Continual Learning with Self-Organizing Maps: A Novel Group-Based Unsupervised Sequential Training Approach

Gaurav Hirani[1]*, Kevin I-Kai Wang[2] and Waleed Abdulla[3]
[123]The University of Auckland, Auckland, New Zealand
[1]E-mail: ghir325@aucklanduni.ac.nz Tel: +64 22 307 2932
[2]E-mail: kevin.wang@auckland.ac.nz Tel: +64 (9) 373 7599 Ext.89226
[3]E-mail: w.abdulla@auckland.ac.nz Tel: +64 (9) 373 7599 Ext.88969

Abstract – Continual learning is essential for intelligent systems to thrive. While deep learning models based on Convolutional Neural Networks are robust, they can be complex and resource-intensive, requiring a long and computationally intensive training phase due to Back-propagation (BP). On the other hand, unsupervised methods are favored for their lightweight, speed, and ease of management, making them ideal for ongoing learning. Self-organizing maps (SOMs) have gained attention in computer vision over the past decade due to their excellent feature extraction abilities and transparency. However, SOMs encounter challenges in learning, such as having a fixed map size that limits new feature accommodation and the risk of interference when retraining a pre-trained map with the latest data. Although some studies have explored using SOMs to tackle these issues, obstacles like catastrophic forgetting and lifelong learning hurdles remain prevalent. To combat forgetting issues, we propose an approach called learning by grouping. In this method, each group generates a trained map stored in memory before the model is reset and trained on the group to create another map. This sequential process facilitates learning while preserving previously acquired knowledge without disruptions. The suggested approach has been tried out on MNIST digit, Fashion MNIST, and CIFAR 10 datasets. The proposed method demonstrates final accuracy improvement of 60-70% across all three datasets when compared to traditional SOM, indicating the effectiveness of grouping in overcoming the limitations of SOM-based continual learning methods.

I. Introduction

Convolutional Neural Network (CNN)-based deep architectures have achieved significant success in the field of computer vision, demonstrating exceptional efficiency across various applications[1] [2]. These deep learning (DL) models are typically trained on labeled data for all available classes or target groups. When new data from a previously unseen class is introduced, the existing trained model cannot adapt without requiring a complete retraining of the entire architecture. The capability to continuously learn over time by incorporating new knowledge while retaining previously acquired experiences is known as continual or lifelong learning [3]. Unlike humans, who seamlessly integrate new information while preserving old knowledge, traditional algorithms face challenges in this area. The complexity and large size of DL architectures result in substantial time and computational power requirements for retraining.

To address the challenge of making deep architectures adaptive, significant approaches have been proposed for continual learning (CL) [4]. Models designed with CL functionality to prevent forgetting old tasks are primarily categorized into three types: retraining with regularization, training with network expansion, and selective network retraining and expansion. [5] [6]. Introducing new sub-networks prevent catastrophic forgetting, the parameters learned for existing tasks remain unchanged, while new parameters are added and trained [7]. Combining both concepts in a balanced manner constitutes the third category of architecture. Additional categories include replay-based approaches, optimization-based approaches, representation-based approaches, and memory-based methods [4] [8]. These advancements have predominantly been applied to supervised models, particularly deep neural networks. DL models often suffer from limited tractability, making their outcomes difficult to interpret, which poses a significant bottleneck to major improvements. To transparently study the effects of adaptive learning, a few studies have focused on unsupervised approaches. Although these models are not as efficient as their supervised counterparts for large datasets, they are lightweight, tractable, and faster.

