent Information Technology and Security Informatics, pp.63–67, 2010, DOI:10.1109/IITSI.2010.74.
[31] S. Madhusudhanan, S. Jaganathan, and L.S. Jayashree, "Incremental Learning for Classification of Unstructured Data Using Extreme Learning Machine," Algorithms, vol.11, no.10, p.158, 2018, DOI: 10.3390/a11100158.
[32] R. Cohn and E. Holm, "Unsupervised machine learning via transfer learning and k-means clustering to classify materials image data," arXiv:2007.08361, 2020.
[33] S. Jaganathan and S. Madhusudhanan, "Polarity Classification of social media feeds using incremental learning - A Deep Learning Approach," IEICE Trans. Fundamentals, vol.E105-A, no.3, pp.584–593, 2022, DOI:10.1587/transfun.2021EAP1046.
[34] G.M. van de Ven, H.T. Siegelmann, and A.S. Tolias, "Brain-inspired replay for continual learning with artificial neural networks," Nature Communications, vol.11, 4069, 2020, DOI:10.1038/s41467-020-17866-2.

Sathya Madhusudhanan is a Research scholar at Anna University pursuing her research in Deep Learning. She did her Masters in Computer Science and Engineering and obtained bachelors degree from Anna University. To her credits, she has published papers in reputed International Conferences and Journals and also filed one patent. Her areas of interest are Data Analytics & Deep Learning.

Suresh Jaganathan is an Associate Professor in the Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, has more than 26 years of teaching experience. He received his PhD in Computer Science from Jawaharlal Nehru Technological University, Hyderabad, M.E Software Engineering from Anna University and B.E Computer Science & Engineering, from Madurai Kamarajar University, Madurai. He has more than 30 publications in refereed International Journals and Conferences. Apart from this, to his credit, he has two patents in the area of image processing and written a book on "Cloud Computing: A Practical Approach for Learning and Implementation", published by Pearson Publications. He is an active reviewer in reputed journals (Elsevier - Journal of Networks and Computer Applications, Computer in Biology and Medicine) and also co-investigator for the SSN-NIVIDA GPU Education centre. His areas of interest are Distributed Computing, Deep Learning, Data Analytics, Machine learning & Blockchain Technology.