



Article

An Efficient Strategy for Catastrophic Forgetting Reduction in Incremental Learning

Huong-Giang Doan ¹, Hong-Quan Luong ², Thi-Oanh Ha ² and Thi Thanh Thuy Pham ^{3,*}¹ Faculty of Control and Automation, Electric Power University, Hanoi 11300, Vietnam; giangdth@epu.edu.vn² MQ Information and Communication Technology Solutions JSC, Hanoi 11700, Vietnam³ Faculty of Information Security, Academy of People Security, Hanoi 12100, Vietnam

* Correspondence: thanh-thuy.pham@mica.edu.vn

Abstract: Deep neural networks (DNNs) have made outstanding achievements in a wide variety of domains. For deep learning tasks, large enough datasets are required for training efficient DNN models. However, big datasets are not always available, and they are costly to build. Therefore, balanced solutions for DNN model efficiency and training data size have caught the attention of researchers recently. Transfer learning techniques are the most common for this. In transfer learning, a DNN model is pre-trained on a large enough dataset and then applied to a new task with modest data. This fine-tuning process yields another challenge, named catastrophic forgetting. However, it can be reduced using a reasonable strategy for data argumentation in incremental learning. In this paper, we propose an efficient solution for the random selection of samples from the old task to be incrementally stored for learning a sequence of new tasks. In addition, a loss combination strategy is also proposed for optimizing incremental learning. The proposed solutions are evaluated on standard datasets with two scenarios of incremental fine-tuning: (1) New Class (NC) dataset; (2) New Class and new Instance (NCI) dataset. The experimental results show that our proposed solution achieves outstanding results compared with other SOTA rehearsal methods, as well as traditional fine-tuning solutions, ranging from 1% to 16% in recognition accuracy.

Keywords: continual learning; memory reconstruction; loss combination; resnet backbone



Citation: Doan, H.-G.; Luong, H.-Q.; Ha, T.-O.; Pham, T.T.T. An Efficient Strategy for Catastrophic Forgetting Reduction in Incremental Learning. *Electronics* **2023**, *12*, 2265. <https://doi.org/10.3390/electronics12102265>

Academic Editor: Silvia Liberata Ullo

Received: 7 April 2023

Revised: 10 May 2023

Accepted: 11 May 2023

Published: 17 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Catastrophic forgetting or catastrophic interference is a serious problem in continuous learning in machine learning. It happens not only in traditional machine learning algorithms such as SVM (Support Vector Machine), NB (Naive Bayes), DT (Decision Tree), and CRF (Conditional Random Field) but also in DNNs. This phenomenon was firstly exposed in [1]. According to the observations in that work, when training on new tasks, the knowledge learned from previous tasks is forgotten by the machine learning model. The forgotten knowledge includes the weights learned from the source tasks that are overridden when learning the target tasks. The authors also demonstrated that the phenomenon of catastrophic forgetting is the main reason for degradation in the performance of the machine learning models. Since the announcement of [1] on catastrophic interference, several publications to address this challenge have emerged [2–8], and they shed more light on the causes of this phenomenon.

Considering traditional machine learning algorithms such as SVM (Support Vector Machine), NB (Naive Bayes), DT (Decision Tree), and CRF (Conditional Random Field), although they have been very successful in practice, they are inherently focused on single-task learning or isolated-task learning. Moreover, the models are fixed after deployment, or there is no learning after deployment. This causes limitations in leveraging the knowledge learned from these models to solve the new tasks or categories [1,3,4].

Recently, DNNs have achieved outstanding efficiency in different areas, but they require a large amount of training data [9–11]. In order for some networks to be effective,

it is frequently necessary to reuse model parameters that were learned from training a sizable dataset [12]. The pre-trained models are then fine-tuned on specialized data to obtain higher efficiency [13–15]. Transfer learning can cause the old knowledge to be overwritten by the new one. This makes it impossible for the model to remember the previous content. Therefore, transfer learning is considered the main reason for catastrophic forgetting in DNNs.

How to mitigate the catastrophic forgetting effect in DNNs has recently attracted great attention from the research community [16–18]. As a result, several continual learning solutions have been proposed for language problems [19,20], object segmentation and 3D reconstruction [21], and the recognition field [7,22–25]. The strategies proposed for mitigating the catastrophic forgetting of DNNs can be grouped into three main approaches: (1) rehearsal-based methods; (2) loss regularization-based methods; and (3) architecture search-based methods. In the first one, an explicit memory is utilized for maintaining (i) raw samples [22,26,27], (ii) pseudo-samples generated using a generator [28], or (iii) network representations/parameters stored from past tasks [29,30]. The previously learned knowledge is then adapted to the new task training, which helps to avoid forgetting the previous tasks. This approach can be applied to different types of continual learning settings, such as class incremental learning, task incremental learning, and domain incremental learning. However, it requires an efficient strategy on what to store and how to update the old knowledge into new tasks; otherwise, overfitting will occur. The second approach focuses on adding a regularization term to the loss function [7,29–31]. The loss, together with the penalty factor, determine the effectiveness of injecting knowledge from the source task model to the target task model. This itself is an attractive research area. The third approach pays attention to the significance of the neural network architecture in continual learning. Individual architecture selections such as the number of hidden layers, batch normalization, skip connections, pooling layers, etc. can affect the performance of the continual learning [32].

As shown in the above analysis, each approach has its own advantages and disadvantages for solving the problem of catastrophic forgetting. Combining the advantages of each with reasonable enhancements is necessary to reduce catastrophic interference in DNN networks. In this study, we propose a novel solution for continual learning without forgetting by exploring both approaches (1) and (2) with efficient improvements for each.

Firstly, a memory reconstruction strategy based on a random selection of samples from the source task to the target task for continual learning is proposed. This helps transfer the knowledge of the old task for the new task and reduce catastrophic forgetting in CNN models. Secondly, we propose an end-to-end incremental learning framework. It contains the baseline networks of Resnet18 [33] with the loss combination of the source training phase and the target phase. Based on this loss combination strategy, the learning process will be optimized at each step of incremental learning.

The proposed method is evaluated on five standard datasets. Three of them have been the most challenging datasets in the field of continual learning in recent years: CIFAR-10 and CIFAR-100 (<https://www.cs.toronto.edu/~kriz/cifar.html>, accessed on 12 January 2023) [34], and CORE-50 (<https://vlomonaco.github.io/core50/index.html#dataset>, accessed on 12 January 2023) [35]. The other two datasets are KinectLeap (https://lstm.dei.unipd.it/downloads/gesture/#kinect_leap, accessed on 3 February 2023) [36], and Creative Senz3D (<https://lstm.dei.unipd.it/downloads/gesture/#senz3d>, accessed on 3 February 2023) [37], which are commonly used in the field of hand gesture recognition. In order to evaluate the proposed solution on these datasets, two scenarios of incremental fine-tuning datasets are set: (1) the New Class (NC) dataset; (2) the New Class and new Instance (NCI) dataset. The NC scenario shows that the classes in the target dataset are different from the ones in the source dataset. However, the NCI scenario considers the case of intersection of some classes between the current and the previous memories. The experiments for the NC scenario are implemented on four above datasets, but only KinectLeap and Creative Senz3D are deployed for NCI. The experimental results show that our proposed solution

achieves outstanding results compared with other SOTA rehearsal methods, such as Lwf [7], iCaRL [22], OCM [25], and AOP [38], as well as the traditional fine-tuning solution, which ranges from 1% to 16% in terms of recognition accuracy.

The remainder of this paper is organized as follows: Section 2 first presents related work; Section 3 explains the proposed solution for incremental learning. Then, the experimental result and discussion are analyzed in Section 4. Finally, Section 5 concludes the paper and proposes research directions for future work.

2. Related Work

The catastrophic interference problem caused by transfer learning in DNNs has been recognized since about 30 years ago [1,3,39]. Currently, several solutions have been proposed to solve the problem of lifelong learning without forgetting [40–42]. In this research, we present some SOTA methods of continual learning in three brief surveys: (1) the proposals based on rehearsal mechanisms, (2) loss regularization-based solutions, and (3) incremental learning benchmark datasets.

2.1. Rehearsal-Based Methods

The methods of this approach try to retain a small subset of the source task to replay in the target task. This is a direct and effective way to lessen catastrophic forgetting through continual learning. In the work of [22], a training strategy named iCaRL is proposed for class-incremental learning. The classifiers and a feature representation are learned simultaneously from a class-incremental data stream. For classification, iCaRL relies on sequential sets of the fixed exemplar images that are dynamically selected from the data stream. All exemplars in each set belong to a certain class. An update routine algorithm is proposed to train the batches of classes at the same time in an incremental way. The iCaRL method uses this algorithm to adjust the exemplars and network parameters stored in the memory to be adapted to the new class set of training. Based on this, it can learn new tasks without forgetting the old tasks. The experimental results on the CIFAR-100 and ImageNet ILSVRC 2012 datasets proved the incremental learning efficiency of iCaRL in comparison with other methods.

In [26], a model of Gradient Episodic Memory (GEM) is proposed for continual learning. In GEM, a small amount of data per task is stored in an explicit memory, while the number of tasks is large. This is the opposite of other previous approaches to continual learning. The advantage of GEM is that it is able to constrain the training using real data. However, it is easily overfitted by the subset of stored samples because the replayed data are the only information that the model has for the previous tasks. An improved version of GEM called A-GEM was proposed in [27]. A-GEM focuses on balancing accuracy and efficiency in sample complexity, computational cost, and memory cost. In addition to the features inherited from GEM, a small change to the loss function was proposed to boost training faster while maintaining similar performance. In addition, cross validation is performed on a set of the disjointed tasks for evaluation. The experiments indicate that there is only a small gap in performance between the lifelong learning methods, including A-GEM. Forgetting in neural networks can be eliminated, but the transfer learning performance does not improve by much.

Different from the abovementioned solutions, the methods of [43,44] utilize greedy selection strategies for replay memory. However, these are inefficient because of the additional memory required for greedy saving. To overcome this, several optimal strategies for sample selection have been proposed. In [45], the authors proposed a memory retrieval technique that recovers memory samples with increased loss as a result of estimated parameter updates based on the current task. The solution in [46] scores samples in the memory based on their capacity to maintain latent decision boundaries for previously observed classes. The strategy in [47] expressly promotes samples from the same class to cluster closely in an embedding space while discouraging samples from other classes during replay-based training.

Instead of storing any old samples in replay memory as in the abovementioned methods, in [28], an infinite number of instances of old classes in the deep feature space are generated to overcome the classifier bias in class-incremental learning. In addition, the work in [48] uses orthogonal and low-dimensional vector subspaces for learning individual tasks. This helps to prevent catastrophic interference of learning target tasks from source tasks. In comparison with the rehearsal baseline methods with the same amount of memory, both accuracy and anti-forgetting ability are improved on deeper networks.