Self-organizing maps (SOM) are an unsupervised clustering technique widely used for dimensionality reduction, feature extraction, and clustering in image datasets. Several key studies have attempted to implement SOM for continual learning. Two SOM-based models were proposed for incremental learning: (i) a modified self-organizing map (GeppNet) and (ii) a SOM extended with a short-term memory (GeppNet+STM) [3][9]. In GeppNet, task-relevant feedback from a regression layer determines whether learning in the self-organizing hidden layer should occur. In GeppNet+STM, the STM stores new knowledge and occasionally replays it to the GeppNet layer during sleep phases interleaved with training phases. This latter approach demonstrated better performance and faster convergence in incremental learning tasks with the MNIST dataset. However, the STM's limited capacity means that new knowledge can overwrite old information [10]. To address the significant memory requirements, a SOM-based memory-less architecture (SOMLP) was proposed, which combines standard

supervised neural networks with SOM to tackle the continual learning problem [11]. Another approach, Dendritic SOM (DendSOM), was proposed as a single-layer SOM for feature extraction using a set of hit-matrices. The input image was divided into patches and projected onto independent SOMs, each accompanied by hit-matrices to associate labels with the units. However, the formation of hit-matrices remains opaque [12]. Additionally, a memory-based model called continual SOM (CSOM) was introduced to generalize the standard SOM with minimal memory allocation. The structural design of DendSOM was adopted to manage the spatial dimension of the data [13]. Despite these advancements, all these models suffer from the stability–plasticity dilemma, a well-known issue in lifelong learning aimed at preventing catastrophic forgetting in computational models[14] [15].

Summarizing the research on Self-Organizing Maps (SOM), it is evident that SOM offers greater tractability in the learning process with new information. However, the introduction of additional classes necessitates excessively large maps for accommodation, leading to computational inefficiency and reduced feasibility. For self-organizing networks, catastrophic interference escalates rapidly with oversized maps, leading to performance degradation. Even stringent predefined map size limitations hinder the integration of new data eventually. Irrespective of map size, retraining exacerbates forgetting, as new features overwrite existing ones. To address these issues, this study proposes handling data in groups and processing them sequentially, "storing" previously learned maps before using the model for the next group. The architecture is a memory-based model that produces a bank of trained map weights as the outcome. The core contributions of the study are outlined below.

1. The proposed model handles new data by creating a new group and producing a new map without updating or interfering with the old maps. This approach eliminates the issue of forgetting, as no new information overwrites the old ones. As each map only represent fewer class, larger map size is not required.
2. The testing process only utilizes trained weight and labeled array to predict the label which makes the SOM available to get trained on new data with a new class independently and simultaneously.
3. The model offers the flexibility to be retrained on new data for which it has already been trained without interfering with any other group. This is achieved by using the current trained weights as filter initialization and performing training on the new data.

The rest of the paper is structured as follows: Section II explains the proposed methods including training, labelling and testing phases. Experimental details and results are provided in section III. Section IV has discussion and comparison of the results to the close studies. The paper is concluded with conclusion and future work in section V.

II. PROPOSED MODEL

The proposed method utilizes the datasets into groups and the concept is explained in the first subsection. The following subsections describes training of the SOM, labelling and testing.

A. Concept of Grouping

Traditionally, the training of nearly all machine learning models involves using the entire training dataset, including SOM. However, when the complete training dataset encompassing all labels is not initially available and new data associated with new classes is introduced over time, recent studies on SOM for continual learning employ various strategies to minimize the impact on previously trained weights. The challenge with SOM lies in its constrained output space: the map size is fixed initially, and adding more nodes to a 2D structure in terms of columns and rows can distort cluster shapes. New clusters for new data cannot simply be linear, requiring them to occupy space previously learned by other samples. The SOM's neighborhood learning attempts to symmetrically shape clusters, but whether the map size is increased or not, the process of retraining with new data inevitably affects existing knowledge. As more data arrives, the phenomenon of catastrophic forgetting becomes unavoidable. Using an excessively large map from the outset also fails to resolve the problem, as it would exponentially increase training time, and at some point, it would hit the bottleneck of accommodating more classes. Also, there is no surety of preventing interference between previously learned nodes and new ones. The issue of uncontrollable learning in SOM stems from the absence of a clear objective function for managing map updates.

