International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 5, October 2023, pp. 5853~5864
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i5.pp5853-5864

Convolutional auto-encoded extreme learning machine for incremental learning of heterogeneous images

Sathya Madhusudanan, Suresh Jaganathan, Dattuluri Venkatavara Prasad
Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India

Article Info

Article history:

Received Dec 10, 2022
Revised Mar 29, 2023
Accepted Apr 7, 2023

Keywords:

Autoencoder
Convolutional neural networks
Extreme learning machine
Heterogeneous data
Incremental learning

ABSTRACT

In real-world scenarios, a system's continual updating of learning knowledge becomes more critical as the data grows faster, producing vast volumes of data. Moreover, the learning process becomes complex when the data features become varied due to the addition or deletion of classes. In such cases, the generated model should learn effectively. Incremental learning refers to the learning of data which constantly arrives over time. This learning requires continuous model adaptation but with limited memory resources without sacrificing model accuracy. In this paper, we proposed a straightforward knowledge transfer algorithm (convolutional auto-encoded extreme learning machine (CAE-ELM)) implemented through the incremental learning methodology for the task of supervised classification using an extreme learning machine (ELM). Incremental learning is achieved by creating an individual train model for each set of homogeneous data and incorporating the knowledge transfer among the models without sacrificing accuracy with minimal memory resources. In CAE-ELM, convolutional neural network (CNN) extracts the features, stacked autoencoder (SAE) reduces the size, and ELM learns and classifies the images. Our proposed algorithm is implemented and experimented on various standard datasets: MNIST, ORL, JAFFE, FERET and Caltech. The results show a positive sign of the correctness of the proposed algorithm.

Corresponding Author:

Suresh Jaganathan
Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering
Chennai, Tamilnadu, India
Email: sureshj@ssn.edu.in

1. INTRODUCTION

Information systems widely differ in their form and application, but converts all the data into meaningful information. Real-world applications generate data in huge volumes, where the process of acquiring knowledge becomes complex. Irrespective of the type of data, which may be homogeneous (same feature set across the chunks) or heterogeneous (different feature set across the chunks) [1], the models generated from the systems must continually learn to predict or classify. Incremental learning (or constructive learning) [2], a machine learning technique introduced for continual learning, updates the existing model when data streams in continually. Figure 1 shows an incremental learning model, which helps grow the network with the data arrival belonging to new classes. It applies on classification and clustering applications, addressing the data availability and resource scarcity challenges. An incremental learning algorithm must meet these criteria [3], i) accommodating new classes, ii) minimal overhead for training new classes, iii) previously trained data must not be retrained, and iv) preserve previously acquired knowledge. Challenges faced by incremental learning are:

a) Concept drift: refers to the changes observed in the data distribution over time [4], which falls under two categories: i) virtual concept drift or covariate concept drift where changes are seen in the data distribution

only and ii) real concept drift where changes are seen in the underlying functionality. Concept drift impacts the system performance until the model re-adapts accordingly.

b) Stability-plasticity dilemma: refers to how quickly the model adapts with the changes. Two possibilities are: i) the model adapts quickly to the new information, but forgets the old information and ii) in contrast, the older information remains for a more extended period, but the model gets adapted slowly. The trade-off between the two is referred as stability-plasticity dilemma [5].
c) Catastrophic forgetting: also known as catastrophic interference [6]. Many online learning models tend to forget previously learned information completely or abruptly upon learning new information.
d) Adaptive model complexity and meta-parameters: incremental learning [7] also has to address the challenge of model complexity, whose estimation in advance is impossible when huge data streams in. Intelligent adaptation methods deal with the model complexities by reallocating the resources whenever the limit is reached.
e) Efficient learning models: the incremental learning models [8] must be efficient enough to deal with the constantly arriving data. Even in limited resources, these models must deal with the newly added data by storing the information provided by the observed data in compact form.

Our proposed work focuses on designing an efficient incremental learning model for classifying image data available in different static batches with varying resolutions, i.e., 64×64, 39×39, 32×32, 28×28. To achieve efficient incremental learning, the proposed convolutional auto-encoded extreme learning machine (CAE-ELM) neural network uses the correctly classified test samples of every batch as a source of new labeled training samples for the successive batches: i) initially, convolutional neural network (CNN) [9] captures the spatial image features; ii) later, stacked auto-encoder (SAE) [10] does the process of dimensionality reduction on the extracted features; and iii) finally, extreme learning machine (ELM) [11] trains the image batches and creates an incremental model using the unlabeled test samples of each batch.

CAE-ELM addresses all the above-mentioned challenges by retaining the learned information on old classes and learning new classes arriving in varying batches with varying resolutions. CAE-ELM maintains only the most recently updated model and reduces memory constraints by discarding all the old trained data and model. It also saves training time by avoiding the retraining of old samples.

An extensive survey of the existing feed-forward neural network, ELM and incremental learning methodologies are done to update our knowledge in the area of our proposed work. The survey confirms the extensive use of ELM for cross-domain tasks and unsupervised learning. Furthermore, the detailed study of various incremental architectures helped us understand each strategy's pros and cons, which made us choose a replay-based technique for our incremental algorithm.

a) Extreme learning machine

Zhang *et al.* [12] optimized ELM using the Firefly algorithm. Song *et al.* [13], Huang *et al.* [14] proposed the improved versions of ELM by incrementally adding hidden nodes to the ELM network, which improved ELM's performance and validated the universal approximators, respectively. Wang *et al.* [15] briefs and reviews the fast ELM algorithm and its applications in different domains. Yousefi-Azar and McDonnell [16] proposed the unsupervised CNN-ELM (UCNN-ELM) for unsupervised learning. Adaptive ELM, an extension to ELM, proposed by Zhang *et al.* [17] for cross task learning uses a new error term and Laplacian graph based manifold regularization term in the objective function. Zhang *et al.* [18] makes a comparison between ELM and SVM in cross-domain tasks, where ELM outperforms SVM with about 4% accuracy.

b) Incremental learning architectures

Discussed here are a few research works contributing to the area of incremental learning. We have briefed two architectures for implementing incremental learning: i) machine learning techniques and ii) neural networks (NN). In neural network architectures, we categorize three strategies to implement incremental learning: i) replay-based methods, ii) regularization-based methods, and iii) parameter isolation methods.

c) Machine learning approaches

Losing *et al.* [19] discussed and compared the following incremental algorithms, LASVM (online approximate SVM) [20] checks each current instance for becoming a support vector and removes all the obsolete support vectors. Incremental learning vector quantization (ILVQ) [21] dynamically creates new prototypes for classes depending on the number of classifiers. NBGauss (naive Bayes) [22] uses likelihood estimation in the naïve-Bayes algorithm by fitting Gaussian distribution on one class at any instance. Stochastic gradient descent (SGDLin+) [23] combines SGD with linear models to minimize loss function and optimize the performance of discriminative models.