2.2. Loss Regularization-Based Solutions

The knowledge from the source task to the target task can be distilled via loss regularization in the training phase of the tasks. In [7], the authors proposed a solution called LwF (Learning without Forgetting), in which training is carried out on new task data while preserving the original capabilities. This overcomes the disadvantage of growing the number of new tasks with the increase in data that needs to be stored and retrained. The LwF method changes the network architecture and combines the sharing parameters of the trained model with the new model in the optimization function. However, the old model is frozen to retain the old knowledge and to optimize the extended network parameter. The LwF method outperforms the current standard practice of fine-tuning if we only care about the performance of the new task. In addition, it highly depends on the relevance between the new tasks and the old tasks, and the training time increases linearly with the number of the learned tasks.

The later proposed solutions for LwF showed more efficient improvements. In [30], the authors proposed two algorithms: Dark Experience Replay (DER) and DER++. DER looks for parameters that fit the current task well while approximating the behavior observed in the previous tasks. In order to retain knowledge of the previous tasks, the objective function implemented in DER is minimized by an optimization trajectory. DER++ feeds an additional term on buffer data points to the objective of DER to promote higher conditional likelihood to their ground truth labels with a minimal memory overhead. The work in [49] learns representations using the contrastive learning objective and reserves learned representations using a self-supervised distillation step. The experiments show that the proposed scheme performs better than baselines in various learning configurations. More efficient solutions for continual learning were recently published [25,38]. The work in [25] used all of the training data's attributes when learning each task. This helps lessen the feature bias brought on by cross entropy, which only learns features that are discriminative for the task at hand but may not be discriminative for another. The network parameters that were previously learned must be adjusted in order to learn a new task well. The promising technique in [38] named AOP (Adaptive Orthogonal Projection) only changes the network parameters in the direction that is orthogonal to the subspace bounded by all of the prior task inputs. As a result, there is no requirement to memorize the input data for each task, and the knowledge about the prior tasks is incrementally updated. According to empirical analysis, these methods significantly outperformed most recent continual learning baselines.

In this work, we deploy both rehearsal-based and loss regulation-based methods in novel settings for continual learning without forgetting. It is different from other rehearsal-based methods in which a small fixed amount of data per task is stored in the memory or greedy storage. In this work, a strategy of selecting samples at random from the previous memory is proposed for replay memory. Furthermore, these samples are then combined with the current random-chosen ones to form a fixed memory for the current task. This process is performed in a cumulative way through a sequence of tasks. This is a key improvement in our memory reconstruction strategy compared with other rehearsal-based methods. It helps efficiently eliminate the over-fitting phenomenon that normally occurs in other related methods.

Another key improvement of our framework in comparison with other solutions is a strategy for loss regularization. It is performed by optimizing the combination of training

losses from the previous task and the current task. This is promoted in the continuous learning framework, which contains the parallel ResNet models. Based on this, catastrophic forgetting will be efficiently avoided.

In comparison with the abovementioned methods at both approaches, our strategy has better performance on the standard datasets. This will be proved in detail in the Section 4.

2.3. Incremental Learning Benchmark Datasets

There are many continual learning benchmarks, as presented in [23], including MNIST [50], CIFAR-100 [34], ILSVRC2012 [12], CUB-200 [51], and CORE-50 [35]. These datasets consist of large categories, as well as the divergent instances, which are suitable for evaluating continual learning algorithms. Moreover, in the application aspect, using specification datasets (such as action, hand gesture, and face) is also necessary when a certain CNN model is pre-trained on a small dataset. It is then extended to recognize a new dataset that not only contains different instances but also has other categories. In this work, our experimental results are achieved on three continual learning benchmarks (CIFAR-10 and CIFAR-100 [34], and CORE-50 [35]) and two hand gesture datasets (KinectLeap [36] and Creative Senz3D [37]).

3. Proposed Method

The proposed framework for overcoming catastrophic forgetting in continual learning is shown in Figure 1. It has two components: (1) a memory reconstruction strategy and (2) end-to-end continual learning via parallel ResNet18 branches with a loss combination strategy.

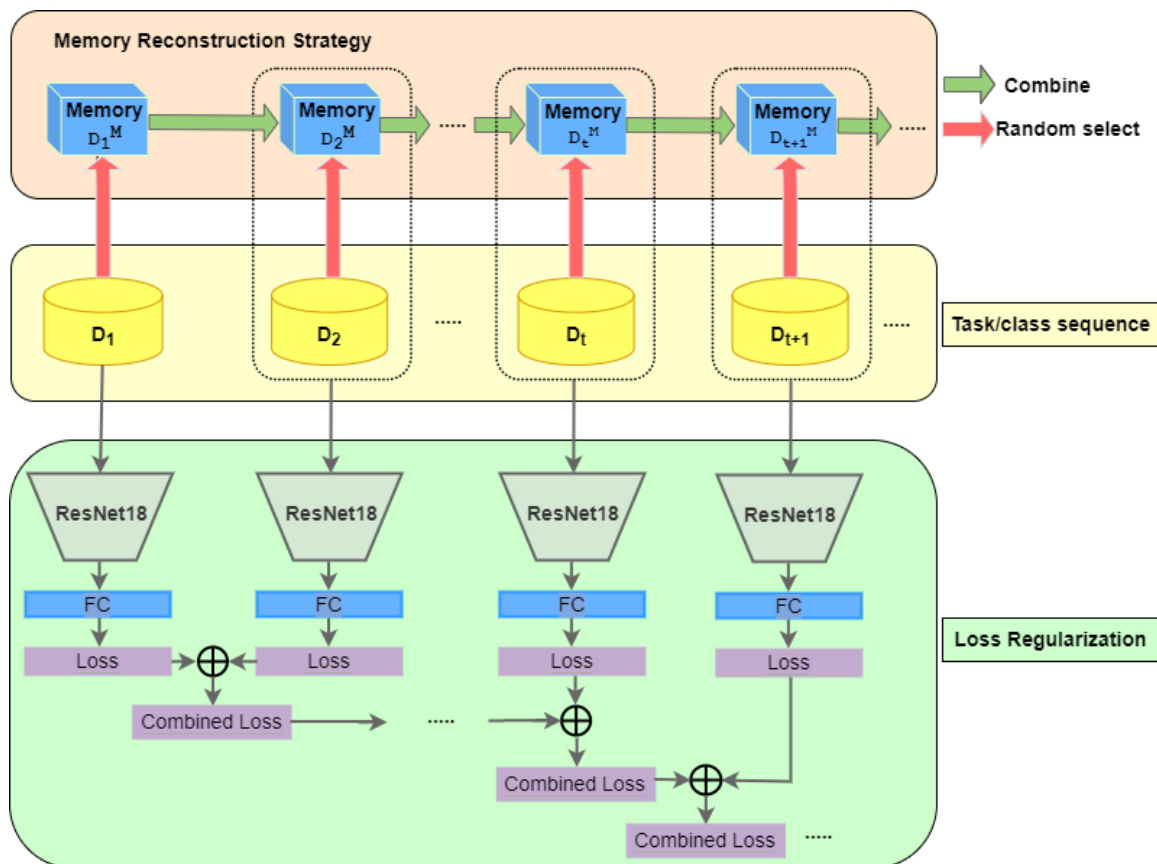


Figure 1. The proposed framework of memory reconstruction and loss regulation strategies for continual learning of a sequence of tasks.

The first component (colored vintage rose in Figure 1) is responsible for creating incremental fine-tuning data for the target task from the source task. This is carried out using a strategy of random sample selection from the data of each task in each step of continual learning (colored yellow in Figure 1). The selected samples are then stored in the replay memory. By doing so, the overfitting phenomenon, which is commonly seen in other rehearsal-based methods, will be avoided. The proposed memory reconstruction strategy is tested on two types of incremental data, including the New Classes (NCs) and New Classes and Instances (NCIs).

Together with the memory reconstruction strategy, the second component (Figure 1) is implemented in the manner of continual parallel learning via Resnet 18 networks. The inputs for each Resnet 18 model are an incremental dataset from the source task and a new dataset from the target task. An optimal strategy based on the combination of the logistic loss functions in each task is deployed to result in a reconstructed buffer memory and a fine-tuned CNN model. This end-to-end learning with the loss combination strategy helps train the target tasks without forgetting the source tasks.

The details of the components in the proposed framework and data scenarios for experiments are indicated in the following sections.

3.1. Memory Reconstructing Strategy

Our memory reconstruction strategy is an improvement to the solution in [22], in which the authors proposed the iCaRL method for class-incremental learning. The constraint for this is that the training data for a small number of classes must be presented at the same time, and the new classes can be progressively added. The solution in [22] comprises three main parts: (1) a 2DCNN feature extractor based on the Resnet18 backbone, (2) a KNN classifier to obtain final result, and (3) exemplar memory. In [22], the samples are selected from the center of the data distribution (mean-sample method), and this could fail into an overfitting problem. In order to overcome this, in our work, a strategy of random sample selection for memory (random-sample memory) is implemented.

As presented in Figure 1, we have a sequence of tasks $D_1, D_2, \dots, D_t, D_{t+1}, \dots$ in which D presents the data of a task. The memory reconstruction strategy is responsible for the random sample selection from each task to store in memory. Firstly, we have the memory D_1^M for the task D_1 , and this memory contains the random samples selected from D_1 . The memory D_2^M is the combination of samples that are randomly chosen from D_2 and the random samples from D_1^M . The same method is deployed for the next memory in the task sequence, and as a result, we have the corresponding memory sequence of $D_1^M, D_2^M, \dots, D_t^M, D_{t+1}^M, \dots$. The memory size for each task is the same and equal to M . It is noted that memory size is fixed to restrict a huge hardware requirement. These memories are then used for training ResNet18 branches in the second component.

In addition, in order to implement the proposed memory reconstruction strategy, in this work, we deploy two cases of data selection based on the types of incremental data, including New Classes (NCs) and New Classes and Instances (NCIs). The first one shows the case in which all categories in the incremental dataset are different from the ones in the previous dataset. The second scenario refers to the interference of some classes between the target and the source memories.

