On the benecial effects of reinjections for continual learning

Miguel Solinas, Marina Reyboz, Stephane Rousset, Julie Galliere, Marion Mainsant, Yannick Bourrier, Anca Molnos, Martial Mermillod

On the beneficial effects of reinjections for continual learning

Solinas Miguel[1*], Reyboz Marina[1*], Rousset Stephane[2*], Galliere Julie[1*], Mainsant Marion[1*], Bourrier Yannick[2*], Molnos Anca[1*] and Mermillod Martial[2*]

[1*]Univ. Grenoble Alpes, CEA, LIST, 17 Av. des Martyrs, Grenoble, 38000, France.
[2*]Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LPNC, 1251 Av. Centrale, Saint-Martin-d'Hères, 38400, France.

*Corresponding author(s). E-mail(s): migue.solinas@gmail.com; marina.reyboz@cea.fr;

Abstract

Deep learning delivers remarkable results in a wide range of applications, but artificial neural networks still suffer from catastrophic forgetting of old knowledge as new knowledge is learned. Rehearsal methods overcome catastrophic forgetting by replaying an amount of previously learned data stored in dedicated memory buffers. Alternatively, pseudo-rehearsal methods generate pseudo-samples to emulate previously learned data, alleviating the need for dedicated buffers. First, we show that it is possible to alleviate catastrophic forgetting with a pseudo-rehearsal method without employing memory buffers or generative models. We propose a hybrid architecture similar to that of an autoencoder with additional neurons to classify the input. This architecture preserves specific properties of autoencoders by allowing the generation of pseudo-samples through reinjections (i.e. iterative sampling) from random noise. The generated pseudo-samples are then interwoven with the new examples to acquire new knowledge without forgetting the previous ones. Second, we combine the two methods (rehearsal and pseudo-rehearsal) in the hybrid architecture. Examples stored in small memory buffers are employed as seeds instead of noise to improve the process of generating pseudo-samples and retrieving previously learned knowledge. We demonstrate that reinjections are suitable for rehearsal and pseudo-rehearsal approaches and show state-of-the-art results on rehearsal

methods for small buffer sizes. We evaluate our method extensively on MNIST, CIFAR-10 and CIFAR-100 image classification datasets.

Keywords: incremental learning, lifelong learning, continual learning, sequential learning, pseudo-rehearsal, rehearsal

1 Introduction

Deep learning has yielded remarkable results in many applications; however, artificial neural networks still suffer from catastrophic forgetting of old knowledge as new information is learned [1–5]. Catastrophic forgetting prevents artificial neural networks from acquiring new skills and evolving in non-stationary environments (e.g. learning to classify a whole dataset class by class instead of learning all classes together). Researchers have been studying this problem since the 90s [6] by looking at the brain's neurogenesis [7], synaptic consolidation [8] and replay systems (hippocampal-neocortex network) [9]. Indeed, modeling real continual learning (CL) as we humans do requires finding appropriate solutions to this problem.

First, neurogenesis methods evolve the neural network architecture to adapt to different training experiences using independent sets of parameters [10–17]. Second, synaptic consolidation methods limit changes in important parameters of previously learned tasks [18–25]. Thus new tasks will employ neurons that are less useful for previous tasks.

It is possible to replay previously learned examples in two ways, with real samples (rehearsal) or with synthetic samples (pseudo-rehearsal). Rehearsal methods overcome catastrophic forgetting by replaying an amount from previously learned examples stored in dedicated memory buffers [26–35, 35]. Rehearsal methods replaying only a fraction of old samples have recently been found to be one of the best solutions to alleviate the catastrophic forgetting problem [3, 36, 37] due to their ability to successively integrate new memories [3] and to their superior performance compared to other CL methods given a similar amount of computational resources. Surprisingly, they still work well when replying only a tiny fraction of the previous samples [36], which we denote as *tiny memory buffers*. The small memory footprint of these solutions justifies their eligibility for large scale datasets benchmarks. However, one of the biggest problems of using small memory buffers is that old knowledge is sub-optimally represented. The latter makes the parameters of the models more adjusted to the new tasks. In the context of classification problems, a task describes categorizing a given set of data into classes. This phenomenon occurs because the model has seen many different samples of the last task but few from previous tasks [4, 26–28, 31, 32, 34, 38, 39]. As a result, the previously learned tasks tend to be biased or misclassified into the most recently learned tasks.