The studies reveal that SVM delivers the highest accuracy at the expense of the most complex model. LASVM reduces the training time, though ORF gives a poor performance, it can be considered for high-speed training and run-time. ILVQ offers an accurate and sparse alternative to the SVMs. NBGauss, SGDLin are viable choices for large-scale learning in high-dimensional space.

d) Neural network architectures

Replay-based methods: replay-based methods replay training samples from previous tasks into the new batch of training data to retain the acquired knowledge. The replay can be done in two ways: a) creating exemplars and b) generating synthetic data. A deep incremental CNN [24] model, where the exemplars of the old classes update the loss function when trained with the new class samples. This architecture used for large-scale incremental learning, where exemplars help in bias correction for the created model representing both the old and new classes. ELM++ [25], an incremental learning algorithm, creates a unique model for each incoming batch. The model is picked by using the intensity mean of the test sample and the classes trained. Here authors included a data augmentation step in generating samples to accommodate old and new classes in the incremental model, using adjusting brightness, contrast normalization, random cropping, and mirroring techniques. Deep generative replay for continual learning of image classification tasks and constitutes a deep generative model ("generator") and a task solving model ("solver"). Brain-inspired replay (BI-R) [26] uses the variational auto-encoder for generating samples learned previously. Variational auto-encoder generates the old samples and serves the purpose of data augmentation. It also uses Gaussian mixture model (GMM) to generate specific desired classes while regenerating trained samples.

Regularization-based methods: regularization-based methods use regularization terms to prevent the current task parameters from deviating too much from the previous task parameters. A deep incremental CNN architecture can use a strong distilling and classification loss in the last fully connected layer to effectively overcome the catastrophic forgetting problem. He *et al.* [27] proposed a two-step incremental learning framework with a modified cross-distillation loss function addressing the challenges in online learning scenario.

Parameter isolation methods: parameter isolation methods eradicate catastrophic forgetting by dedicating a subset of a model's parameters from previous tasks to each current incremental task. Guo *et al.* [28] implemented incremental ELM (I-ELM), which trains online application data one-by-one or chunk by chunk using three alternatives-minimal-norm incremental ELM (MN-IELM), least square incremental ELM (LS-IELM), kernel-based incremental (KB-IELM). Among the three methods, KB-ELM provides best accuracy results. Online sequential ELM (OS-ELM) [29] trains data sequentially one by one or chunk by chunk based on recursive least-squares algorithm. In the sequential learning phase, for each new observation, OS-ELM calculates the current hidden and output layer weight based on the previous hidden and output weights. Error-minimized ELM (EM-ELM) [30] adds random hidden nodes to ELM network one by one or chunk by chunk and reduces the computational complexity by updating the output weights incrementally. Learn++ [31], an incremental training algorithm trains an ensemble of weak classifiers with different distributions of samples. A majority voting scheme generates the final classification rule, which eliminates over-fitting and fine-tuning. ADAIN [32], a general adaptive incremental learning framework, maps the distribution function estimated on the initial dataset to the new chunk of data. The pseudo-error value assigns the misclassified instances with a higher weight, making an efficient decision boundary. A non-iterative, incremental, hyperparameter-free learning method iLANN-SVD, updates the network weights for the new set of data by applying singular value decomposition on the previously obtained network weights.

Our proposed work comes more or less under the first category, as the correctly classified test samples of the current batch serve as augmented samples. Thus, the next set of images generates the incremental model along with the augmented samples. Table 1 (see in Appendix) summarizes the strategies adopted by various neural network architectures to achieve incremental learning.

The rest of the article is organized as follows: the contributions listed in section 1 discuss the strategies to achieve incremental learning. The proposed work, CAE-ELM, is detailed in section 2. The experimental results and implementation of sample scenarios can be found in section 3 and also it discusses the pros and cons of CAE-ELM. Finally, section 4 concludes the paper with possible future work.

2. METHOD

CNN-ELM [33] is a recently developed deep neural network that replaces the multi-layer perceptron classifier part of CNN with the extremely fast ELM neural network. The flattened features of the input image retrieved by applying convolutional filters are fed to the ELM to fasten the classification process. CNN is combined with ELM to achieve the following benefits: i) the convolutional filters in CNN capture an image's spatial dependencies, classifying it with significant precision, ii) ELM randomly assigns the hidden node parameters initially, and they are never updated, and iii) ELM learns the output weight in a single step. Thus, it gives the best generalization performance at a breakneck learning speed.

Exploiting the advantages of CNN-ELM, we propose the novel CAE-ELM, an incremental learning algorithm for classifying supervised images. CAE-ELM is designed to solve two issues: i) classifies images with varying dimensions and ii) performs incremental learning. CAE-ELM classifies the images arriving in different batches with varying resolutions. CAE-ELM constitutes the following: i) CNN extracts feature from input images, ii) SAE does the dimensionality reduction process, and iii) ELM incrementally trains every batch

of images and generates an updated model with no retraining of samples. Figure 1 outlines the overview of the proposed work CAE-ELM. (Note: CAE-ELM framework embeds only one CNN, SAE, and ELM architecture for the entire incremental image classification process).

As CAE-ELM focuses on handling varied resolution images arriving in different batches, initially, CNN outputs varied size features from the flattened layer for every batch. Before the training process, we zero-pad the extracted varied size features from CNN to a fixed length (maximum length among the extracted features). The SAE dimensionally reduces the zero-padded features to ease the training process. Finally, the ELM neural network trains and classifies the dimensionally reduced features extracted from the varied resolution input images.