3.1.1. The Incremental Data of New Classes (NCs)

For the NC type, all categories of a target dataset are different from the ones in the source memory. In continual learning, a dataset (D_1) is first given. It concludes s_1 categories, and each category has a unique label of y ($y = 0, \dots, s_1 - 1$). Each category consists of a set of C_1 images $\{x = x_i, (i = 0, \dots, C_1)\}$, and C_1 is the number of images in a class. The D_1 dataset is shown as in Equation (1):

$$\mathcal{D}_1 = \bigcup_{y=(0, \dots, s_1-1)} \{(x, y) : x = \{x_i\}, i = (0, \dots, C_1)\} \quad (1)$$

The pre-trained CNN model (pre-trained on the ImageNet dataset [12]) is fine-tuned by the D_1 dataset. The size of the FC layer is s_1 . A memory is then constructed, and its size is set by M . The images in each class are randomly selected by $f_{rd}(f_{random})$ from C_1 to obtain $K_1 = M/s_1$ images. The memory D_1^M is first created, as illustrated in Equation (2):

$$D_1^M = f_{rd}(D_1) = \bigcup_{y=(0,\dots,s_1-1)} \{(x, y) \in D_1 : x = \{x_i\}, i = (0, \dots, K_1)\} \tag{2}$$

Given the $(t + 1)$ th incremental dataset D_{t+1} with z_{t+1} classes and the previous memory D_t^M with s_t classes, the labels of z_{t+1} classes in the D_{t+1} dataset are different from the labels of s_t categories in the D_t^M buffer memory. The t th buffer memory is presented as in Equation (3):

$$D_t^M = \bigcup_{y=(0,\dots,s_t-1)} \{(x, y) : x = \{x_i\}, i = (0, \dots, K_t)\} \tag{3}$$

Additionally, the $(t + 1)$ th incremental dataset is presented in Equation (4):

$$D_{t+1} = \bigcup_{y=(s_t,\dots,s_{t+1}-1)} \{(x, y) : x = \{x_i\}, i = (0, \dots, C_{t+1})\} \tag{4}$$

For the incremental data type NC, the number of classes in the next memory (D_{t+1}^M) increases from s_t to N . In addition, the output size of the CNN model is also changed from s_t to N as in Equation (5):

$$N = s_{t+1} = s_t + z_{t+1} \tag{5}$$

The reconstructed memory of D_{t+1}^M contains samples that are randomly selected from both the D_t^M buffer and the D_{t+1} dataset. The samples of D_t^M (K_t images) and D_{t+1} (C_{t+1} images) are reduced to $K_{t+1} = M/N$ images, as shown in Equation (6):

$$D_{t+1}^M = \begin{cases} f_{rd}(D_t^M) = \bigcup_{y=(0,\dots,s_t-1)} \{(x, y) \in D_t^M : x = \{x_i\}, i = (0, \dots, K_{t+1})\} \\ f_{rd}(D_{t+1}) = \bigcup_{y=(s_t,\dots,N-1)} \{(x, y) \in D_{t+1} : x = \{x_i\}, i = (0, \dots, K_{t+1})\} \end{cases} \tag{6}$$

3.1.2. The Incremental Data of New Classes and New Instances (NCIs)

In the NCI type, some categories in the incremental dataset have the same labels as some classes in the memory. Therefore, memory reconstruction in an NCI incremental dataset is more complex than in the NC case. For the first dataset D_1 , the initiation memory D_1^M is similarly achieved as in Equation (2). In the NCI case, D_1^M is split into two parts: $D_1^{M(1)}$ and $D_1^{M(2)}$. Similarly, D_2^M is broken into $D_2^{M(1)}$ and $D_2^{M(2)}$. The separation causes $D_1^{M(2)}$ have the same class labels as $D_2^{M(1)}$. In general, given the $(t + 1)$ th incremental dataset D_{t+1} ($D_{t+1} = D_{t+1}^{(1)} \cup D_{t+1}^{(2)}$) with z_{t+1} categories and the previous memory D_t^M ($D_t^M = D_t^{M(1)} \cup D_t^{M(2)}$) with s_t categories, the separation for these are illustrated in Equations (7) and (8), respectively.

$$D_{t+1} = \begin{cases} D_{t+1}^{(1)} = \bigcup_{y=(s_t-k_{t+1},\dots,s_t-1)} \{(x, y) \in D_{t+1} : x = \{x_i\}, i = (0, \dots, C_{t+1})\} \\ D_{t+1}^{(2)} = \bigcup_{y=(s_t,\dots,N-1)} \{(x, y) \in D_{t+1} : x = \{x_i\}, i = (0, \dots, C_{t+1})\} \end{cases} \tag{7}$$

$$D_t^M = \begin{cases} D_t^{M(1)} = \bigcup_{y=(0,\dots,s_t-k_{t+1}-1)} \{(x, y) \in D_t^M : x = \{x_i\}, i = (0, \dots, K_t)\} \\ D_t^{M(2)} = \bigcup_{y=(s_t-k_{t+1},\dots,s_t-1)} \{(x, y) \in D_t^M : x = \{x_i\}, i = (0, \dots, K_t)\} \end{cases} \tag{8}$$

The divisions in Equations (7) and (8) ensure that both $D_{t+1}^{(1)}$ and $D_t^{M(2)}$ have k_{t+1} classes, in which $k_{t+1} < z_{t+1}$ and $k_{t+1} < s_t$. The category labels in $D_{t+1}^{(1)}$ and $D_t^{M(2)}$

are similar. The number of classes (N) of the reconstructed memory D_{t+1}^M is calculated as follows:

$$N = s_{t+1} = s_t + z_{t+1} - k_{t+1} \tag{9}$$

Four subsets— $D_{t+1}^{(1)}$, $D_{t+1}^{(2)}$, $D_t^{M(1)}$, and $D_t^{M(2)}$ —are used to choose the samples for memory reconstruction of D_{t+1}^M , as shown in Equation (10):

$$D_{t+1}^M = f_{rd}(D_t^M) \cup f_{rd}(D_{t+1}) = \begin{cases} f_{rd}(D_t^{M(1)}), & y = (0, \dots, s_t - k_{t+1} - 1) \\ f_{rd}(D_t^{M(2)}) \cup f_{rd}(D_{t+1}^{(1)}), & y = (s_t - k_{t+1}, \dots, s_t - 1) \\ f_{rd}(D_{t+1}^{(2)}), & y = (s_t, \dots, N - 1) \end{cases} \tag{10}$$

Both $D_t^{M(1)}$ and $D_t^{M(2)}$ have remaining samples of $K_{t+1} = M/N$ in each class; $D_t^{M(2)}$ and $D_{t+1}^{(1)}$ have $(K_{t+1}/2)$ images as presented in Equations (11) and (12).

$$f_{rd}(D_t^M) = \begin{cases} f_{rd}(D_t^{M(1)}) = \bigcup_{y=(0, \dots, s_t - k_{t+1} - 1)} \{(x, y) \in D_t^M : x = \{x_i\}, i = (0, \dots, K_{t+1})\} \\ f_{rd}(D_t^{M(2)}) = \bigcup_{y=(s_t - k_{t+1}, \dots, s_t - 1)} \{(x, y) \in D_t^M : x = \{x_i\}, i = (0, \dots, K_{t+1}/2)\} \end{cases} \tag{11}$$

$$f_{rd}(D_{t+1}) = \begin{cases} f_{rd}(D_{t+1}^{(1)}) = \bigcup_{y=(s_t - k_{t+1}, \dots, s_t - 1)} \{(x, y) \in D_{t+1} : x = \{x_i\}, i = (0, \dots, K_{t+1}/2)\} \\ f_{rd}(D_{t+1}^{(2)}) = \bigcup_{y=(s_t, \dots, n-1)} \{(x, y) \in D_{t+1} : x = \{x_i\}, i = (0, \dots, K_{t+1})\} \end{cases} \tag{12}$$

These exemplar memories will be used as a part of the training data for the CNN model, as presented in Section 3.2.

3.2. End-to-End CNN Model with Loss Combination for Continual Learning

Equation (13) shows learning without forgetting deployed for the CNN model with a logistic loss function for data $D = \{(x, y); x = \cup x_i; i = (1, C); y = (0, \dots, s_t, \dots, N)\}$ (obtained from the source data D_t and the target data D_{t+1}), as presented in Equation (13):

$$\mathcal{L} = - \sum_{(x,y) \in \mathcal{D}} \left\{ \sum_{y=s_t}^N (\delta_{y=y_i} \log(g_y(x_i))) + \delta_{y \neq y_i} \log(1 - g_y(x_i)) + \sum_{y=0}^{s_t-1} (q_i^y \log(g_y(x_i))) + (1 - q_i^y) \log(1 - g_y(x_i)) \right\} \tag{13}$$

where $\{(x, y) (y = (0, 1, \dots, s_t - 1))\}$ are training images of the old classes and $\{(x, y) (y = (s_t, \dots, N))\}$ are training images of the new classes.

In this study, our proposed CNN framework for continual learning contains parallel CNN models:

- The \mathcal{CNN}_t model is trained on the previous dataset. The FC layer of the \mathcal{CNN}_t model is equal to the label number s_t of the D_t^M memory;
- The extension model of \mathcal{CNN}_t called \mathcal{CNN}_{t+1} has the outputs $N = s_t + z_{t+1}$ for the NC case and $N = s_t + z_{t+1} - k_{t+1}$ for the NCI case.

The optimal strategy for our continual learning framework is presented in Equation (14). It is the optimal combination of the loss functions of \mathcal{CNN}_t (\mathcal{L}_{CNN_t}) and \mathcal{CNN}_{t+1} ($\mathcal{L}_{CNN_{t+1}}$) models.

$$\mathcal{L}_{t+1} = (1 - \alpha) \mathcal{L}_{CNN_t} + \alpha \mathcal{L}_{CNN_{t+1}} \tag{14}$$

In Equation (14), the loss function of the previous CNN model is \mathcal{L}_{old} ($\mathcal{L}_{old} = \mathcal{L}_{CNN_t}$) and the new one is \mathcal{L}_{new} ($\mathcal{L}_{new} = \mathcal{L}_{CNN_{t+1}}$). They are related to each other through a hyperparameter α , which is also considered in our experiments. Although this formula nearly looks like an optimization function of the LwF method in [7], the details of our components are not similar to this. In [7], the authors blocked all parameters of the old model and created a shared network to balance the weights between the old model and the new one. Moreover, the old data do not participate in retraining the new model. In

our work, we do not block the \mathcal{CNN}_t network. The deepest layers of the \mathcal{CNN}_t model are frozen during fine-tuning on the top layers (the layers before the FC layer). Furthermore, our strategy for data storage is different from the LwF method. In our research, we pass \mathcal{D}_{old} to the \mathcal{CNN}_t network with the loss function \mathcal{L}_{old} , as illustrated in Equation (15):

$$\mathcal{L}_{old} = \mathcal{L}_{CNN_t} = - \sum_{(x_i, y_i) \in \mathcal{D}_{old}}^{\mathcal{CNN}_t} \left\{ \sum_{y=0}^{s_t-1} (P_i^y \log(g_y(x_i))) + (1 - P_i^y) \log(1 - g_y(x_i)) \right\} \quad (15)$$

where the sub-dataset for fine-tuning the \mathcal{CNN}_t model is $\mathcal{D}_{old} = \mathcal{D}_t^M$ for the NC case and $\mathcal{D}_{old} = \mathcal{D}_t^M \cup \mathcal{D}_{t+1}^{(1)}$ for the NCI scenario.