To address the issue of forgetting, we propose a novel approach that involves processing the initially available data in groups rather than as a single set. These groups consist of samples belonging to specific labels, although the labels themselves are not used during training to maintain an unsupervised approach. Initially, the labels are only employed to partition the large dataset into these groups and can be considered as data-preprocessing. Fig. 1 illustrates the overall structure of our proposed method for handling data over time. Each group includes datasets associated with two different labels. Number of classes per group is a hyperparameter, and chosen as two to facilitate comparison with previous studies.

During the training phase, labels are not employed. As depicted in Fig. 1, after training each group, the corresponding map is generated and stored in a memory bank. Subsequently, the weights are re-initialized for the next group, and a new map is generated and stored in the same manner. This process iterates through all the groups. Once training is completed, the labeling process utilizes these trained maps alongside labels to create labeled arrays. In the testing phase, predictions are made using the trained maps, labeled arrays, and test samples with their actual labels, allowing comparison between predicted and actual labels. Detailed explanations of the training, labeling, and testing phases follow in subsequent subsections.

B. Training of the SOM

The SOM training follows the original method introduced by Kohonen [16]. As explained in previous subsection, the input dataset is divided into groups. Each group sequentially trains the SOM according to Algorithm 1 without any labeled information. As illustrated in Fig. 2, the Euclidean distance is computed from

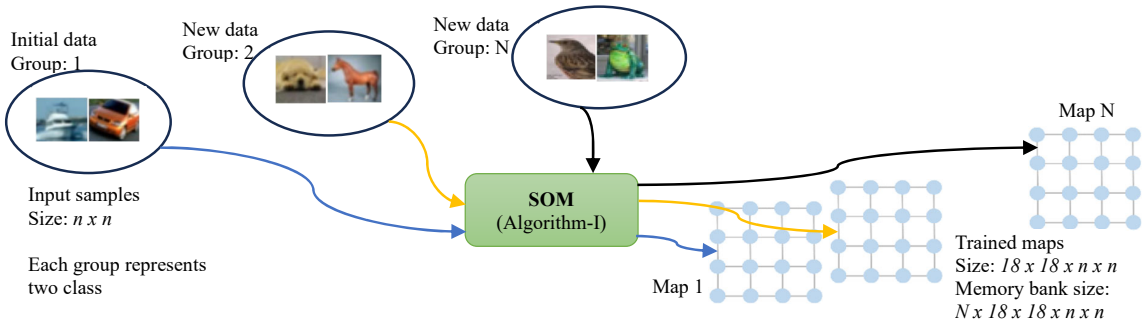

Fig. 1. The overview of the proposed structure. Introduction of new data in groups and generation of trained weights to store as learnt maps.

each image to all nodes within the assigned map. The node with the shortest distance is identified as the Best Matching Unit (BMU), representing the node most closely resembling the image in terms of data distribution or pattern similarity (lines 3-7, Algorithm I).

Once the BMU is determined, the next step involves updating the weights to move the BMU closer to the input image. Simultaneously, the weights of neighboring nodes to the BMU are adjusted based on a neighborhood function (lines 8-10, Algorithm I). $N(t)$ defines the extent of the neighborhood, which decreases over epochs; roughly halfway through training, only the BMU undergoes weight updates.

The training concludes upon reaching the maximum number of epochs. The main hyperparameters for SOM are map size, maximum number of Epochs (t), which defines the training cycles, learning rate $\eta(0)$ and initial neighborhood width or radius $\delta(0)$. η and δ are linear and monotonically decreasing functions respectively (lines 13 and 14, Algorithm I). The Fig. 1 mimics training process as continual learning which generates map banks as outcome.