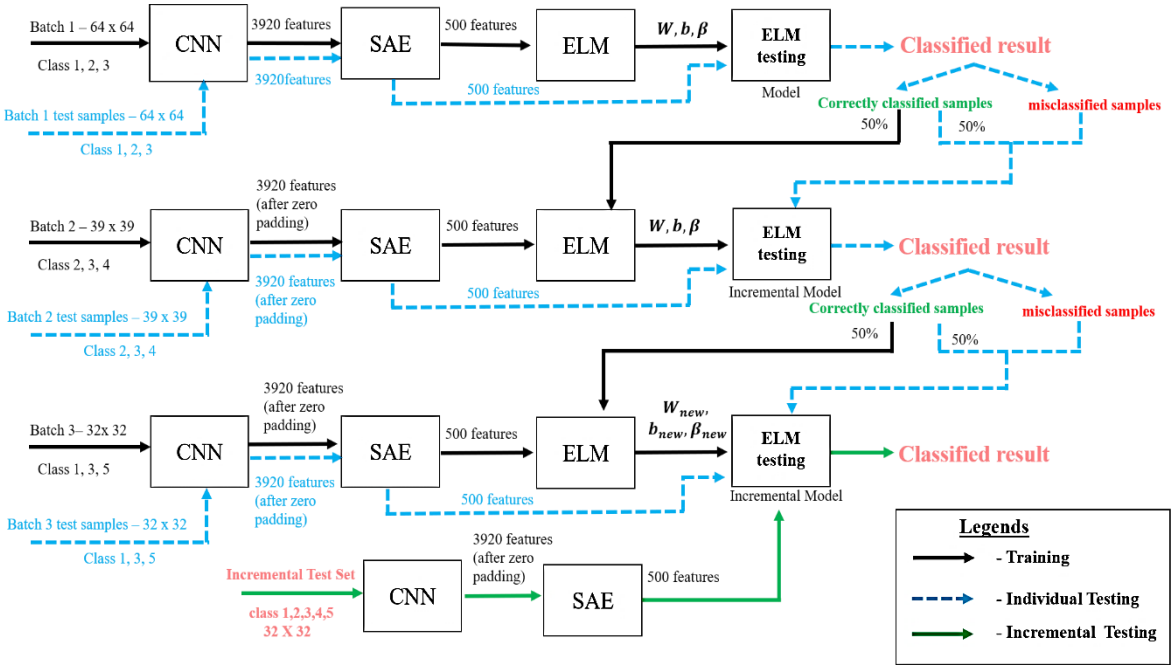


Figure 1. Incremental learning using CAE-ELM

2.1. CAE-ELM for image classification with varied resolution

In CAE-ELM, the CNN framework had two convolutional layers and two pooling layers alternatively-*conv1* with ten 3×3 filters, *pool1* with ten 2×2 filters, *conv2* with twenty 3×3 filters and *pool2* with twenty 2×2 filters. In our proposed work, the images arrive in batches with varying resolutions each day, i.e., batch 1 holds images with 64×64, batch 2 carries images with 39×39, and batch 3 holds images with 32×32 resolution. The CNN with the designed framework produces flattened features with varying sizes for each batch of images fed into it. For example, batch 1, batch 2 and batch 3 has 3920, 800, 720 features, respectively. The output dimension of each layer in the CNN is,

$$\left[\frac{(w+2p-cf)}{s}+1\right]\times\left[\frac{(h+2p-cf)}{s}+1\right]\times n_{cf} \quad (1)$$

where w is image width, h is height, p is image padding, s is stride, cf is convolutional filter size and n_{cf} is the number of convolutional filters applied. We zero-pad the extracted features from varied batches to a fixed-length features. For example, when the batch images end up with varied sized features 3920, 800, 720, we zero-pad the features to a maximum length 3290.

SAE dimensionally reduces the zero-padded features to ease the training process. The autoencoder generates a compact code representation, which successfully reconstructs the input image. The dimensionally reduced features from the stacked autoencoder are fed into the ELM, a single hidden layer neural network. ELM generates a model with parameters (W, b, β). Algorithm 1 explains how SAE converts the varied resolution images to the same features and the ELM training process for image classification.

Algorithm 1. CAE-ELM

Input: Input training images
Output: Classified Output
for batches $i = 1,2,\ldots,S$
A. Do the following steps to extract features from every image $j = 1,2,\ldots,N$ in the dataset:
   1. Convolve the input image $(w\,x\,h\,x\,d)$ with (n_{cf}) filters to produce the convolved image $(w \times h \times n_{cf})$.
   2. Pool the convolved image, reducing the dimensions to $\left(\frac{w}{2} \times \frac{h}{2} \times \frac{n_{cf}}{2}\right)$, which helps reduce the computation time, overfitting, and the need for substantial memory resources.
   3. Flatten the pooled output into a single 1-D array feature $f(x)(of\ size\ n_1)$.
B. Reduce the single 1-D array feature $f(x)\ of\ size\ n_1$ to a constant feature $size\ n_p$ using Stacked Autoencoder, such that $\forall_i n_p < n_i$.
C. ELM neural network trains the compact feature set $f(x)$ and generates a classifier model. The (2) computes the single hidden layer (H). The (3) computes the output matrix β.

$$H = \begin{pmatrix} g(f(x_1).W_1 + b_1) & \cdots & g(f(x_1).W_{N'} + b_{N'}) \\ \vdots & \ddots & \vdots \\ g(f(x_N).W_1 + b_1) & \cdots & g(f(x_N).W_{N'} + b_{N'}) \end{pmatrix} \quad (2)$$

where N': hidden neurons and W: random weights are assigned. The output matrix β is given by (3):

$$\beta = H^{\dagger}T \quad (3)$$

where,

$$H^{\dagger} = \begin{cases} H^t(HH^t)^{-1}, if\ m < f(x) \\ (HH^t)^{-1}H^t, if\ m > f(x) \end{cases} \quad (4)$$

and T is a target matrix, t represents the transpose of a matrix, m represents output neurons.
Endfor

2.2. CAE-ELM for incremental learning

Incremental learning methodology learns the arriving data batches continuously, updates the current model information addressing stability-plasticity dilemma, and overcomes the problem of catastrophic forgetting. CAE-ELM uses two types of testing to check the performance of the generated incremental model. i) Individual testing: the model is tested with class samples as in the individual training batches and ii) Incremental testing: samples belonging to all the classes are tested using the generated model.