For the element $\mathcal{L}_{new} = \mathcal{L}_{CNN_{t+1}}$, it is different from the loss function of the iCaRL method [22] that is shown in Equation (13). In this, the authors used the output of the $\mathcal{CNN}_t (q_i^y)$ for calculating the samples in \mathcal{D}_{t+1} . In this work, we apply the ground-truth samples P_i^y in \mathcal{D}_{old} . In addition, the method of [22] only concentrates on the \mathcal{L}_{new} element, and our study balances the \mathcal{L}_{old} part and the \mathcal{L}_{new} part. The remaining part of the loss function is composed of two elements, as illustrated in Equation (16):

$$\mathcal{L}_{new} = \mathcal{L}_{CNN_{t+1}} = - \sum_{(x_i, y_i) \in \mathcal{D}_{new}}^{\mathcal{CNN}_{t+1}} \left\{ \sum_{y=s_t}^N (\delta_{y=y_t} \log(g_y(x_i))) + \delta_{y \neq y_t} \log(1 - g_y(x_i)) + \sum_{y=0}^{s_t-1} (P_i^y \log(g_y(x_i))) + (1 - P_i^y) \log(1 - g_y(x_i)) \right\} \quad (16)$$

where the sub-dataset \mathcal{D}_{new} for the NC incremental data is presented in Equation (17):

$$\mathcal{D}_{new} = \mathcal{D}_t^M \cup \mathcal{D}_{t+1} = \begin{cases} \mathcal{D}_t^{M(1)}, & y = (0, \dots, s_t - 1) \\ \mathcal{D}_{t+1}, & y = (s_t, \dots, N - 1) \end{cases} \quad (17)$$

Additionally, the sub-dataset \mathcal{D}_{new} for the NCI data is indicated in Equation (18):

$$\mathcal{D}_{new} = \mathcal{D}_t^M \cup \mathcal{D}_{t+1} = \begin{cases} \mathcal{D}_t^{M(1)}, & y = (0, \dots, s_t - k_{t+1} - 1) \\ \mathcal{D}_t^{M(2)} \cup \mathcal{D}_{t+1}^{(1)}, & y = (s_t - k_{t+1}, \dots, s_t - 1) \\ \mathcal{D}_{t+1}^{(2)}, & y = (s_t, \dots, N - 1) \end{cases} \quad (18)$$

In both cases of NC and NCI, the continual learning CNNs utilize two inputs: an incremental dataset and a new dataset. They output a reconstructed buffer memory and the fine-tuned CNN model.

4. Experimental Results

In this section, we first investigated the efficiency of our memory reconstruction strategy. The experiments for this were set in two scenarios: (1) using the network architecture of iCaRL in [22] and our proposed memory reconstruction strategy (random-sample memory), and (2) using the proposed architecture with loss combination of CNN models and random-sample memory (as shown in Figure 1). The results of the first scenario were compared with those of iCaRL method to show the better performance of our memory reconstruction strategy in comparison with iCaRL. In the second scenario, the performance of our memory reconstruction strategy with loss combination was evaluated with the different cases of the α coefficient in Formula (14). Based on this, the best α value was chosen for further comparative evaluations with other SOTA methods.

Secondly, the two classification strategies the end-to-end method and the discrete method were deployed in two cases of NC and NCI with two data protocols: (i) KinectLeap-Creative Senz3D (Figure 2a) and (ii) Creative Senz3D-KinectLeap (Figure 2b):

- End-to-end method: Resnet18 was applied to the entire architecture from the first layer to the last layer (FC layer) of both the training and testing phases.
- Discrete method: For the training stage, Resnet18 is an end-to-end system, but for the testing stage, Resnet18 was only used as a feature extractor and KNN was utilized as a classifier.

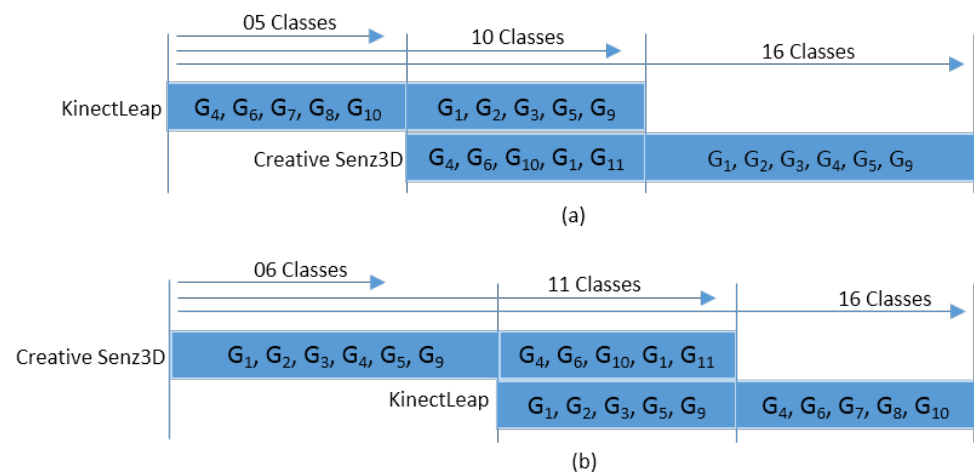


Figure 2. Two data protocols are used for both NC and NCI evaluations: (a) KinectLeap-Creative Senz3D protocol and (b) Creative Senz3D-KinectLeap protocol.

The experiments were conducted at different memory sizes and numbers of classes in the CIFAR-10, CIFAR-100, and CORE-50 datasets. The memory size with the best result was chosen for the evaluation of the continual learning of NC and NCI cases. The proposed framework for continual learning was also evaluated and compared with other SOTA methods. The evaluation schemes were written in Python on a Pytorch deep learning framework and run on a workstation with a NVIDIA GPU 11G. The optimal function was stochastic gradient descent (SGD). The learning rate (lr) was equal to 2.0 and automatically reduced after each 30 epochs. Resnet18 was used as the backbone model.

4.1. Datasets and Protocols

4.1.1. Datasets

Five datasets, including CIFAR-10 and CIFAR-100 [34], CORE-50 [35], KinectLeap [36], and Creative Senz3D [37], were used in the experiments to demonstrate the effectiveness of our method. CIFAR-10 consists of 10 classes with 6000 color images of 32×32 resolution for each class. CIFAR-100 contains 100 classes with 600 images for each class. The 100 classes in the CIFAR-100 were grouped into 20 superclasses. For example, a superclass *insect* includes the classes (*bee, beetle, butterfly, caterpillar, cockroach*). Each image of a class has two labels: one indicates the class, and the other presents the superclass to which it belongs. CORE50 has 50 domestic objects belonging to 10 categories: plug adapters, mobile phones, scissors, light bulbs, cans, glasses, balls, markers, cups, and remote controls. The classification can be performed at the object level with 50 classes or at the category level with 10 classes. The first task is much more challenging because objects of the same category are very difficult to distinguish in certain poses. In this work, the experiments were conducted with the first task. KinectLeap includes 1400 gestures performed by 14 different people, each performing 10 different gestures repeated 10 times each. The Creative Senz3D dataset contains different static gestures acquired with the Creative Senz3D camera. Four different people performed the gestures, each repeating 11 different gestures 30 times for a total of 1320 samples. For each sample, color, depth, and confidence frames are available. In our work, the color frames were utilized for the experiments.

In this work, two datasets, CIFAR-100 and CORE-50, were utilized for evaluating the case of NC continual learning. These datasets are the most challenging in the field of continual learning. Several advanced methods use them for incremental learning evaluation. KinectLeap and Creative Senz3D are the most commonly used datasets in the field of hand gesture recognition. Ten classes of KinectLeaps ($D_{KinectLeap} = \{G_j, j = (1, \dots, 10)\}$) and eleven classes of Creative Senz3D ($D_{CreativeSenz3D} = \{G_j, j = (1, \dots, 11)\}$) were used to evaluate both cases of NC and NCI. The experimental protocols deployed for these datasets are presented in detail in the next section.

4.1.2. Data Protocols

For NC evaluation: two benchmark continual learning datasets, CIFAR-100 and CORE-50, were individually utilized for this evaluation. The dataset contains P classes, $P = 100$ for CIFAR-100 and $P = 50$ for CORE-50. We equally divided each dataset into T subsets ($T = 10$). Thus, class labels in incremental datasets ($z_t = P/T, t = 0, \dots, T - 1$) are independent of each other. All images of the first category $s_0 = z_0$ (e.g., $s_0 = 10$ and $s_0 = 5$ for CIFAR-100 and CORE-50, respectively) were used for training and testing the first CNN model. The classes for the incremental dataset at an instance in time were $s_{t+1} = s_t + z_{t+1}$, ($t = 1, \dots, T - 1$), with s_t of the CIFAR-100 dataset at (20, 30, ..., 100), and s_t of the CORE-50 dataset at (10, 15, 20, ..., 50). At an instance in time, we applied the “leave-one-subject-out” method [13] for data splitting in the training and testing phases of a continuous learning CNN model.

Two published hand gesture datasets, KinectLeap and Creative Senz3D, contain categories with various hand images. Both datasets were used to evaluate our incremental learning method in both the NCI and NC types. Because five hand postures (G_1, G_2, G_3, G_5 , and G_9) of the KinectLeap dataset are similar to those of the Creative Senz3D dataset (G_4, G_6, G_{10}, G_1 , and G_{11}), we divided the KinectLeap dataset into two parts, as shown in Equation (19):

$$\mathcal{D}_{KL} = D_{KL}^1 \cup D_{KL}^2 = \begin{cases} D_{KL}^1 = G_4 \cup G_6 \cup G_7 \cup G_8 \cup G_{10} \\ D_{KL}^2 = G_1 \cup G_2 \cup G_3 \cup G_5 \cup G_9 \end{cases} \quad (19)$$

Creative Senz3D is also separated into two subsets, as shown in Equation (20):

$$\mathcal{D}_{Zen} = D_{Zen}^1 \cup D_{Zen}^2 = \begin{cases} D_{Zen}^1 = G_4 \cup G_6 \cup G_{10} \cup G_1 \cup G_{11} \\ D_{Zen}^2 = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_9 \end{cases} \quad (20)$$

Based on these divisions, the categories of D_{KL}^2 are similar to those of D_{Zen}^1 , and some pairs of hand gestures in D_{KL}^2 and D_{Zen}^1 are similar ($G_1-G_4, G_2-G_6, G_3-G_{10}, G_5-G_1$, and G_9-G_{11}). The classes of D_{KL}^1 are different from those of D_{KL}^2 . Thus, we deployed two evaluation protocols, as shown in Figure 2, in which the incremental classes were divided into three parts. In order to compare the cross-evaluation between the datasets, we deployed the scenarios for the NC evaluation as follows:

- KinectLeap was first used (Figure 2a). The incremental datasets $D_t (t = 0, \dots, 2)$ were consequently implemented by $D_{KL}^1, D_{KL}^2 \cup D_{Zen}^1$, and D_{Zen}^2 .
- The first incremental dataset belongs to Creative Senz3D, as illustrated in Figure 2b. It has three incremental folds of $D_t (t = 0, \dots, 2)$: $D_{Zen}^2, D_{Zen}^1 \cup D_{KL}^2$, and D_{KL}^1 .