Alternatively, pseudo-rehearsal methods generate pseudo-samples that may emulate previously learned data [40–47]. They were conceived to avoid the utilization and storage of previously learned samples. Instead of replaying past training data from buffers, a complementary learning system approximates previous examples through another ANN (e.g. a generative neural network). This second ANN generates pseudo-samples that , together with the new samples, become inputs during the incremental training. The term *pseudo* denotes the fact that the samples representing previous knowledge are often artificially generated by employing a sampling procedure and random noise.

Replaying what has been previously learned through examples or pseudo-samples while learning new tasks allows adapting the global set of parameters for past and new tasks —-overcoming catastrophic forgetting similarly to classical deep learning training when the entire dataset is present [48]. Thus, the biggest challenge of replay methods is to represent correctly and globally what has been previously learned [48], as they often rely on limited memory buffers or roughly generative models.

In this work, we focus on incremental classification problems. We propose a hybrid architecture that retrieve its knowledge by means of samples generated through reinjections (i.e. iterative sampling [49, 50]) and allows for continual learning. Reinjections consists in injecting an input sample in a replicator ANN (e.g. an autoencoder) and in *reinjecting* its output multiple times until a stop condition is reached. The hybrid architecture consists of an autoencoder with additional neurons in the output layer to perform the classification (i.e. auto-hetero encoder or auto-hetero associative neural network). The interesting feature about this architecture is that it preserves the ability of the autoencoder to memorize what it has previously learned [49, 51]. In addition, the model generates pseudo-samples with their corresponding labels through the reinjection mechanism. The obtained input-output pair is called knowledge [50]. The goal is to retrieve the knowledge represented by the learned classification function (i.e. the mapping function) to alleviate forgetting of earlier tasks. More specifically, pseudo-samples are replayed by the model when it learns a new task.

The proposed generic continual learning framework can be deployed for several applications. First, the pseudo-samples (i.e. synthetic data) are generated with random noise and reinjections for retrieving previous knowledge. Since the knowledge retrieval process does not employ a memory buffer, it is a Data-free solution. This work exploits the Data-free approach to validate that hybrid architectures perform continual learning. However, as shown in previous findings [4, 46], the Data-free solution struggles to scale on larger datasets. Second, the Data-free solution is equipped with a memory buffer to increase the efficiency of the acquired knowledge retrieval process. Instead of using random noise in the reinjections, examples are used from tiny memory buffers, which allows the hybrid model to cope with larger data sets. More specifically, the hybrid model generates variations of the examples from the tiny memory

buffers through reinjections. During training, these pseudo-samples are interleaved with real samples of a new set of classes to incrementally integrate new knowledge while consolidating the previous one. We show that the hybrid model alleviates catastrophic forgetting and incrementally solves classification problems without memory buffers or generative models. Next, we show that the hybrid buffered model enables state-of-the-art results. Since the proposed solution combines the central ideas of the rehearsal and pseudo-rehearsal methods, it is called Combined replay.

During the incremental learning step, the hybrid model minimizes a classification loss to integrate the new task and minimizes a distillation loss to consolidate the previous tasks. The distillation loss exploits the pseudo-samples obtained through reinjections. Overall, we show that the reinjection sampling procedure in this hybrid architecture alleviates catastrophic forgetting with pseudo-samples from Gaussian noise or real examples. In particular, Combined replay improves the knowledge retrieval process and provides results comparable to state-of-the-art approaches. We also show that Combined replay suffers from bias problems like most rehearsal methods and that this can be alleviated by training with larger learning batches for old tasks. To evaluate the impact of the use of the proposed pseudo-samples vs the real-samples, we compare the performance of existing replay models such as ICARL [26] and Tiny Episodic Memory Replay (ER) [52] with our method (Combined replay). The experiments focus on continual learning scenarios applied to classification tasks. The Data-free solution is validated and then Combined replay is evaluated using the classification accuracy in the following datasets: MNIST [53] , CIFAR-10 [54] and CIFAR-100 [54].