TABLE I
TRAINING OF SOM

	Algorithm I. SOM Training
	Inputs: A batch of training dataset (No labels) *Output:* Trained SOMs
1:	*Weight Initialization:* Random [0,1]
2:	**for** t in epochs
3:	**for** s in train samples per group //find the BMU
4:	**for** i in nodes in SOM
5:	calculate $d(s, w_i)$ //$d(s, w_i) = \|s - w_i\|^2$
6:	**end**
7:	$\arg\min_i d(s, w_i)$ → BMU // n_{BMU} // weight update
8:	**for** i in nodes in SOM
9:	Compute the neighborhood function $N(t)$ // $N(t) = exp\left(-\frac{\|n_{BMU} - n\|^2}{2\delta(t)^2}\right)$
10:	Weight update $w_i(t+1)$ //$w_i(t+1) = w_i(t) + \eta(t)N(t)(s - w_i(t))$
11:	**end**
12:	**end**
13:	Update learning rate $\eta(t)$// $\eta(t) = 0.49\left(1 - \frac{t}{epoch}\right) + 0.01$
14:	Update neighborhood width $\delta(t) = \delta(0)\, exp^{\left(-\frac{t}{epoch}\right)}$
15:	**end**

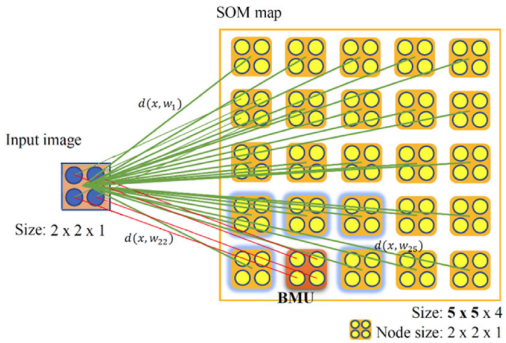

Fig. 2. BMU selection in SOM. For demonstration, node 22 is depicted as the BMU for the highlighted patch considering $d(x, w_{22})$ is the minimum distance. The neighborhood nodes of the BMU are highlighted with blue backlight.

C. *Labelling of the arrays*

The labeling process commences immediately after the training phase concludes, following the steps outlined in Algorithm II. The inputs to these sections include subsets of the labeled training dataset and the trained SOM maps. These inputs generate labeled arrays as outputs, where each array aligns spatially with the SOM map grid. Each element within the array functions as a unit storing a label represented by the node at the corresponding location on the SOM map.

TABLE II
LABELING OF THE ARRAYS

	Algorithm II. SOM Labeling
	Inputs: A subset (s) of the batch of the training dataset (with labels) and Trained SOMs *Output:* Arrays with label information
1:	The number of patches (=*SOMs)* defines the number of arrays.
2:	**for** s in a subset of train samples // Find the BMU with designated SOM(k)
3:	**for** n in group branches
4:	Index (BMU) → j
5:	Label[s] → array[n, j]
6:	**end**
7:	**end**

For each image, the Best Matching Unit (BMU) is identified using the trained map based on Euclidean distance. Once the BMU is determined, its index is recorded, and the corresponding label for that sample is assigned to the corresponding index in the respective array. Each group maintains its own map and associated array to represent the labels associated with that map.

Unlike during training, no further weight updates occur once the BMU is determined. Each sample is processed once, and this process repeats for all samples. As each group is sequentially trained over time, the memory bank accumulates trained maps along with their respective labeled arrays.

D. Testing and decision making

The model evaluation utilizes the test dataset, which includes labels. Each test image is projected onto all trained maps. We have implemented a multi-BMU technique, where instead of identifying just one closest distance, we determine the three closest distances per map. These distances are summed for each map, and the map with the smallest total distance determines the winning map and corresponding group. The three node indexes contributing to the decision of the winning group are recorded, and the labels associated with these indexes are extracted from the array belonging to the winning group. The mode of these three labels is then compared against the actual label to assess the model's performance. Since each map and array represent only two classes, the mode operation consistently identifies a clear winner.