For NCI evaluation: the experiments were deployed for KinectLeap and Creative Senz3D in the following two cases:

- KinectLeap was utilized in the previous dataset $D_t = D_{KL} = D_{KL}^1 \cup D_{KL}^2$ (Figure 2a), and Creative Senz3D is an incremental dataset: $D_{t+1} = D_{Zen} = D_{Zen}^1 \cup D_{Zen}^2$.
- Creative Senz3D was utilized in the previous dataset $D_t = D_{Zen} = D_{Zen}^2 \cup D_{Zen}^1$ (Figure 2b), and KinectLeap is an incremental dataset: $D_{t+1} = D_{KL} = D_{KL}^2 \cup D_{KL}^1$.

4.2. Efficiency Evaluation of the Memory Reconstruction Strategy

In this evaluation, two experimental datasets, CIFAR-100 and CORE-50, were utilized.

4.2.1. Scenario 1: Using the iCaRL Architecture with Random-Sample Memory

The solution of using the iCaRL architecture in [22] and our proposal of random sample selection for memory is named RiCaRL (Random iCaRL). We compared RiCaRL with the mean-sample method of iCaRL. The comparative results are shown in scenarios with different memory sizes (1000, 2000, 3000, and 4000) and class number (CIFAR-100 includes the arithmetic progression from 10 to 100 with a step of 10; CORE-50 contains

an arithmetic sequence from 5 to 50 with a deviation of 5). Tables 1 and 2 present the experimental results for the CIFAR-100 and CORE-50 datasets, respectively. In general, the recognition accuracy obtained by the RiCaRL method in all experimental cases is higher than that of the iCaRL method.

Table 1. Comparison of gesture recognition accuracy (%) between random-sample method (RiCaRL) and mean-sample solution (iCaRL) at different memory sizes and task sizes of the CIFAR-100 dataset. The deviation values between RiCaRL and iCaRL are shown under the uparrow.

Task Size	Memory Size = 1000			Memory Size = 2000			Memory Size = 3000			Memory Size = 4000		
	iCaRL	RiCaRL	(↑)	iCaRL	RiCaRL	(↑)	iCaRL	RiCaRL	(↑)	iCaRL	RiCaRL	(↑)
10	82.18	86.02	3.84	85.8	88.1	2.3	86.19	88.14	1.95	87.92	88	0.08
20	73.02	74.97	1.95	75.35	76.85	1.5	75.69	78.51	2.82	76.25	80	3.75
30	68.05	70.78	2.73	70.23	72.4	2.17	71.23	75.81	4.58	71.66	76.03	4.37
40	62.71	64.07	1.36	64.52	66.8	2.28	66.47	68.79	2.32	66.52	70.4	3.88
50	57.08	60.52	3.44	61.04	62.06	1.02	61.29	64.62	3.33	62.28	65.84	3.56
60	54.39	57.33	2.94	57.55	59.02	1.47	58.37	60.18	1.81	58.91	62.08	3.17
70	51.21	53.46	2.25	54.96	56.14	1.18	55.61	58.43	2.82	56.92	60.52	3.6
80	48.47	50.77	2.3	51.56	52.6	1.04	52.83	54.15	1.32	53.41	55.72	2.31
90	44.96	47.81	2.85	48.57	50.14	1.57	50.96	52.39	1.43	51.88	54.8	2.92
100	41.19	43.65	2.46	45.93	47.24	1.31	47.11	48.68	1.57	49.19	49.48	0.29

Table 2. Comparison of gesture recognition accuracy (%) between random-sample method (RiCaRL) and mean-sample solution (iCaRL) at different memory sizes and task sizes of the CORE-50 dataset. The deviation values between RiCaRL and iCaRL are shown under the uparrow.

Task Size	Memory Size = 1000			Memory Size = 2000			Memory Size = 3000			Memory Size = 4000		
	iCaRL	RiCaRL	(↑)	iCaRL	RiCaRL	(↑)	iCaRL	RiCaRL	(↑)	iCaRL	RiCaRL	(↑)
5	95.24	98.06	2.82	100	100	0	100	100	0	100	100	0
10	92.79	95.61	2.82	99.08	98.74	−0.34	98.47	99.36	0.89	98.21	99.75	1.54
15	89.32	91.52	2.2	91.69	93.48	1.79	92.09	96.16	4.07	93	98.84	5.84
20	82.07	84.29	2.22	85.12	88.13	3.01	87.39	92.29	4.9	89.68	96.77	7.09
25	73.5	77.65	4.15	79.64	83.16	3.52	81.95	88.46	6.51	85.99	94.1	8.11
30	67.39	71.14	3.75	73.63	78.93	5.3	77.58	84.21	6.63	81.47	89.17	7.7
35	63.29	67.37	4.08	69.8	74.02	4.22	72.31	81.39	9.08	78.86	90.19	11.33
40	58.19	62.42	4.23	65.03	68.34	3.31	70.08	78.64	8.56	76.3	89.99	13.69
45	54.71	59.47	4.76	61.034	64.32	3.286	67.81	73.63	5.82	73.51	86.49	12.98
50	48.56	53.37	4.81	57.491	59.82	2.329	61.79	71.19	9.4	71.05	84.87	13.82

In Table 1, at a memory size of 1000, the average increase in the recognition accuracy of our solution RiCaRL compared with iCaRL is 2.61%. The results at other memory sizes of 2000, 3000, and 4000 are 1.58%, 2.395%, and 2.793%, respectively. It can be seen that the highest deviation in the accuracy between RiCaRL and iCaRL belongs to the memory size of 4000, with the largest one being 4.37% at 30 classes. The results for the CORE-50 dataset in Table 2 show that the average increase in the recognition accuracy of the RiCaRL method compared with iCaRL is 3.58% at a memory size of 1000. The results for other memory sizes of 2000, 3000, and 4000 are 2.64%, 5.59%, and 8.21%, respectively. In comparison with the CIFAR-100 dataset, the results for the CORE-50 dataset are higher in all experiments, and the highest variation in the accuracy between RiCaRL and iCaRL also belongs to the memory case of 4000, with the biggest one being 13.82% at 50 classes.

From the above experimental analysis on both the CIFAR-100 and CORE-50 datasets, we see that, as the memory size increases, the gesture recognition accuracy also climbs up at all task sizes for both methods of iCaRL and RiCaRL. This shows that adding more samples at each step of incremental learning will be helpful for leveraging the knowledge from source tasks to target tasks. Accordingly, the performance of gesture recognition will be improved. In the experiments, the memory size was increased from 1000 to 4000.

At the maximum size of 4000, the perfect recognition accuracy of 100% was obtained for the task size of 5. When the task size increases, the percentage rate of accuracy decreases. This happens obviously in the classification problem. However, what we want to show through these results is that the memory size of 4000 is a reasonable choice for having high recognition accuracy (71.055% for iCaRL and 84.87% for RiCaRL). It ensures the balance between memory capacity and recognition accuracy. The larger the memory capacity, the more computing power and storage space there are. This is not suitable for real-world applications. In the next evaluations, we apply a random-sample memory size of 4000 for the experiments.

In addition, the higher experimental results of our RiCaRL method compared with iCaRL prove the important role of random sample selection for memory reconstruction in continual learning.

4.2.2. Scenario 2: Using the Incremental Architecture of CNN Models with Loss Combinations and Random-Sample Memory

The recognition accuracy of the proposed architecture with loss combination and random-sample memory (presented in Figure 1) was first evaluated on different coefficients of α . Tables 3 and 4 show the recognition accuracy obtained at a memory size of 4000. The learning rate was 2.0 and then reduced after 30 epochs. The values of α were tested at 0.25, 0.5, and 0.75.

Table 3. The recognition results (%) of the RLC-Resnet method on the CORE-50 dataset with different α coefficients of 0.25, 0.5, 0.75; the task size was 5; and the memory size was 4000.

Alpha/Task Size	0.25	0.5	0.75
5	99.97	100	99.93
10	99.88	100	99.78
15	98.97	99.40	99.00
20	97.02	99.15	97.10
25	95.22	97.79	94.57
30	89.35	96.10	89.14
35	91.16	94.48	90.10
40	89.98	92.36	88.74
45	87.72	91.55	86.55
50	85.21	88.22	84.51

Table 4. The recognition results (%) of the RLC-Resnet method on the CIFAR100 dataset with different α coefficients of 0.25, 0.5, 0.75; the task size was 5; and the memory size was 4000.

Alpha/Task Size	0.25	0.5	0.75
10	87.50	88.92	89.70
20	81.05	82.25	83.05
30	75.13	77.86	77.86
40	69.35	72.72	70.85
50	65.56	66.89	66.60
60	61.56	64.58	62.65
70	58.41	61.77	60.74
80	53.75	57.07	56.55
90	53.11	54.78	55.55
100	48.32	52.56	50.18

It can be seen from Table 3 that the best results belong to the α value of 0.5 at all task sizes in the CORE-50 dataset. This is almost the same for the CIFAR-100 dataset (in Table 4), except for the task sizes of 10, 20, and 90, with the best ones belonging to $\alpha = 0.75$. From these observations, we chose an α value of 0.5 for the experiments in the next sections.

The experimental results shown in Tables 3 and 4 prove the efficiency of using the incremental architecture of CNN networks with loss combinations and random-sample

memory. In comparison with the RiCaRL method, it has higher recognition rates. At $\alpha = 0.5$, memory size = 4000, and maximum task sizes of CORE50 and CIFAR-100, the results are 88.22% and 52.56%, respectively. These percentages are higher than the ones of RiCaRL (presented in Tables 1 and 2), with 84.87% for CORE50 and 49.48% for CIFAR-100. These results indicate that the combination of losses in training the end-to-end architecture of Resnet networks (shown in Figure 1) helps to find a compromise between minimizing \mathcal{L}_{CNN_t} and minimizing $\mathcal{L}_{CNN_{t+1}}$, with the extent of compromise chosen by $\alpha = 0.5$.