This work is structured as follows: Related work is presented in Section 2. The background on which we build our continual learning approaches is presented in Section 3. The evaluation and results of Data-free experiments are presented in subsection 4.4. The evaluation and results of Combined replay experiments are presented in subsection 4.5. Our findings are discussed in Section 5. Finally, the conclusion and the perspectives are drawn in Section 6.

2 RELATED WORK

Catastrophic forgetting was formalized for the first time by [6]. Between 1990 and 2010, catastrophic forgetting was initially studied by the cognitive science community [55, 56]. Those pioneer works gave raise not only to actual terminologies as rehearsal, pseudo-rehearsal [40] and complementary learning system [57] but also to the solutions based on knowledge distillation [40], dual networks [58] and reduced overlap representations [59]. These solutions were mainly focused on shallow ANNs architectures trained on low dimensional datasets learned with few incremental learning steps.

Nowadays, catastrophic forgetting is one of the most challenging problems when working with data streams in dynamic environments and real-world scenarios. In these cases, ANNs learn to perform a task (e.g. the process of

categorizing a given set of data into classes) by finding an "optimal" point in the parameter-space. When ANNs subsequently learn a new task (e.g. the process of categorizing a new set of data into a new class), their parameters will move to a new solution point that allows the ANNs to perform the new task. Catastrophic forgetting [6] arises when the new set of parameters is completely inappropriate for the previously learned tasks. The latter is mainly a consequence of the gradient descent algorithm that is typically used to find the ANN parameters during training. Indeed, all ANN parameters are adapted for the new task without taking into account previous knowledge. Catastrophic forgetting is related to the stability-plasticity dilemma [60], which is a more general problem in neural networks, due to the fact that learning models require both: plasticity to learn new knowledge and stability to prevent the forgetting of previously learned knowledge. The objective, in CL, is to overcome the catastrophic forgetting problem by looking for a trade-off between stability and plasticity.

The recent development of deep neural networks has led to a great deal of interest in this field, which is now being addressed as continual learning [2, 42], sequential learning [6, 61], lifelong learning [62–64] and incremental learning [22, 26]. These fields aim at learning new information from a continuous stream of data without erasing previous knowledge (i.e. the performance on previously learned tasks must not be degraded significantly over time as new tasks are learned). For clarity, we simplify the terminology by referring to these fields as continual learning. Continual learning state-of-the-art approaches might be divided into three paradigms [3]: regularization-based, parameter isolation and replay methods. Figure 1 gives a brief overview of the CL methods regarding their plasticity-stability abilities.

Regularization approaches are often associated with synaptic consolidation in the brain, in which the simultaneous activation of cells leads to a pronounced increase in synaptic strength between those cells (i.e. the more an activation pathway is used in a neural network, the more stable the circuit becomes). Synaptic consolidation enables continual learning and allows for the consolidation of previously learned information because vital synapses of previously learned experiences are maintained [65]. In this way, knowledge of previously acquired information is durably encoded in a portion of synapses stabilized over long time scales.

In continual learning, regularization-based approaches introduce an additional regularization term in the classification loss function to maintain previous knowledge. The regularization term encourages the mapping function for the new task not to deviate too far from the mapping function of the previous task. Depending on how the regularization term is implemented, it is possible to distinguish between weight regularization and function regularization approaches.