III. EXPERIMENTS

This section addresses the datasets utilized in the experiments, the hyperparameters selection and evaluation metrices. The algorithm implementation was carried out in Python 3.8.8, utilizing the MiniSOM library for SOM calculations within the Keras framework [17]. The experiment was conducted on a Core i7-10700 processor without a GPU acceleration.

A. Datasets

The experiments utilized three datasets: MNIST-digit [18], Fashion-MNIST [19] and CIFAR-10 [20]. The data characteristics are detailed in Table III. MNIST-digit and Fashion-MNIST datasets consist of 8-bit grayscale images depicting digits and clothing items, respectively. Both datasets have comparable sizes for their training and testing subsets, with images uniformly sized at 28 x 28 pixels. In contrast, CIFAR-10 is a more complex dataset containing 24-bit color images representing real-world objects. Each image in CIFAR-10 measures 32 x 32 pixels, with objects randomly positioned within the spatial domain.

As part of the data preparation, the dataset undergoes normalization. Furthermore, the conversion of RGB to grayscale is achieved by assigning equal weight to each color channel. No additional data processing steps are applied. Training exclusively utilizes the clean training dataset.

For the colored datasets like CIFAR-10, using only the blue channel for both training and testing results in a performance enhancement of 2 - 2.5% compared to using grayscale or other channels. This finding was incorporated into subsequent experiments on these datasets, where all operations were conducted exclusively using the blue channels.

B. Hyperparameters selection

The experiments primarily involve two types of hyperparameters: 1) Key SOM parameters, as detailed in section II, and 2) parameters related to data selection. The SOM parameters used in the current experiments are specified in Table IV. As the study does not focus extensively on fine-tuning hyperparameters and their impact on model performance, they undergo moderate tuning and are selected following a few trial-and-error iterations.

C. Performance metrics

1. **Accuracy:** The model's performance is assessed as groups are sequentially added to the model, with each addition considered a task. Initially, the model begins with only task 1, where it is trained using the first group of data. Adding subsequent groups results in tasks labeled as Nth, corresponding to the Nth group. Two types of performance metrics are observed:
   1) *Task-specific accuracy*: it evaluates performance using only the test dataset from the current task against all previously learned maps [12],

$$Accuracy_t = \frac{1}{N_t} \sum_{n=1}^{N_t} AL_n \cap PL_n \quad (1)$$

Where, N_t is the size of the test dataset per task, AL_n is actual label, PL_n is the predicted label over all previous learnt maps.

2) *Average accuracy:* The performance on task j after applying it on all previous task n_j is denoted as $a_{n_j,j}$. The average accuracy at task T after testing it on all previous tasks from 1 to T-1 is as follows [21]:

$$Average\ Accuracy_T = \frac{1}{T} \sum_{j=1}^{T} a_{n_j,j} \quad (2)$$

2. **Average Interference:** It is defined as the average decrease in performance for each task from its highest accuracy to the accuracy attained at the end of training, computed as follows: [21]:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} f_T^j \quad (3)$$

Where, f_T^j is forgetting term for task j. after model is applied on task T and is defined as follows:

$$f_T^j = \max_{\tau=[1,2,\dots T]} \left(a_{\tau,j} - a_{T,j}\right) \quad (4)$$

TABLE III
DATASET STATISTICS

Dataset	Train dataset	Classes	Images per class	Test dataset	Size of a sample	Used train dataset for training	Used train dataset for labeling	Used test dataset for testing
MNIST-digit	60000	10	6000	10000	28 x 28	4000	4000	10000
Fashion-MNIST	60000	10	6000	10000	28 x 28	4000	4000	10000
CIFAR-10 (RGB)	50000	10	5000	10000	32 x 32 x 3	7000	7000	10000

TABLE IV
HYPERPARAMETERS FOR THE PROPOSED STRUCTURE

Parameter	Value
Class per group	2
Number of SOM per dataset	5
SOM Map Size	15 x 15 x 28 x 28
	15 x 15 x 32 x 32*
Array size	15 x 15
Iteration	27000
Neighborhood radius	1
Learning rate	0.5