4.3. Evaluations with NC Type

In this section, two classification strategies are deployed: (1) end-to-end Resnet18 classification, and (2) a discrete method with a Resnet18 feature extractor and KNN classifier. The first strategy was implemented in [7,22,38] with the LwF, iCaRL, and AOP methods, respectively. In this work, we compare these methods with the solutions named **RiCaRL-Resnet** (using the proposed memory reconstruction strategy and Resnet backbone as in [7] or [22]) and **RLC-Resnet** (using the proposed memory reconstruction strategy and loss combination of Resnet18 models). The second classification strategy is tested on three solutions: **iCaRL-Knn**, **RiCaRL-Knn**, and **RLC-Knn**. The evaluations are deployed on the four above datasets, and the data protocols are applied for the incremental NC type.

Firstly, the CORE-50 and CIFAR-100 datasets are used in the experimental results presented in Figures 3 and 4, respectively. Figures 3a and 4a show the results for the first classification strategy. The results for the second strategy are presented in Figures 3b and 4b. It can be seen from these figures that the experimented methods show decreases in accuracy as the number of classes increases. This happens for both classification strategies. However, the RLC-Resnet method has the least reduction compared with the other methods. This is indicated more clearly in the first classification strategy (Figures 3a and 4a) than in the second one (Figures 3b and 4b). In a comparison between the two classification strategies on the same experimental dataset, either CORE-50 or CIFAR-100, in general, we find that the second strategy has higher accuracy than the first one for all class numbers. This is remarkable for all solutions but not the RiCaRL-Resnet and RLC-Resnet methods. In practical applications, an end-to-end architecture is more preferred than a discrete one. This means that our end-to-end solutions, RiCaRL-Resnet and RLC-Resnet, balance both high recognition accuracy and applicability.

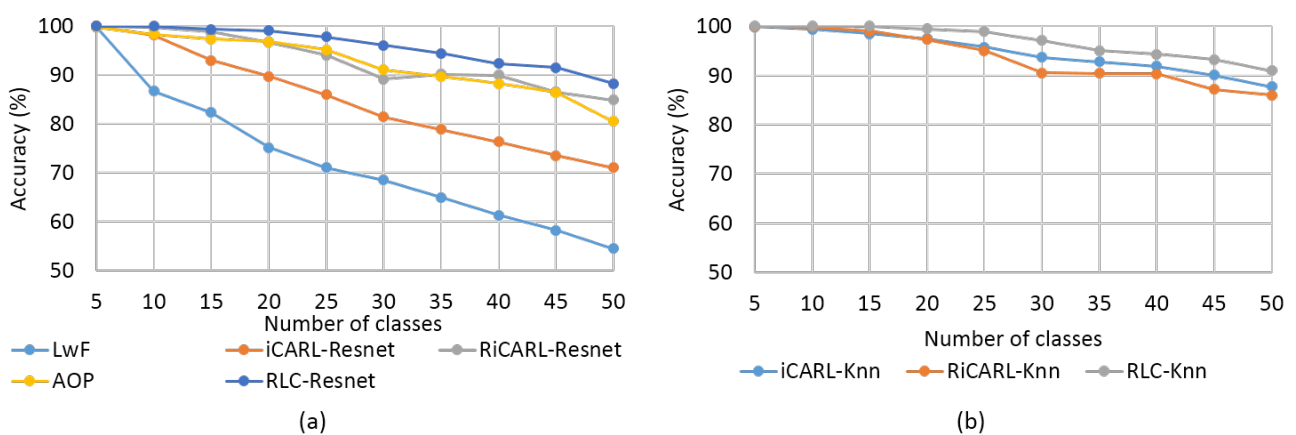


Figure 3. The recognition accuracy for the CORE-50 dataset at different class numbers and methods with two classification strategies: (a) End-to-end Resnet18 classifier, (b) Resnet18 feature extractor and KNN classifier.

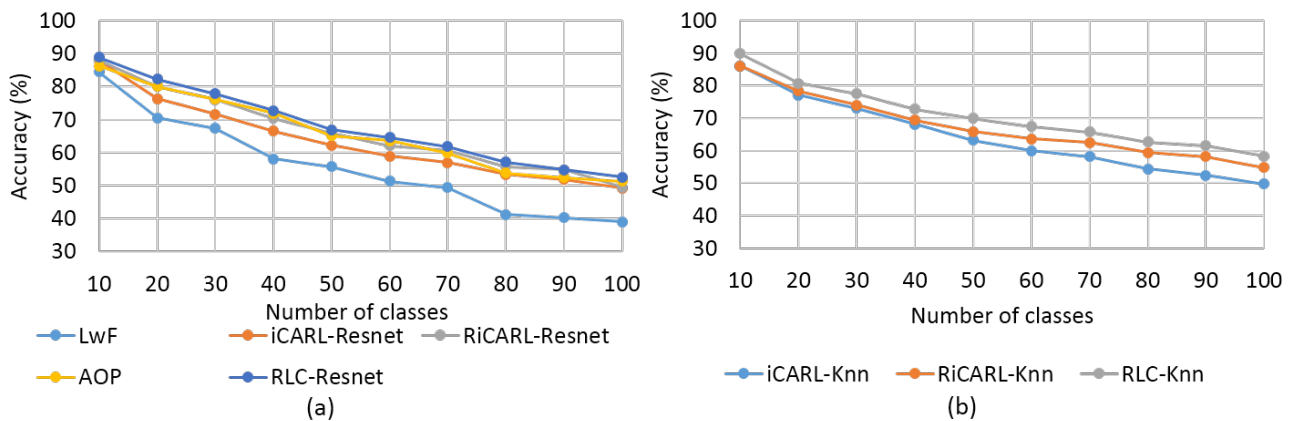


Figure 4. The recognition accuracy for the CIFAR-100 dataset at different class numbers and methods with two classification strategies: (a) End-to-end Resnet18 classifier, (b) Resnet18 feature extractor and KNN classifier.

In a detailed analysis of the CORE-50 dataset, we see that, with the first classification strategy (Figure 3a), the LwF method has the lowest accuracy, with 86.72% for 10 classes, followed by the methods of iCaRL-Resnet, with 98.21%; AOP, with 98.42%; RiCaRL-Resnet, with 99.75%; and the highest one, RLC-Resnet, with 100%. These results are reduced significantly when the class number increases. For 50 classes, LwF gains only 54.49%, iCaRL-Resnet gains 71.05%, AOP gains 80.53%, RiCaRL-Resnet and RLC-Resnet gains 84.87% and 88.22%, respectively. In the second classification strategy (Figure 3b), iCaRL-Knn has the lowest accuracy, with 99.43% and 87.74% for 10 and 50 class numbers, respectively. These are 1.22% and 16.69% higher than those of the first strategy. RiCaRL-Knn and RLC-Knn have higher accuracy compared with iCaRL-Knn, with 99.81% and 100% for 10 classes, and 86.05% and 91.03% for 50 classes, respectively. In comparison with the ones from the first strategy, these numbers are only about 2% to 3% higher.

The experimental results on the CIFAR-100 dataset also show the same progress as CORE-50. Figure 4a shows the results of the first classification strategy. It can be seen that LwF has the lowest accuracy, with 84.51%, for 10 classes compared with the other four methods: AOP (86.31%), iCaRL-Resnet (87.92%), RiCaRL-Resnet (88%), and RLC-Resnet (88.92%). However, for 100 classes, these numbers decrease significantly to 39.04%, 51.38%, 49.19%, 49.48%, and 52.56% for each. In the second classification strategy (Figure 4b), we see slight decreases in accuracy for 10 classes of iCaRL-Knn (86%) and RiCaRL-Knn (86.1%) compared with the ones with the first strategy. However, RLC-Knn has a tiny increase in accuracy of 89.9% in comparison with 88.92% for the first strategy. The modest reduction also occurs in class numbers 20, 30, and 40 for the methods RiCaRL-Knn and RLC-Knn, but for other class numbers, the trend increases slightly. For 100 classes, iCaRL-Knn has an accuracy of 49.79%, while RiCaRL-Knn and RLC-Knn have accuracies of 54.74% and 58.32%, respectively. These numbers are all higher than those of the first classification strategy.

In addition to experiments on two commonly used databases for continual learning, CORE-50 and CIFAR-100, the NC incremental evaluation is also investigated on two hand gesture datasets: KinectLeap and Creative Senz3D. The memory size is 500 images, and down-sampling of the buffer is implemented as presented in Section 3.2. The two data protocols for these datasets are deployed as shown in Section 4.1. The learning rate (lr) is equal to 2.0, which is then reduced after 30 epochs. Resnet18 is used as the backbone for both models.

Table 5 shows the recognition accuracy of the various continual learning methods with two data protocols: KinectLeapCreative Senz3D (the left side of Table 5), and Creative Senz3D - KinectLeap (the right side of Table 5). In the first data protocol for NC type, three tasks are set up, in which task 1 contains 5 classes, task 2 includes 10 classes, and task 3 has 16 classes (as shown in Figure 2a). The second data protocol for the NC type also has three

tasks: task 1 contains 6 classes, task 2 has 11 classes, and task 3 has 16 classes (as indicated in Figure 2b). All values of recognition accuracy in Table 5 are the averages of five runs, with the standard deviation shown beside each result.

It can be seen from Table 5 that the RLC-Resnet method obtains a higher accuracy on both data protocols, as well as the entire continual learning step of the tasks. For protocol 1, for tasks with 5, 10, and 16 classes, the RLC-Resnet method stably reduces from 94.06% to 86.58%. For a task with five classes, it is slightly higher than the LwF, iCaRL, and AOP methods. However, for tasks sizes of 10 and 16, it is extremely larger than the LwF method and about 6% higher than the iCaRL method. This trend is similar to protocol 2, with the RLC-Resnet method obtaining 98.61%, 92.37%, and 85.85% for task sizes of 6, 11, and 16, respectively.

The experimental results on KinectLeapCreative and Senz3Dshow with two data protocols for the NC type show the prospect of our RLC-Resnet method for improving the hand gesture recognition system on various hand gesture datasets of the NC type. This also shows its meaning in the practical implementation of hand gesture recognition systems, in which the hand gesture datasets belong mainly to the NC case.

Table 5. Hand gesture recognition results (%) of the RLC-Resnet method and other solutions on two data protocols—KinectLeap-Creative Senz3D and Creative Senz3D-KinectLeap—for different task sizes of the NC case. All values are averaged over five runtimes, and the standard deviation is shown beside each result.