CAE-ELM supports incremental learning of images with varied resolutions through the individual test samples of each arriving batch. Individual test samples serve as support to synthesize new training samples to create an incremental model, a form of data augmentation. Each batch of images is divided into training and testing sets. CNN extracts the image features, and SAE reduces the features dimensionally to a countable number of features, which remains common to all the incoming data batches. ELM creates a model by training the samples in the training set. The model parameters generated (W, b, β) classify the test samples of each batch. The proposed work feeds one-half of the correctly classified test samples (50%) along with the following training batch to support incremental learning. In this way, ELM learns the newly arrived classes, retaining the previously acquired knowledge about the old classes with model parameters $(W_{new}, b_{new}, \beta_{new})$. The other half of the correctly classified samples and misclassified samples test the updated model, along with the subsequent batch of the test set. Thus, *CAE-ELM* serves the following advantages: i) learns newly arriving classes, ii) remembers the previously learned information on odd classes, iii) does not retrain any of the input samples and saves training time, and iv) discards old training samples and models when it generates the updated model, thus saving memory space.

2.3. Illustration

Figure 2 illustrates CAE-ELM, which individually trains 3 batches. Batch 1 images constitute three classes 1, 2 and 3 with 64×64 resolution (3920 features). Batch 2 images hold classes 2, 3 and 4 with 39×39 resolution (800 features), while Batch 3 images have resolution 32×32 (720 features) holding classes 1, 3 and 5. Starting from the second batch of data, CAE-ELM implements incremental learning by training one-half of the correctly classified test samples from the previous batch. So, CAE-ELM trains batch 2 data along with one-half of the correctly classified test samples from the batch 1 data, producing model M2 with classes 1, 2, 3 and 4, i.e., both old and new classes. Similarly, the updated model M3 generated on training batch

3 data, holds information on all the classes 1, 2, 3, 4 and 5. CAE-ELM discards all the previous models, and uses the most recent model M3 for classifying the test data.

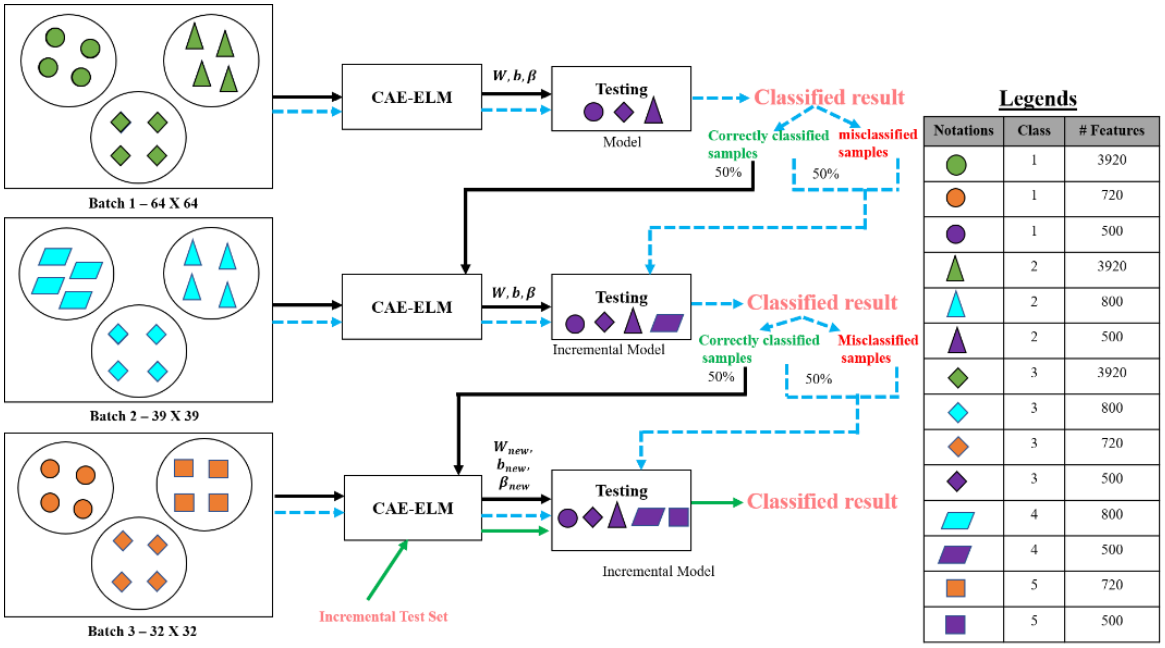


Figure 2. Illustration of incremental learning in CAE-ELM

3. RESULTS AND DISCUSSION

Experiments test the incremental performance of CAE-ELM using standard datasets like MNIST, JAFFE, ORL, FERET. Tables 2 to 5 prove the outstanding incremental performance of CAE-ELM against the existing incremental methods. Subsection 3.3.2 discusses the pros and cons of using CNN, SAE, and ELM in CAE-ELM for the process of feature extraction, feature size conversion (dimensionality reduction), and training for image classification.

3.1. Incremental learning results

We compared CAE-ELM against different incremental methods for MNIST and CIFAR100 datasets as follows:

a) MNIST dataset: MNIST dataset comprises of 10 classes, with a total of 20,000 (28×28) images.

Without adding new classes: for the MNIST dataset with a fewer number of classes, we compared our algorithm CAE-ELM with existing incremental learning algorithms, Learn++, and ELM++, which use neural networks for image classification. We split the MNIST dataset into six sets $(S1 - S6)$, where each set holds samples from classes $0 - 9$. Table 2 (case 1) and Figures 3(a) and 3(b) show the accuracy comparison between Learn++, ELM++, and CAE-ELM for the MNIST dataset when adding new images to the already existing classes. The entire dataset was divided into six train sets $(S1 - S6)$ and one test set. Each train set consists of 10 classes (0-9) with 200 images in each class. The set S2 will hold the same classes but different images from S1. Set S3 will have images that are not present in S1 and S2, and so on. The test set contains images from all these ten classes.