*Denotes the modification of hyperparameter for CIFAR-10

IV. RESULTS AND DISCUSSION

The model was applied to all three datasets using the hyperparameters outlined in Table IV. The accuracy for the task-specific dataset (Equation 1) and the overall average accuracy across all test datasets up to a given task (Equation 2) were recorded at each task, as detailed in Table V. With each task, a group representing two classes is added to the model, increasing the size of the memory bank by one map and one array. During the decision-making process at each task, all maps learned thus far, including the current one, are considered to determine the winning group.

A low task-specific accuracy at a particular task indicates that the group containing two classes has poorly learned a map, which does not emerge as the winner in the decision-making process. For instance, Group 5 for MNIST-digit, Group 4 for Fashion-MNIST, and Group 3 for CIFAR-10 were identified as the worst performers individually.

Typically, forgetting occurs when new features learned from new data overwrite or modify previously learned information. Previous SOM-based studies have employed similar maps to accommodate new knowledge or have retrained models using previously learned maps as initial weight states. However, in our approach, when new data is introduced, learning results in the creation of a new map while leaving old maps untouched. Consequently, there is technically no forgetting or interference during the training process itself. However, during the decision-making phase, all trained maps must be considered to identify the most closely related map to the sample. If multiple maps representing classes closely resemble each other, an incorrect group might be selected as the winner, which fits the definition of "interference." This can lead to a decline in accuracy at each task. The performance based average interference or forgetting

TABLE V
TASK SPECIFIC ACCURACY OVER PREVIOUSLY LEARNT TASKS

	MNIST-digit	Fashion-MNIST	CIFAR-10
Task 1	100%	97.44%	75.05%
Task 2	97.55%	92%	47.77%
Task 3	98.11%	91.16%	25.22%
Task 4	96.56%	73.16%	35.72%
Task 5	92.4%	91.33%	37.11%

AVERAGE ACCURACY OVER PREVIOUSLY LEARNT TASKS

	MNIST-digit	Fashion-MNIST	CIFAR-10
Task 1	100%	97.44%	75.05%
Task 2	98.78%	94.72%	60.16%
Task 3	98.55%	93.53%	48.51%
Task 4	98.06%	83.44%	45.32%
Task 5	96.92%	84.14%	43.67%

TABLE VI
AVERAGE INTERFERENCE AT EACH TASK

	MNIST-digit	Fashion-MNIST	CIFAR-10
Task 1	0.00%	0.00%	0.00%
Task 2	0.08%	3.68%	7.9%
Task 3	1.91%	12.22%	13.25%
Task 4	2.07%	19.48%	36.38%
Task 5	2.21%	18.19%	39.2%

(Equation 3) is recorded in Table VI. At the first task, there is only one map and no interference in selecting winner group.

The results obtained from our study were compared with previous SOM-based research. Fig. 3. displays the overall average accuracy at each task for MNIST-digit, with numbers extracted from relevant papers. Our proposed study achieves the highest accuracy throughout all tasks, including the final task encompassing all classes. Regarding Fashion-MNIST, we did not find specific results from the mentioned studies. However, results from previous studies on CIFAR-10 have been plotted alongside our proposed study in Fig. 4. DendSOM performs better for all tasks due to its effective patch-based learning for much diverse CIFAR-10. However, by the end of Task 5, the proposed method catches up with DendSOM, suggesting lower interference during incremental learning. It is worth noting that our proposed approach utilizes a single SOM model, whereas DendSOM employs 225 SOM models. Encompassing all classes, a single tradition SOM attains an accuracy of 58.95% for the MNIST-digit dataset, 52% for Fashion-MNIST, and 25.7% for CIFAR-10. In contrast, the proposed method achieves 96.92%, 84.14%, and 43.67% for these datasets, respectively. This indicates an improvement in final accuracy by grouping over the traditional SOM by 60-70%.