KinectLeap-Creative Senz3D				
Task Size	LwF	iCaRL	AOP	RLC-Resnet
5	93.22 ± 0.5	92.37 ± 0.4	93.67 ± 0.5	94.06 ± 0.2
10	24.31 ± 0.8	84.00 ± 0.7	86.32 ± 0.4	90.25 ± 0.3
16	20.73 ± 0.7	80.88 ± 0.4	82.39 ± 0.5	86.58 ± 0.5
Creative Senz3D-KinectLeap				
Task Size	LwF	iCaRL	AOP	RLC-Resnet
6	98.07 ± 0.5	97.91 ± 0.6	98.54 ± 0.5	98.61 ± 0.4
11	23.15 ± 0.7	86.57 ± 0.9	89.25 ± 1.1	92.37 ± 0.7
16	17.68 ± 0.8	79.59 ± 0.5	82.91 ± 0.7	85.85 ± 0.5

Further evaluations are shown in Table 6 to show the comparative results between the RLC-Resnet method and other solutions on the CIFAR-10 and CIFAR-100 datasets. The memory sizes and task sizes for these experiments are chosen as the best ones in [25], with memory sizes of 5000 and 1000 and task sizes of 10 and 5 for the CIFAR-100 and CIFAR-10 datasets, respectively. In addition, in this work, we also evaluate the proposed solution of RLC-Resnet on the CORE-50 dataset, with a memory size of 4000 and a task size of 5. All experimental results are averaged over fifteen runtimes, and the standard deviation is shown beside each result.

It can be seen that, in comparison with the results of the best methods—OCM [25], with 81.09% for the CORE-50 dataset, and AOP [38], with 80.53% and 51.38% for the CIFAR-10 and the CIFAR-100 datasets, respectively—the results obtained with our RLC-Resnet method are higher, with 88.46%, 85.37%, and 53.84% for the CORE-50, CIFAR-10, and CIFAR-100 datasets, respectively.

Table 6. Recognition results in the NC case of different solutions and RLC-Resnet method on the datasets of CORE-50 (5 tasks), CIFAR-10 (5 tasks), and CIFAR-100 (10 tasks). All values are averaged over 15 runtimes, and the standard deviation is shown beside each result.

No	Method	CORE-50 Dataset (%)	CIFAR-10 Dataset (%)	CIFAR-100 Dataset (%)
1	iCaRL, 2017 [22]	71.28 ± 0.6	64.28 ± 0.9	50.29 ± 0.5
2	AGEM, 2018 [27]	-	22.6 ± 0.7	6.5 ± 0.2
3	LwF, 2018 [7]	54.61 ± 0.7	31.16 ± 1.2	41.28 ± 0.7
4	GSS, 2019 [44]	-	40.1 ± 1.4	17.4 ± 0.1
5	MIR, 2019 [45]	-	41.0 ± 0.6	24.1 ± 0.2
6	ER, 2020 [48]	-	44.3 ± 0.4	14.4 ± 0.9
7	GDumb, 2020 [43]	-	63.5 ± 0.5	36.0 ± 0.5
8	DER++, 2020 [30]	-	54.7 ± 2.2	27.0 ± 0.7
9	ASER, 2021 [46]	-	44.7 ± 1.2	27.1 ± 0.3
10	SCR, 2021, [47]	-	64.1 ± 1.2	36.5 ± 0.2
11	IL2A, 2021 [28]	-	58.2 ± 1.2	22.4 ± 0.2
12	Co ² L, 2021 [49]	62.17 ± 0.5	58.8 ± 0.4	32.2 ± 0.5
13	OCM, 2022 [25]	81.09 ± 0.4	77.2 ± 0.5	42.4 ± 0.5
14	AOP, 2022 [38]	72.81 ± 0.5	80.53 ± 0.7	51.38 ± 0.6
15	RLC-Resnet (our)	88.46 ± 0.4	85.37 ± 0.6	53.84 ± 0.5

4.4. Evaluations with NCI Type

This evaluation is implemented on two hand gesture datasets. For the NCI type, the data protocols and the optimization function are presented in detail in Sections 3.1.2, 4.1.2 and 3.2. The memory size, learning rate, and Resnet18 model are similar to the ones in the previous section. The task size is 16 for both data protocols: KinectLeap-Creative Senz3D and Creative Senz3D-KinectLeap.

The continual learning methods of LwF, iCaRL, AOP, and RLC-Resnet are compared with the original transfer learning of Resnet18 [33], as illustrated in Table 7. In the case of the pre-trained Resnet model on KinectLeap dataset and testing on Creative Senz3D, the recognition result is 30.38%. Compared with the continual learning methods with D_t being KinectLeap and D_{t+1} being Creative Senz3D, this result is a little higher than that for LwF (29.75% for the NC case and 30.25% for the NCI case) but much lower than that for the iCaRL (80.88% for the NC case and 81.74% for the NCI case), AOP (81.67% for the NC case and 83.03% for the NCI case), and RLC-Resnet (86.58% for NC and 88.21% for NCI case) method. This also happens to the case of D_t with Creative Senz3D and D_{t+1} with KinectLeap. Transfer learning of ResNet18 obtains only 32.08% compare with LwF, with 28.56% and 30.26%; iCaRL, with 79.59% and 81.40%; AOP, with 80.46% and 82.34%; and RLC-ResNet, with 85.85% and 87.91%, for the cases of NC and NCI, respectively.

The cross-dataset evaluation results of Resnet18 on two different datasets—KinectLeap and Creative Senz3D—are much lower than the cases of training and testing ResNet on one dataset. For KinectLeap, the recognition result of ResNet is 72%, and for Creative Senz3D, the result is 72.04%. This is obvious in classification problems, in which cross-dataset evaluation is much more challenging than using a unified dataset for evaluation. However, when deploying these datasets for our continual learning method RLC-Resnet, the hand gesture recognition results are higher than those of ResNet18. In the case of D_t being KinectLeap and D_{t+1} being Creative Senz3D, the recognition results are 86.58% for the NC type and 88.21% for the NCI type. In the case of D_t being Creative Senz3D and D_{t+1} being KinectLeap, RLC-ResNet obtains 85.85% and 87.91% for NC and NCI, respectively. All

values of recognition accuracy are the averages of five runs with the standard deviation, shown beside each result.

Based on the above analysis of experimental results, we see that the RLC-Resnet method not only outperforms other continual learning methods such as LwF, iCaRL, and AOP but also is extremely higher than the cross-dataset evaluation of ResNet18.

Table 7. The recognition results (%) of the RLC-Resnet method and other solutions on two data protocols of KinectLeap-Creative Senz3D and Creative Senz3D-KinectLeap in the NC and NCI cases with a task size of 16 classes. All values are averaged over five running times, and the standard deviation is shown beside each result.

D_t		KinectLeap	Creative Senz3D	KinectLeap	Creative Senz3D
D_{t+1}		Creative Senz3D	KinectLeap	KinectLeap	Creative Senz3D
Resnet18		30.38 ± 0.2	32.08 ± 0.3	72.00 ± 0.3	72.04 ± 0.2
LwF	NC	29.75 ± 0.3	28.56 ± 0.4	-	-
	NCI	30.25 ± 0.2	30.26 ± 0.6	-	-
iCaRL	NC	80.88 ± 0.4	79.59 ± 0.3	-	-
	NCI	81.74 ± 0.5	81.40 ± 0.6	-	-
AOP	NC	81.67 ± 0.6	80.46 ± 0.3	-	-
	NCI	83.03 ± 0.4	82.34 ± 0.0	-	-
RLC-Resnet	NC	86.58 ± 0.4	85.85 ± 0.3	-	-
	NCI	88.21 ± 0.2	87.91 ± 0.3	-	-

In addition, Table 7 shows that the scenario of NCI incremental data obtains a little higher accuracy than the NC type for a task size of 16 classes. For KinectLeap-Creative Senz3D, the NCI type obtains 30.25%, 81.71%, 83.03%, and 88.21% for the LwF, iCaRL, AOP, and RLC-Resnet methods, respectively. They are approximately 1% higher than those of the NC type. For Creative Senz3D-KinectLeap, the results for NCI types are 30.26% for the LwF method, 81.40% for the iCaRL method, 82.34% for AOP, and 87.91% for the RLC method. The corresponding results for the NC type are 28.56%, 79.59%, 80.46%, and 85.85%.

From the above analysis of the experimental results, we see the efficiency of our RLC-Resnet method in incremental learning on two different datasets: KinectLeap and Creative Senz3D. Although these datasets have some similar labels for hand gestures, they were captured by different camera types at distinctive environments. When deploying these datasets for our RLC-Resnet method in both the NC and NCI types, the performance of hand gesture recognition is remarkably improved in comparison with other methods and even with a one-dataset evaluation of ResNet18. In addition, the higher recognition performance of the NCI type compared with the NC type proves the efficiency of our framework for continual learning, even for the more complex case of NCI. This result also shows the meaning of adding source task samples with similar class labels to the target task into the memory. This helps boost the performance of hand gesture recognition.

5. Conclusions

In this paper, an efficient solution for continual learning is proposed. It takes advantage of the simplicity of the rehearsal-based approach but gains efficiency with two strategies: (1) a random sample selection strategy for replay memory and (2) end-to-end continual learning with a loss combination strategy. The first strategy helps avoid the overfitting phenomenon commonly seen in rehearsal-based methods. The second one is useful for reducing catastrophic forgetting in DNNs. The enhanced experiments are conducted on various data distribution scenarios for standard datasets. Two popular datasets, CIFAR-100 and CORE-50, for lifelong learning are experimented with. Furthermore, two other common datasets of hand gesture recognition are first utilized for evaluating the proposed improvements in continual learning. These are the KinectLeap and Creative Senz3D datasets. In this work, different classification strategies are also tested to prove the outperformance of our proposal compared with the corresponding SOTA methods. Besides the positive results, this work also has some limitations. It requires memory for storing samples at each

step of incremental learning. The optimal size of memory depends on the experimental dataset. As in this work, the reasonable memory size is 4000. In addition, the experiments are only conducted on Resnet18 as the backbone. For further improvement in recognition accuracy, other cutting-edge networks can take its place.

In the future, some enhancements can be applied to the proposed framework. A multi-criteria strategy could be applied for memory reconstruction in continual learning. The random-sample memory can be enhanced with additional criteria such as the intra-class variation in source task and target task or the distance to the prototype. In order to apply the proposed framework for hand gesture recognition and other fields in the future, more experiments on various datasets of different fields should be implemented.