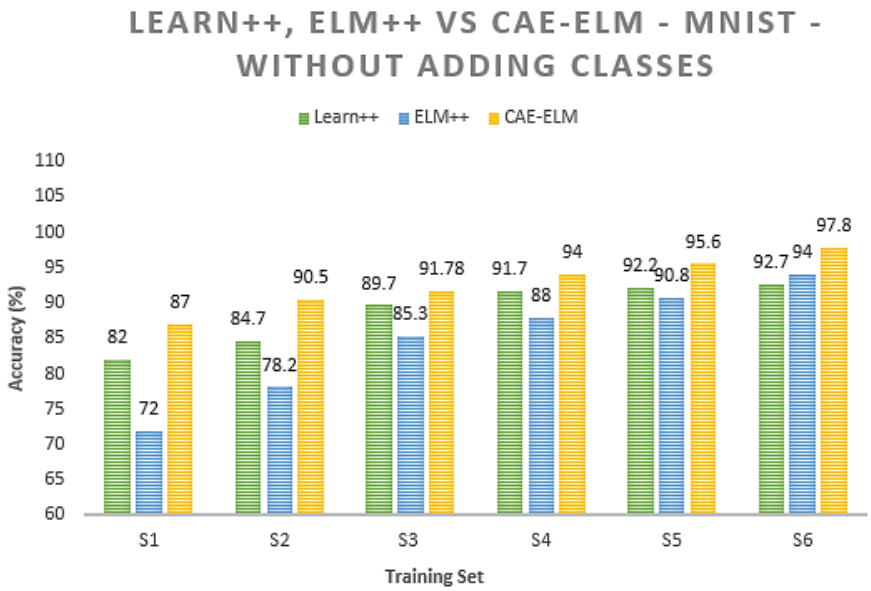
With adding new classes: We also compared CAE-ELM against Learn++ and ELM++ methods with MNIST dataset under scenario where new classes gets added in every new arriving batch of data. We split the MNIST dataset into three sets S1, S2, S3, where S1 trains classes (0-2), S2 trains classes (3-5) and S3 holds samples from classes (6-9). Table 2 (case 2) and Figure 3 tabulates the results obtained.

b) CIFAR dataset: CIFAR dataset comprises of 100 classes, with a total of 60,000 (32×32) images.

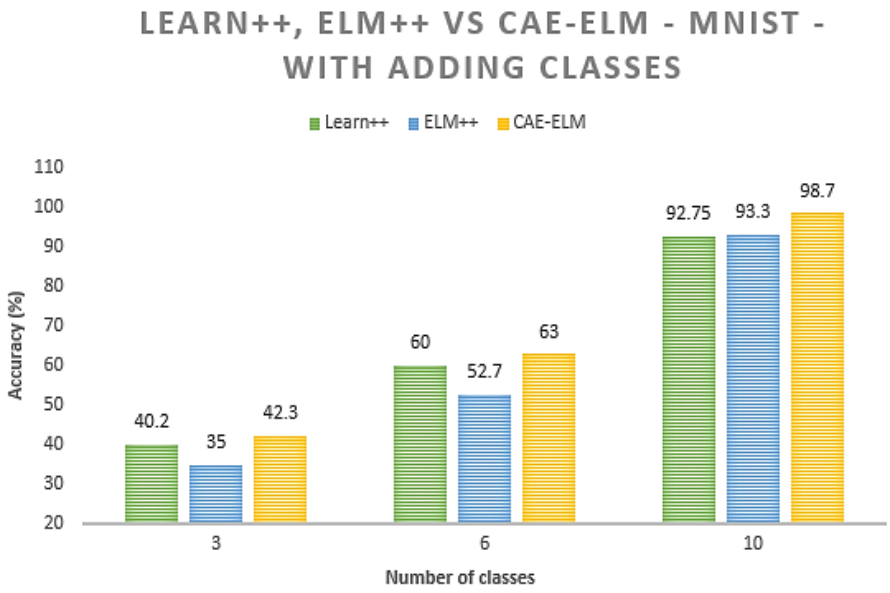
For the CIFAR100 dataset with a larger number of classes, we compared CAE-ELM against brain-inspired replay, end-to-end and large-scale incremental learning methods. We split the dataset into five train sets S1-S5, where every set train 20 new classes in addition to the old classes. Table 3 and Figure 4 tabulates the results obtained.

Table 2. Comparison between learn++, ELM++, and CAE-ELM – MNIST dataset

Case	Set	Training Accuracy (%)			Incremental testing Accuracy (%)		
		Learn++	ELM++	CAE-ELM	Learn++	ELM++	CAE-ELM
Without adding classes	S1(0-9)	94.2	92	95	82	72	87
	S2(0-9)	93.5	90	95.5	84.7	78.2	90.5
	S3(0-9)	95	90	96	89.7	85.3	91.78
	S4(0-9)	93.5	92	96.7	91.7	88	94
	S5(0-9)	95	96	97	92.2	90.8	95.6
	S6(0-9)	95	96	96.8	92.7	94	97.8
With adding classes	S1(0, 1, 2)	98.7	97	96.45	40.2	35	42.3
	S2(3, 4, 5)	96.1	93	98.9	60	52.7	63
	S3(6, 7, 8, 9)	92.6	90.75	92.8	92.75	93.3	98.7



(a)



(b)

Figure 3. Comparison between Learn++, ELM++, and CAE-ELM – MNIST database: (a) comparison between Learn++, ELM++, and CAE-ELM – MNIST database without adding classes and (b) comparison between Learn++, ELM++, and CAE-ELM for MNIST database with adding classes

Table 3. Comparison between BI-R, end-to-end, large-scale incremental learning and CAE-ELM – CIFAR100 dataset (with adding classes)

Set	Number of Classes	Accuracy (%)			
		BI-R	End-to-End	Large-scale	IL
S1	20	74	82	84	86.9
S2	40	43.7	77.6	74.69	77.2
S3	60	35.4	65	67.93	70.8
S4	80	33.1	60	61.25	64.4
S5	100	28.8	53.4	56.69	60.4