The main idea of function regularization approaches is to limit the drift of the learned mapping function from previously learned tasks while learning new tasks. Thus, this approach maintains the mapping function from

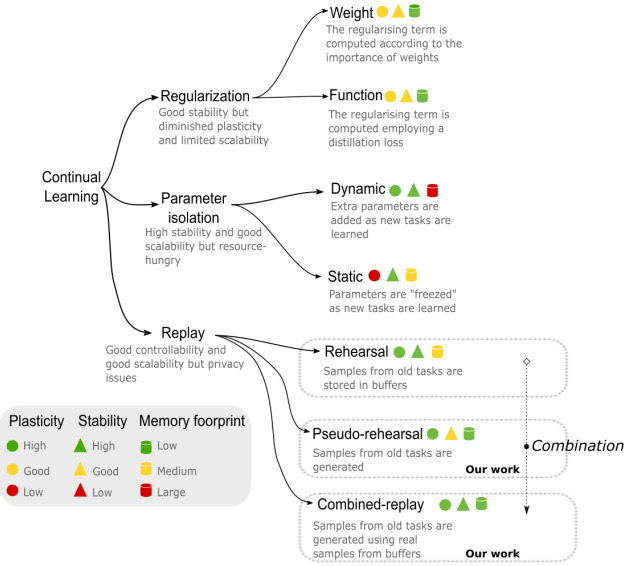

Fig. 1 Continual learning methods indexed regarding their plasticity-stability ability.

inputs to outputs (i.e. knowledge) of previous tasks by distilling the knowledge [18, 26, 27, 62, 66, 67]. Weight regularization approaches introduce an extra regularization term in the loss function [19–21, 68]. The regularization term can be computed in an online [21, 69] or offline [19, 69] fashion. It is implemented locally at each synapse by penalizing important changes in the weights which were particularly influential in the past. In terms of storage, weight and function regularization-based methods are not demanding because they do not require memory buffers or a second ANN to maintain previous knowledge. Nevertheless, a buffer can be used to alleviate forgetting further [26, 27, 30]. When many tasks must be performed, the penalty introduced to increase stability might not be sufficient to overcome catastrophic forgetting as shown in previously published experiments [3, 70]. Thus, the major limitation of regularization methods is to not efficiently compute the regularization term that allows to preserve previous knowledge. Moreover, the main risk of these approaches is to trade plasticity for stability. The plasticity is limited when the parameters of ANNs are "frozen" to maintain previous knowledge through the regularization term.

Parameter isolation approaches are often associated with neurogenesis in the brain. Neurogenesis involves the growth of new neurons to assimilate and consolidate new information while learning new patterns. In this way, new information is associated with a specific set of neurons that specialize in an input stimulus [71]. Recent studies suggest that neurogenesis in the brain declines sharply in childhood and it is undetectable in adulthood [72, 73]. While those work may suggest that lifelong neurogenesis would not be the main avenue for memory consolidation, several approaches in the continual

learning literature mimic to some extent neurogenesis in the brain. Notably, recent work argues the contrary, neurogenesis is maintained during ageing [74].

Parameter isolation methods aim to preserve and maintain previously learned knowledge in a specific set of parameters. The previously learned task-specific parameters are frozen at each incremental step while a new set of parameters is adapted only for the new task. In this way, the specific parameters of the previous task remain unchanged. These methods can be sub-classified into dynamic [13–17] or static [10–12, 75, 76]. Dynamic architectures add new parameters to the architecture of an ANN for each learned task. They are stable enough in a system comprising large memory resources and where high performance is the priority. Static architectures gradually reduce the model plasticity by "freezing" a set of parameters for each new learning task. Regarding the memory footprint, deep and large models are often needed to extend the number of tasks that can be learned. However, they only partially circumvent the catastrophic forgetting problem since new architecture is added/employed to learn new tasks. Thus, parameter isolation methods are not designed to overcome catastrophic forgetting but to dodge it. Indeed, there is no model with global structural plasticity for any of the tasks learned so far, but small specific blocks for each learned task [13, 62, 77, 78]