In summary, the proposed approach of learning classes in groups and storing learned maps in a memory bank effectively eliminates the limitation of catastrophic forgetting. By utilizing a single SOM that is reinitialized and trained with new data, the model achieves higher efficiency and lower computational load compared to previous methods. This method also supports lifelong learning, where each new group of data adds a map and a labeled array to the memory bank for future use.

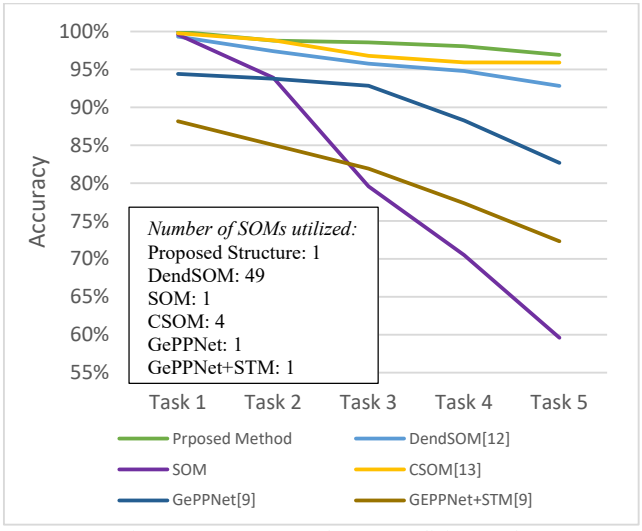

Fig. 3. Average accuracy for MNIST-digit per task

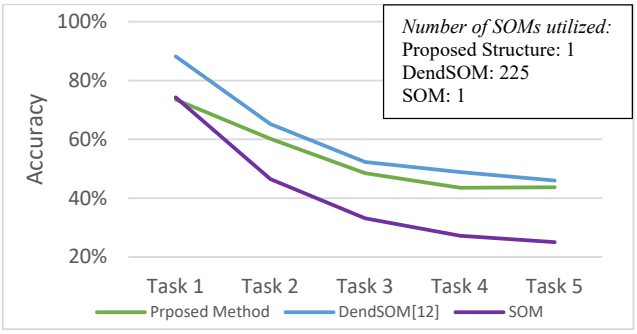

Fig. 4. Average accuracy for CIFAR-10 per task

For groups with additional available data, the SOM can be initialized with the existing trained weights and further trained with new data to enhance the map for that group. This is crucial if a group map performs poorly providing BMUs for own test dataset (e.g., Group 4 for Fashion-MNIST). The labeling and testing phases only require the trained weights and labels, not the entire model itself, which ensures independence between training and real-time testing operation parallelly.

An additional advantage is that labeled arrays can be optimized independently without modifying the weights, potentially improving accuracy. One limitation associated with the grouping concept is interference in winning group selection. As the number of maps increases, traditional distance measurement techniques become less effective in accurately identifying the correct group, which can result in incorrect predictions. To mitigate this issue, we intend to explore the implementation of additional metrics alongside Euclidean distance to determine the winning nodes.

V. Conclusion

This research presents a back-propagation free, unsupervised memory-based approach that utilizes a single Self-Organizing Map (SOM) as a tool for continuous learning. Unlike SOM-based approaches, this method offers several key benefits;

1. Prevention of Forgetting: Training individual maps helps prevent the issue of forgetting.

2. Separate Testing: This allows the model to learn from data while being used in real-time applications.

3. Efficiency: The model operates faster due to its design.

The model adjusts to data using multiple maps while keeping previously learned maps intact. Each new dataset creates a map without impacting existing ones. During testing, only the learned maps and labeled arrays are required to predict labels, enabling the SOM model to train on available data with fresh class labels simultaneously with testing phases.