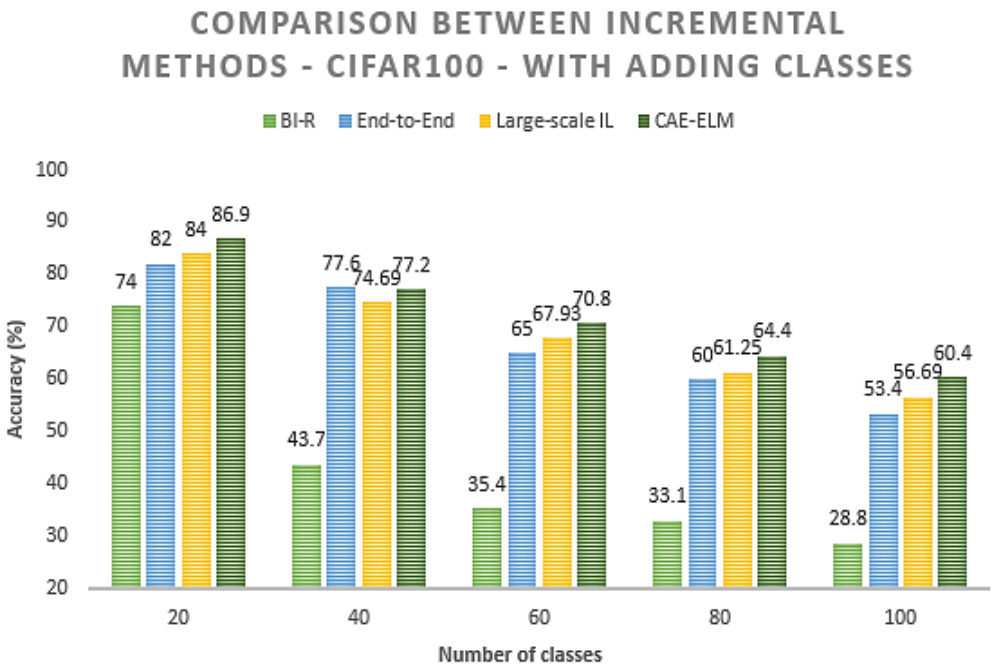


Figure 4. CIFAR100 database with adding classes

3.2. Implementation of application scenario

The proposed CAE-ELM incremental neural network is deployed in various lab to access the systems by different group of students. The student's images will be trained previously by the CAE-ELM, where the training images will be captured using cameras with varying resolutions fixed in different labs. For example, consider five batches (Batch B1, B2, B3, B4, and B5) and four labs (Python, R, MATLAB, and Java). Every batch will hold 10 students, where each student represents a unique class. Every student will learn two different subjects and have access to the corresponding labs. Python lab captures images with 64×64 resolution, R lab captures images with 39×39 images, MATLAB with 32×32 resolution whereas, Java lab with 28×28 resolution images. The student's images will be captured using these lab cameras with varying resolutions, for example, Batch B1 students will have MATLAB and Java classes, so their images will be captured in two different resolutions 64×64 and 28×28. CAE-ELM handles and trains all the student's images captured using varying resolutions. CAE-ELM model tests the captured image and marks the student's presence in the lab. Table 4 shows the different labs attended by different batch of students.

We implemented the application scenario with three datasets: ORL, JAFFE, and FERET. ORL dataset comprises of 440 (92×112) images, which includes 40 classes. JAFFE dataset comprises of 10 classes, with a total of 200 (256×256) images. FERET includes 11,338 (512×768) images with 994 classes. 10 classes from each dataset were chosen. In batch B1, classes 0, 1, 2 were trained with resolution 64×64 from Python lab. In batch B2, classes 3, 4, 5 were trained with resolution 32×32 from MATLAB lab. Similarly, in batch B3, classes 6, 7, 8, 9 were trained with resolution 28×28 from Java lab. In each dataset, the images were resized to specific resolutions to implement the scenario. Table 5 illustrates the incremental results obtained for ORL, JAFFE, and FERET datasets respectively. Individual testing involves samples only from trained classes in the specific batches. Incremental Testing holds samples from all the classes (0, 1, 2, …, 8, 9).

Table 4. Scenario for incremental learning using CAE-ELM

Batch	Classes	Labs			
		Python(64x64)	R (39x39)	MATLAB (32x32)	Java (28x28)
B1	0-9			✓	✓
B2	10-19	✓			✓
B3	20-29	✓	✓		
B4	30-39		✓	✓	
B5	40-79		✓		✓

Table 5. CAE-ELM incremental results for ORL, JAFFE and FERET dataset

Dataset	Batch	Classes	Individual testing			Incremental testing
			Python (64x64)	R (39x39)	MATLAB (32x32)	
ORL	B1	0, 1, 2	87.6	-	-	35.8
	B2	3, 4, 5	-	100	-	57.2
	B3	6, 7, 8, 9	-	-	95.8	96.5
JAFFE	B1	0, 1, 2	100	-	-	33.3
	B2	3, 4, 5	-	100	-	64.2
	B3	6, 7, 8, 9	-	-	87.9	94.7
FERET	B1	0, 1, 2	95	-	-	32.1
	B2	3, 4, 5	-	98	-	60.23
	B3	6, 7, 8, 9	-	-	96	97.5

3.3. Discussions

3.3.1. Pros and cons of CAE-ELM

The advantages of CAE-ELM lie mainly in addressing the varied resolution images arriving in different batches and creating a single updated model for them without forgetting the learned knowledge. The data augmentation method helps us in preserving the learned information of every batch, and propagates it to the other upcoming batches. The use of CNN and ELM enhances the efficiency of the proposed work by extracting the best features and providing a faster training time. CAE-ELM stays efficient in i) memory by discarding all the old models and data and ii) time by avoiding the retraining of old samples.

ELM stands superior to other conventional methods like CNN with its fast-training time. The advantage of using ELM in CAE-ELM lies with the replacement of the backpropagation part in CNN for training purpose. With no doubt, ELM is a better choice than the conventional back-propagation process. ELM lacks efficiency with the feature extraction process, specifically for the red, green, and blue (RGB) images. In such cases, definitely CNN is a better choice over ELM. So, we have used CNN for the feature extraction process. So, the combined use of CNN and ELM in CAE-ELM definitely proves to show a significant improvement in the classification accuracy.