Replay methods are often related to the theory of complementary learning systems (CLS) [57] which involve two neural networks, the hippocampus and the neocortex, to explain consolidation in the brain. At the brain level, recent studies support that the hippocampal system exhibits short-term adaptation that allows for rapid integration of new information [9]. More specifically, during the sleep phase, the hippocampal system plays back recent memories to the neocortical system for long-term retention and consolidation. Thus, the hippocampal system is considered a fast learner, integrating new patterns without disrupting the structure and minimizing interference. Complementarily, the neocortical system is seen as a slow learner whose overlapping representations of new and old patterns are consolidated for continual learning. Such a system suggests that the neocortex slowly uncovers the hidden structure of experiences, which is crucial for finding regularities and specificities. In this context, we use the term "consolidation" to describe the post-experience memory stabilization processes that occur when memories are replayed.

In continual learning, replay-based methods encourage models to consider previous knowledge by revisiting previously seen examples or approximation of them. Replaying old examples when learning new ones encourages the mapping function of new tasks to include the mapping function of previous tasks. It acts as a spring that prevents the model from deviating too much from its correct behaviour such as regularization loss in regularization-based methods. Replay methods exploit the inner plasticity of ANNs by rehearsing old knowledge when learning new tasks instead of diminishing this ability. Replay methods can be sub-classified into rehearsal [3, 30, 36, 37, 79–83] and pseudo-rehearsal [40–43, 46, 47, 84–86].

Rehearsal methods [26, 27, 36] explicitly retrain on a subset of stored samples from previous tasks and the performance is usually constrained by a fixed memory budget. The larger the memory buffer, the greater the stability, so the lesser the forgetting. The parameters controlling the stability of old knowledge is usually determined by the size of the memory buffer used to store the old samples and how the old examples are exploited [87, 88]. The usual way to exploit the memory buffers is to train the models on a new task along with old samples from tiny buffers [36, 37]. However, the buffer size to store old data and the way the data are used vary with each rehearsal CL implementation. For example, ICARL [26] is a double-memory system that employs a memory buffer size of 2000 samples and a second model to retrieve previously learned knowledge. The captured knowledge is replayed when learning a new task.

Pseudo-rehearsal methods have been recently improved with the development of powerful generative models capable of modeling complex data distributions such as generative adversarial networks [89] and variational autoencoders [90]. The performance of pseudo-rehearsal methods rely on both the generative power and the quality of the synthetic data set provided by the generative model. In fact, these two characteristics play a key role in the stability of previously learned knowledge. Pseudo-rehearsal methods are often outperformed by rehearsal methods when many tasks must be learned. Thus, the main challenge facing pseudo-rehearsal methods is to be stable enough to produce optimal pseudo-data as the ANN continuously learns a growing number of tasks. Among the generative models, auto-associative neural networks (i.e. autoencoders) are often employed to generate samples from previous tasks [45, 46, 91–93]. In these works, ancestral sampling is performed to generate samples from the latent space of autoencoders. Alternatively, the approach in [49, 50] differs from this research area because it does not sample from the latent space of the autoencoder but from the input space. Their work generates pseudo-samples from the input space by performing a reinjection sampling procedure (i.e. iterative sampling).

This paper presents a dual memory framework for continual learning based on a hybrid architecture that performs iterative sampling to capture knowledge. First, a Data-free solution using iterative sampling and random noise is proposed for privacy-preserving applications. Second, a Combined replay solution using iterative sampling and examples from small memory buffers is proposed to improve the knowledge capture process. We show that the generated pseudo-samples improve the retrieval process of previously acquired knowledge and compare it with state-of-the-art approaches. The dual memory system adopted in this work is presented in the following section.

3 SET-UP

This study draws inspiration from some previously proposed continual learning approaches employing two artificial neural networks (ANNs) as in [26, 41, 42], which, for consolidation, rely on distillation.