This study confirms the effectiveness of group learning utilizing a Vanilla SOM, demonstrating substantially faster processing times compared to studies that employ multiple SOMs and various data processing modifications. Upon presenting all classes in the dataset, the proposed methods achieved accuracies of 96.92% for MNIST-digit, 84.14% for Fashion MNIST, and 43.67% for CIFAR-10, representing an improvement of nearly 60-70% over traditional SOM.

Future research will concentrate on improving class selection within each group and introducing methods to improve the correct group selections.

VI. Reference

[1] A. Khan *et al*, "A survey of the recent architectures of deep convolutional neural networks," *Artif Intell Rev,* vol. 53, *(8),* pp. 5455, 2020. DOI: 10.1007/s10462-020-09825-6.

[2] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access,* vol. 7, pp. 53040, 2019. DOI: 10.1109/access.2019.2912200

[3] G. I. Parisi *et al*, "Continual lifelong learning with neural networks: A review," *Neural Networks,* vol. 113, pp. 54, 2019. DOI: 10.1016/j.neunet.2019.01.012.

[4] L. Wang *et al*, "A Comprehensive Survey of Continual Learning: Theory, Method and Application," *IEEE Trans. Pattern Anal. Mach. Intell.,* pp. 1, 2024. DOI: 10.1109/tpami.2024.3367329.

[5] M. K. Benna and S. Fusi, "Computational principles of synaptic memory consolidation," *Nat Neurosci,* vol. 19, *(12),* pp. 1697, 2016. DOI: 10.1038/nn.4401.

[6] G. Hinton, O. Vinyals and J. Dean, "Distilling the Knowledge in a Neural Network," 2015.

[7] A. A. Rusu *et al*, "Progressive Neural Networks," *arXiv Preprint arXiv:1606.04671, 2016.*

[8] B. L. Üders, M. Schläger and S. Risi, "Continual Learning through Evolvable Neural Turing Machines," 2016.

[9] A. Gepperth and C. Karaoguz, "A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems," *Cogn Comput,* vol. 8, *(5),* pp. 924, 2016. DOI: 10.1007/s12559-016-9389-5.

[10] A. Gepperth and C. Karaoguz, "A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems," *Cogn Comput,* vol. 8, *(5),* pp. 924, 2016. DOI: 10.1007/s12559-016-9389-5.

[11] P. Bashivan *et al*, "Continual learning with self-organizing maps," *arXiv Preprint arXiv:1904.09330,* 2019.

[12] K. Pinitas, S. Chavlis and P. Poirazi, "Dendritic Self-Organizing Maps for Continual Learning," *arXiv Preprint arXiv:2110.13611,* 2021.

[13] H. Vaidya *et al*, "Neuro-mimetic Task-free Unsupervised Online Learning with Continual Self-Organizing Maps," *arXiv Preprint arXiv:2402.12465,* 2024.

[14] G. I. Parisi *et al*, "Lifelong learning of human actions with deep neural network self-organization," *Neural Networks,* vol. 96, pp. 137, 2017. DOI: 10.1016/j.neunet.2017.09.001.

[15] F. M. Richardson and M. S. C. Thomas, "Critical periods and catastrophic interference effects in the development of self-organizing feature maps," *Developmental Science,* vol. 11, *(3),* pp. 371, 2008. DOI: 10.1111/j.1467-7687.2008.00682.x.

[16] T. Kohonen, "The self-organizing map," *Proc IEEE,* vol. 78, *(9),* pp. 1464-1480, 1990.

[17] V. Giuseppe, "MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map," unpublished.

[18] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[19] H. Xiao, K. Rasul and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[20] Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009. Available: http https://www.cs.toronto.edu/~kriz/cifar.html

[21] P. Kumar and D. Toshniwal, "Lifelong learning gets better with MixUp and unsupervised continual representation," *Appl Intell,* vol. 54, *(7),* pp. 5235, 2024. DOI: 10.1007/s10489-024-05434-w