However, when does the performance of CAE-ELM becomes a concern? CAE-ELM becomes time-consuming when the varied resolution images are very high dimensional i.e., 2048×1536, and 7680×6876. Running the CNN for feature extraction followed by the stacked autoencoder to convert into the same feature sizes may be time-consuming. Though the performance accuracy might be achieved, the time consumed in running CNN and SAE together will wash away the advantage of avoiding retraining samples. In such cases, some other efficient techniques must be used to handle the high-dimensional varied resolution images. However, in very high-dimensional images, the use of ELM in place of back-propagation for image classification compensates the time spent with SAE to some extent.

3.3.2. Pros of ELM training over the use of back-propagation

The CNN extracts the spatial features of images with the use of convolutional filters. After feature extraction and reduction, the ELM classifier replaces the backpropagation for training the datasets. ELM does not undergo any iterations for finding trained weights, thus requiring less time to train a model, even in the case of a large number of hidden neurons. ELM is also resistant to overfitting, except in the case of tiny datasets. As the training sample increases, ELM provides good performance accuracy.

Using the advantages of CNN and ELM, the proposed work CAE-ELM overcomes the memory requirement of a large dataset by training each batch of arriving data independently to create individual models, which avoids the retraining of previous batch data. CAE-ELM works better than the existing algorithm Learn++ and ELM++ for the following reasons. i) CAE-ELM uses the best feature extractor CNN, whereas ELM++ extracts the minute features of the image and ii) The training time decreases in CAE-ELM. It uses a feedforward ELM network that does not involve any iterations, whereas Learn++ uses multiple neural network classifiers, which involve iterations for backpropagating the errors.

3.3.3. Challenges addressed

From the illustration shown in section 2.3, it is clear that CAE-ELM addresses the challenges of incremental learning. Irrespective of the classes added or removed in different batches occurring at different times, CAE-ELM adapts the model efficiently and learns all the classes, addressing the challenges of stability-plasticity dilemma and catastrophic forgetting. Irrespective of the varying image features and classes, CAE-ELM adapts the model dynamically and never forgets previously learned classes.

4. CONCLUSION

In this paper, our proposed incremental algorithm CAE-ELM learns varied resolution images arriving in different batches efficiently, both in terms of memory and time, using the ELM neural network. Except for the very high-dimensional images, the use of Stacked Autoencoder helps the incremental model to accommodate information about varied resolution images. Addition/deletion of new/old classes to the forthcoming batches never affects the learning performance of CAE-ELM. The most recently updated model is only retained with all previously learned information, discarding all other trained models and data, saving memory space. No input sample is retrained to save training time. Instead, CAE-ELM uses the correctly classified test samples as an augmented input source to achieve an efficient incremental model. The current incremental version of CAE-ELM works perfectly for varying image datasets. Our future work is to design an incremental algorithm for time series data with concept drift.

APPENDIX

Table 1. Strategies adopted for incremental learning in NN

S. No	Paper title (Author)	NN architecture used	Strategy adopted
1	End-to-end incremental learning (Francisco *et al.* [24])	Deep CNN	1. exemplars + Generating synthetic data 2. regularization using the cross-entropy and distilling loss functions
2	Incremental learning for classification of unstructured data using extreme learning machine (Madhusudhanan *et al.* [25])	ELM	Using exemplars
3	Brain-inspired replay for continual learning with artificial neural networks (Van *et al.* [26])	Variational auto-encoders	Regenerates old samples
4	Incremental learning in online scenario (He *et al.* [27])	CNN	Regularization using cross-distillation loss function
5	An incremental extreme learning machine for online sequential learning problems (Guo *et al.* [28])	ELM	Parameter-isolation method using three alternatives-minimal-norm incremental ELM (MN-IELM), least square incremental ELM (LS-IELM), kernel-based incremental (KB-IELM)
6	Online sequential extreme learning machine (Huang *et al.* [29])	ELM	Parameter-isolation using recursive least-squares algorithm
7	Error-minimized extreme learning machine (Feng *et al.* [30])	ELM	Parameter-isolation by adding random hidden nodes
8	Learn++: an incremental learning algorithm for supervised neural networks (Polikar *et al.* [31])	MLP	Updating of weights
9	Incremental learning from stream data Haibo (He *et al.* [32])	MLP	Updating of weights
10	Convolutional auto-encoded extreme learning machine (CAE-ELM)	CNN-SAE-ELM	Generating synthetic data

REFERENCES

[1] R. Kashef and M. Warraich, "Homogeneous Vs. heterogeneous distributed data clustering: A taxonomy," in *Data Management and Analysis*, 2020, pp. 51–66, doi: 10.1007/978-3-030-32587-9_4.
[2] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Symposium on Artificial Neural Networks (ESANN)*, 2016, pp. 27–29.
[3] R. R. Ade and P. R. Deshmukh, "Methods for incremental learning: A survey," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 4, pp. 119–125, Jul. 2013, doi: 10.5121/ijdkp.2013.3408.
[4] Y. Kim and C. H. Park, "An efficient concept drift detection method for streaming data under limited labeling," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 10, pp. 2537–2546, 2017, doi: 10.1587/transinf.2017EDP7091.
[5] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in Psychology*, vol. 4, no. 504, 2013, doi: 10.3389/fpsyg.2013.00504.
[6] S. Sukhov, M. Leontev, A. Miheev, and K. Sviatov, "Prevention of catastrophic interference and imposing active forgetting with generative methods," *Neurocomputing*, vol. 400, pp. 73–85, Aug. 2020, doi: 10.1016/j.neucom.2020.03.024.