Author Contributions: Conceptualization, H.-G.D. and T.T.T.P.; data curation, T.-O.H. and H.-Q.L.; formal analysis, H.-G.D. and T.T.T.P.; funding acquisition, H.-Q.L.; investigation, T.-O.H.; methodology, H.-G.D. and T.T.T.P.; project administration, H.-G.D. and T.T.T.P.; software, H.-G.D. and T.T.T.P.; supervision, T.T.T.P.; validation, T.T.T.P.; visualization, H.-G.D.; writing—original draft, H.-G.D.; writing—review and editing, T.T.T.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by research project of Viet Nam government under grant number KC-4.0.28/19-25.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AOP	Adaptive Orthogonal Projection
DNNs	Deep Neural networks
NC	New Class
NCI	New Class and new Instance
SVM	Support Vector Machine
NB	Naive Bayes
DT	Decision Tree
CRF	Conditional Random Field
GEM	Gradient Episodic Memory
LwF	Learning without Forgetting
DER	Dark Experience Replay
KNN	k-Nearest Neighbors
CNN	Convolutional Neural Network
FC	Fully Connected layer

References

- McCloskey, M.; Cohen, N.J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*; Elsevier: Amsterdam, The Netherlands; Academic Press: Cambridge, MA, USA, 1989; Volume 24, pp. 109–165. [[CrossRef](#)]
- Caruana, A. Corporate reputation: Concept and measurement. *J. Prod. Brand Manag.* **1997**, *6*, 109–118. [[CrossRef](#)]
- Abraham, W.C.; Robins, A. Memory retention—the synaptic stability versus plasticity dilemma. *Trends Neurosci.* **2005**, *28*, 73–78. [[CrossRef](#)]
- Richardson, F.M.; Thomas, M.S. Critical periods and catastrophic interference effects in the development of self-organizing feature maps. *Dev. Sci.* **2008**, *11*, 371–389. [[CrossRef](#)]
- Mesnil, G.; Dauphin, Y.; Glorot, X.; Rifai, S.; Bengio, Y.; Goodfellow, I.; Lavoie, E.; Muller, X.; Desjardins, G.; Warde-Farley, D.; et al. Unsupervised and transfer learning challenge: A deep learning approach. In Proceedings of the ICML Workshop on Unsupervised and Transfer Learning, Edinburgh, UK, 27 June 2012; JMLR Workshop and Conference Proceedings. pp. 97–110.
- Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; JMLR Workshop and Conference Proceedings; Volume 32, pp. 647–655.
- Li, Z.; Hoiem, D. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 2935–2947. [[CrossRef](#)]

8. Parisi, G.I.; Tani, J.; Weber, C.; Wermter, S. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Front. Neurobot.* **2018**, *12*, 78. [[CrossRef](#)]
9. Liu, H.; Liu, T.; Zhang, Z.; Sangaiyah, A.K.; Yang, B.; Li, Y. ARHPE: Asymmetric relation-aware representation learning for head pose estimation in industrial human–computer interaction. *IEEE Trans. Ind. Inform.* **2022**, *18*, 7107–7117. [[CrossRef](#)]
10. Liu, H.; Fang, S.; Zhang, Z.; Li, D.; Lin, K.; Wang, J. MFDNet: Collaborative poses perception and matrix Fisher distribution for head pose estimation. *IEEE Trans. Multimed.* **2021**, *24*, 2449–2460. [[CrossRef](#)]
11. Liu, H.; Nie, H.; Zhang, Z.; Li, Y.F. Anisotropic angle distribution learning for head pose estimation and attention understanding in human–computer interaction. *Neurocomputing* **2021**, *433*, 310–322. [[CrossRef](#)]
12. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
13. Truong, D.M.; Giang, D.; Tran, T.H.; Hai, V.; Le, T. Robustness Analysis of 3D Convolutional Neural Network for Human Hand Gesture Recognition. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 135–142. [[CrossRef](#)]
14. Doan, H.-G.; Nguyen, N.T. End-to-end multiple modals deep learning system for hand posture recognition. *Indones. J. Electr. Eng. Comput. Sci.* **2022**, *27*, 214–221. [[CrossRef](#)]
15. Molchanov, P.; Gupta, S.; Kim, K.; Kautz, J. Hand gesture recognition with 3D convolutional neural networks. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Boston, MA, USA, 7–12 June 2015; pp. 1–7. [[CrossRef](#)]
16. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **2019**, *113*, 54–71. [[CrossRef](#)] [[PubMed](#)]
17. Mirzadeh, S.I.; Farajtabar, M.; Gorur, D.; Pascanu, R.; Ghasemzadeh, H. Linear Mode Connectivity in Multitask and Continual Learning. In Proceedings of the International Conference on Learning Representations, Virtual Event, Austria, 3–7 May 2021.
18. Kudithipudi, D.; Aguilar-Simon, M.; Babb, J.; Bazhenov, M.; Blackiston, D.; Bongard, J.; Brna, A.; Chakravarthi Raja, S.; Cheney, N.; Clune, J.; et al. Biological underpinnings for lifelong learning machines. *Nat. Mach. Intell.* **2022**, *4*, 196–210. [[CrossRef](#)]
19. Douillard, A.; Chen, Y.; Dapogny, A.; Cord, M. PLOP: Learning without Forgetting for Continual Semantic Segmentation. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 4039–4049.
20. Saporta, A.; Douillard, A.; Vu, T.H.; P’erez, P.; Cord, M. Multi-Head Distillation for Continual Unsupervised Domain Adaptation in Semantic Segmentation. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 19–20 June 2022; pp. 3750–3759.
21. Jin, Y.; Jiang, D.; Cai, M. 3d reconstruction using deep learning: A survey. *Commun. Inf. Syst.* **2020**, *20*, 389–413. [[CrossRef](#)]
22. Rebuffi, S.A.; Kolesnikov, A.; Sperl, G.; Lampert, C. iCaRL: Incremental Classifier and Representation Learning. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5533–5542. [[CrossRef](#)]
23. Maltoni, D.; Lomonaco, V. Continuous learning in single-incremental-task scenarios. *Neural Netw.* **2019**, *116*, 56–73. [[CrossRef](#)]
24. Carta, A.; Cossu, A.; Lomonaco, V.; Bacciu, D. Ex-Model: Continual Learning from a Stream of Trained Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 21–24 June 2022; pp. 3790–3799.
25. Guo, Y.; Liu, B.; Zhao, D. Online Continual Learning through Mutual Information Maximization. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; Proceedings of Machine Learning Research; Volume 162, pp. 8109–8126.
26. Lopez-Paz, D.; Ranzato, M. Gradient episodic memory for continual learning. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 6468–6477.
27. Chaudhry, A.; Ranzato, M.; Rohrbach, M.; Elhoseiny, M. Efficient Lifelong Learning with A-GEM. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
28. Zhu, F.; Cheng, Z.; Zhang, X.y.; Liu, C.l. Class-Incremental Learning via Dual Augmentation. In Proceedings of the 35th Conference on Neural Information Processing Systems, Virtual-only Conference, 6–14 December 2021; Volume 34, pp. 14306–14318.
29. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [[CrossRef](#)]
30. Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D.; Calderara, S. Dark experience for general continual learning: A strong, simple baseline. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 15920–15930.
31. Zenke, F.; Poole, B.; Ganguli, S. Continual Learning through Synaptic Intelligence. In Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 3987–3995.
32. Gao, Q.; Luo, Z.; Klabjan, D.; Zhang, F. Efficient architecture search for continual learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–11. [[CrossRef](#)]
33. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
34. Krizhevsky, A. Learning multiple layers of features from tiny images. In *Technical Report*; University of Toronto: Toronto, ON, Canada, 2009.

35. Lomonaco, V.; Maltoni, D. CORE50: A New Dataset and Benchmark for Continuous Object Recognition. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; Levine, S., Vanhoucke, V., Goldberg, K., Eds.; PMLR—Proceedings of Machine Learning Research. Volume 78, pp. 17–26.
36. Marin, G.; Dominio, F.; Zanuttigh, P. Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. *Multimed. Tools Appl.* **2015**, *75*, 14991–15015. [[CrossRef](#)]
37. Memo, A.; Zanuttigh, P. Head-mounted gesture controlled interface for human-computer interaction. *Multimed. Tools Appl.* **2017**, *77*, 27–53. [[CrossRef](#)]
38. Guo, Y.; Hu, W.; Zhao, D.; Liu, B. Adaptive Orthogonal Projection for Batch and Online Continual Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual Event, 22 February–1 March 2022.
39. French, R.M. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* **1999**, *3*, 128–135. [[CrossRef](#)]
40. Liu, H.; Zheng, C.; Li, D.; Shen, X.; Lin, K.; Wang, J.; Zhang, Z.; Zhang, Z.; Xiong, N.N. EDMF: Efficient deep matrix factorization with review feature learning for industrial recommender system. *IEEE Trans. Ind. Inform.* **2021**, *18*, 4361–4371. [[CrossRef](#)]
41. Liu, T.; Yang, B.; Liu, H.; Ju, J.; Tang, J.; Subramanian, S.; Zhang, Z. GMDL: Toward precise head pose estimation via Gaussian mixed distribution learning for students’ attention understanding. *Infrared Phys. Technol.* **2022**, *122*, 104099. [[CrossRef](#)]
42. Liu, H.; Liu, T.; Chen, Y.; Zhang, Z.; Li, Y.F. EHPE: Skeleton cues-based gaussian coordinate encoding for efficient human pose estimation. *IEEE Trans. Multimed.* **2022**, 1–12. [[CrossRef](#)]
43. Prabhu, A.; Torr, P.H.S.; Dokania, P.K. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In Proceedings of the European Conference on Computer Vision, Online Event, 23–28 August 2020.
44. Aljundi, R.; Lin, M.; Goujaud, B.; Bengio, Y. Gradient based sample selection for online continual learning. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–12 December 2019.
45. Aljundi, R.; Belilovsky, E.; Tuytelaars, T.; Charlin, L.; Caccia, M.; Lin, M.; Page-Caccia, L. Online Continual Learning with Maximal Interfered Retrieval. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Sydney, NSW, Australia, 2019; pp. 11849–11860.
46. Shim, D.; Mai, Z.; Jeong, J.; Sanner, S.; Kim, H.J.; Jang, J. Online Class-Incremental Continual Learning with Adversarial Shapley Value. *Proc. Aaai Conf. Artif. Intell.* **2021**, *35*, 9630–9638. [[CrossRef](#)]
47. Mai, Z.; Li, R.; Kim, H.J.; Sanner, S. Supervised Contrastive Replay: Revisiting the Nearest Class Mean Classifier in Online Class-Incremental Continual Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 21–24 June 2021; pp. 3589–3599.
48. Chaudhry, A.; Khan, N.; Dokania, P.; Torr, P. Continual Learning in Low-rank Orthogonal Subspaces. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, 6–12 December 2020.
49. Cha, H.; Lee, J.; Shin, J. Co2L: Contrastive Continual Learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Montreal, BC, Canada, 11–17 October 2021; pp. 9516–9525.
50. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
51. Wah, C.; Branson, S.; Welinder, P.; Perona, P.; Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. In *Computation & Neural Systems Technical Report, 2010-001*; California Institute of Technology: Pasadena, CA, USA, 2011.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.