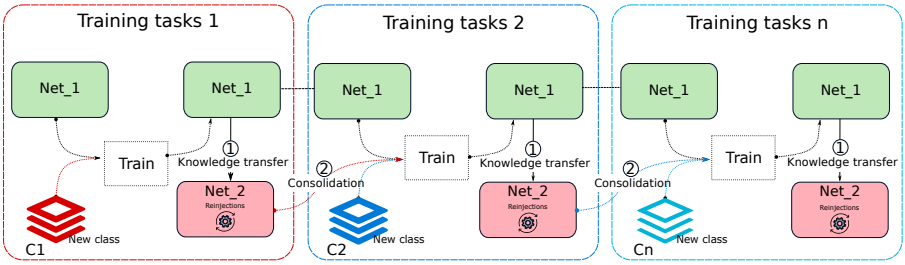

Fig. 2 Dual-memory system. Knowledge transfer: Net_2 acquires Net_1 knowledge by learning the pseudo-samples generated by Net_1. Consolidation: Net_1 searches for a parameter set for new tasks and old tasks by replaying pseudo-samples from the previously learned tasks.

Figure 2 illustrates the two ANNs and the two learning phases of our approach. During the first learning phase ①, the knowledge from the first ANN, named Net_1, is "transferred" to the second ANN, named Net_2, through pseudo-samples. That is, Net_2 is trained with the knowledge of Net_1, which is the model used to generate a pseudo-dataset and that represents the knowledge we want to transfer. As both ANNs are identical, we use an easy way to transfer knowledge. We duplicate the parameters of Net_1 into Net_2 instead of using pseudo-samples in phase ①. It should be noted that if we need to expand the neural network, we can use the pseudo-samples for the knowledge transfer procedure. During the second learning phase ②, new classes have to be integrated without degrading previously learned knowledge. Net_1 learns the new classes as well as the pseudo dataset generated by Net_2.

In this section, we present the ANN architecture employed in the dual-memory system of Figure 2, the sampling procedure used to generate pseudo-samples, the knowledge transfer procedure that employs distillation to transfer the knowledge from one ANN to another and the incremental learning procedure. In the subsection 3.5 we explained in detail the consolidation step ② and the mechanism for generating pseudo-samples for consolidating previous tasks.

Since the dual-memory system described above consists of two identical ANNs, the description that follows is of a single ANN. The employed hybrid architecture is formally called *Auto-Hetero* (AH) associative ANN because it is trained with a two-fold aim: replication and classification. The first aim is referred to as "replication", where for an input x_i, the goal is to output a $\hat{x}_i$ as close as possible to the input x_i. The second aim is referred to as "classification", where for the input x_i, the goal is to output a label $\hat{y}_i$ as close as possible to the ground-truth label y_i. Let us note that for a dataset D with C classes, x_i represents the ith sample and y_i represents the ith label. The ground-truth label y_i is a one-hot C-dimensional vector and $\hat{y}_i$ is a C-dimensional vector whose values are in between 0 and 1. The samples x_i and $\hat{x}_i$ are F-dimensional vectors also in between 0 and 1.The notation $[.,.]$ refers to the concatenation of two vectors. For example, $[x_i, y_i]$ is the P-dimensional

vector ($P = F + C$) that concatenates the F-dimensional vector x_i and the C-dimensional vector y_i.

3.1 The auto-hetero associative architecture

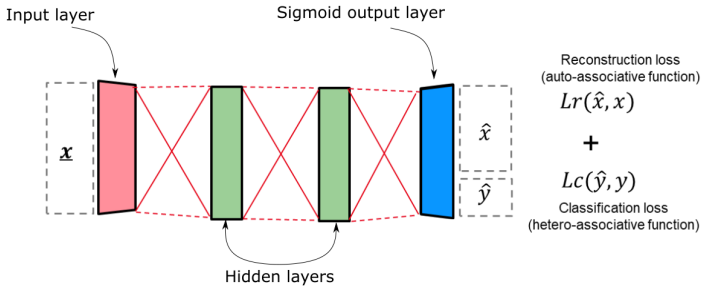

Fig. 3 Auto-Hetero associative architecture.