[7] L. C. Jain, M. Seera, C. P. Lim, and P. Balasubramaniam, "A review of online learning in supervised neural networks," *Neural Computing and Applications*, vol. 25, no. 3–4, pp. 491–509, Sep. 2014, doi: 10.1007/s00521-013-1534-4.
[8] J. Zhong, Z. Liu, Y. Zeng, L. Cui, and Z. Ji, "A survey on incremental learning," in *5th International Conference on Computer, Automation and Power Electronics*, 2017, vol. 1, doi: 10.25236/cape.2017.034.
[9] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network- a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679–688, 2018, doi: 10.1016/j.procs.2018.05.069.
[10] R. Katuwal and P. N. Suganthan, "Stacked autoencoder based deep random vector functional link neural network for classification," *Applied Soft Computing*, vol. 85, Dec. 2019, doi: 10.1016/j.asoc.2019.105854.
[11] C. Chen, K. Li, M. Duan, and K. Li, "Extreme learning machine and its applications in big data processing," in *Big Data Analytics for Sensor-Network Collected Intelligence*, Elsevier, 2017, pp. 117–150, doi: 10.1016/B978-0-12-809393-1.00006-4.
[12] Q. Zhang, H. Li, C. Liu, and W. Hu, "A new extreme learning machine optimized by firefly algorithm," in *2013 Sixth International Symposium on Computational Intelligence and Design*, Oct. 2013, pp. 133–136, doi: 10.1109/ISCID.2013.147.
[13] S. Song, M. Wang, and Y. Lin, "An improved algorithm for incremental extreme learning machine," *Systems Science & Control Engineering*, vol. 8, no. 1, pp. 308–317, Jan. 2020, doi: 10.1080/21642583.2020.1759156.
[14] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, Jul. 2006, doi: 10.1109/TNN.2006.875977.
[15] J. Wang, S. Lu, S.-H. Wang, and Y.-D. Zhang, "A review on extreme learning machine," *Multimedia Tools and Applications*, vol. 81, no. 29, pp. 41611–41660, Dec. 2022, doi: 10.1007/s11042-021-11007-7.
[16] M. Yousefi-Azar and M. D. McDonnell, "Semi-supervised convolutional extreme learning machine," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1968–1974, doi: 10.1109/IJCNN.2017.7966092.
[17] L. Zhang, Z. He, and Y. Liu, "Deep object recognition across domains based on adaptive extreme learning machine," *Neurocomputing*, vol. 239, pp. 194–203, May 2017, doi: 10.1016/j.neucom.2017.02.016.
[18] L. Zhang, D. Zhang, and F. Tian, "SVM and ELM: Who wins? object recognition with deep convolutional features from ImageNet," in *Proceedings of ELM-2015*, 2016, pp. 249–263, doi: 10.1007/978-3-319-28397-5_20.
[19] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: a review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, Jan. 2018, doi: 10.1016/j.neucom.2017.06.084.
[20] J. F. García Molina, L. Zheng, M. Sertdemir, D. J. Dinter, S. Schönberg, and M. Rädle, "Incremental learning with SVM for multimodal classification of prostatic adenocarcinoma," *PLoS ONE*, vol. 9, no. 4, Apr. 2014, doi: 10.1371/journal.pone.0093600.
[21] Y. Xu, F. Shen, and J. Zhao, "An incremental learning vector quantization algorithm for pattern classification," *Neural Computing and Applications*, vol. 21, no. 6, pp. 1205–1215, Sep. 2012, doi: 10.1007/s00521-010-0511-4.
[22] P. Kang, "One-class naïve Bayesian classifier for toll fraud detection," *IEICE Transactions on Information and Systems*, vol. E97.D, no. 5, pp. 1353–1357, 2014, doi: 10.1587/transinf.E97.D.1353.
[23] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Twenty-first international conference on Machine learning - ICML '04*, 2004, doi: 10.1145/1015330.1015332.
[24] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," *Prepr. arXiv.1807.09536*, Jul. 2018.
[25] S. Madhusudhanan, S. Jaganathan, and J. L S, "Incremental learning for classification of unstructured data using extreme learning machine," *Algorithms*, vol. 11, no. 10, Oct. 2018, doi: 10.3390/a11100158.
[26] 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, no. 1, Aug. 2020, doi: 10.1038/s41467-020-17866-2.
[27] J. He, R. Mao, Z. Shao, and F. Zhu, "Incremental learning in online scenario," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 13923–13932, doi: 10.1109/CVPR42600.2020.01394.
[28] L. Guo, J. Hao, and M. Liu, "An incremental extreme learning machine for online sequential learning problems," *Neurocomputing*, vol. 128, pp. 50–58, Mar. 2014, doi: 10.1016/j.neucom.2013.03.055.
[29] G. Bin Huang, N. Y. Liang, H. J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine," in *Proceedings of the IASTED International Conference on Computational Intelligence*, 2005, vol. 2005, pp. 232–237.
[30] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352–1357, Aug. 2009, doi: 10.1109/TNN.2009.2024147.
[31] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 4, pp. 497–508, 2001, doi: 10.1109/5326.983933.
[32] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011, doi: 10.1109/TNN.2011.2171713.
[33] P. Wang, X. Zhang, and Y. Hao, "A method combining CNN and ELM for feature extraction and classification of SAR image," *Journal of Sensors*, vol. 2019, pp. 1–8, Nov. 2019, doi: 10.1155/2019/6134610.

BIOGRAPHIES OF AUTHORS

Sathya Madhusudanan research scholar at Anna University pursuing her research in deep learning. She did her master's in computer science and engineering and obtained a bachelor's degree from Anna University. To her credit, she has published papers in reputed International Conferences and Journals and also filed one patent. Her areas of interest are data analytics and deep learning. She can be contacted at email: sathyamadhusudhanan@gmail.com.

Suresh Jaganathan 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 referred International Journals and Conferences. Apart from this to his credit, he has two patents in the area of image processing and has 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 center. His areas of interest are distributed computing, deep learning, data analytics, machine learning and blockchain technology. He can be contacted at email: sureshj@ssn.edu.in

Dattuluri Venkatavara Prasad Professor in the Department of Computer Science and Engineering and has more than 20 years of teaching and research experience. He received his B.E degree from the University of Mysore, M.E. from the Andhra University, Visakhapatnam, and PhD from the Jawaharlal Nehru Technological University Anantapur. His PhD work is on “Chip area minimization using interconnect length optimization”. His area of research is computer architecture, GPU computing and machine learning. He is a member of IEEE, ACM and also a Life member of CSI and ISTE. He has published a number of research papers in International and National Journals and conferences. He has published two textbooks: Computer Architecture, and Microprocessors\Microcontrollers. He is a principle Investigator for SSN-nVIDIA GPU Education/Research Center. He can be contacted at dvvprasad@ssn.edu.in.