The architecture of the AH associative ANN comprises an input layer which receives inputs x_i, hidden layers which transform x_i from the input layer and an output sigmoid layer that delivers the P-dimensional vector ($[x_i, y_i]$). An example of our AH associative ANN architecture is presented in Figure 3. The proposed architecture fulfills three main procedures: the training, the inference and the generation of pseudo-samples.

The first procedure, the training, is performed by minimizing the binary cross-entropy loss between the output of the AH network $[\hat{x}_i, \hat{y}_i]$ and the ground-truth outputs $[x_i, y_i]$ using gradient descent. Equation (1) defines this binary cross-entropy loss.

$$\ell_{AH} = -\sum_{(x_i,y_i)\in D} \ell_(\,[x_i, y_i]_p, [\hat{x}_i, \hat{y}_i]_p) \tag{1a}$$

$$= -\sum_{(x_i,y_i)\in D}$$

$$\Big[\frac{1}{F}\sum_{f=0}^{F}\Big([x_i]_f \log([\hat{x}_i]_f) + (1-[x_i]_f)\log(1-[\hat{x}_i]_f)\Big) \tag{1b}$$

$$+$$

$$\Big[\frac{1}{C}\sum_{c=1}^{C}\Big([y_i]_c \log([\hat{y}_i]_c) + (1-[y_i]_c)\log(1-[\hat{y}_i]_c)\Big)\Big] \tag{1c}$$

where P is the dimension of the output of the neural network, $[x_i, y_i]_p$ represents the pth element of the P-dimensional ground truth vector $[x_i, y_i]$ and $[\hat{x}_i, \hat{y}_i]_p$ represents the pth element of the P-dimensional predicted vector

$[\hat{x}_i, \hat{y}_i]$. F is the output dimension for replication and C is the output dimension for classification. We split the output of dimension P into two vectors, one for replication loss ($[x_i]_f$) (Equation (1b)) and the other for classification loss ($[x_i]_c$) (Equation (1c)). In this way, the AH architecture is a hybrid model that performs classification and replication.

The second procedure, the inference, employs the knowledge gained during the training to infer the replication and the label of a given input. While the auto-associative output indicates how well the model is capable of reproducing a given input, the hetero-associative output indicates how well the model has built the decision boundaries for classification. Finally, the generalization ability (i.e. classification on unobserved data) of the model is always measured only by taking into consideration the hetero-associative output for the classification task. That is, the accuracy of the model on the training and testing sets is computed using the classification output. This the pseudo-sample generation procedure is described in the next subsection.

3.2 Reinjection sampling procedure

The pseudo-sample generation procedure, referred to as *reinjection* [58] or iterative sampling [49], employs the Auto-associative component of the AH ANN to perform several inferences.

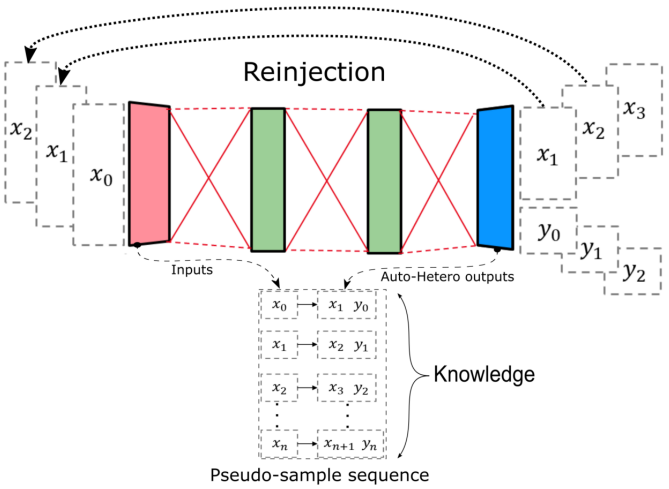

Fig. 4 Reinjection sampling procedure

The reinjection sampling procedure consists in creating a sequence of pseudo-samples with the auto-associative output by following two steps: i. injecting a random sample or an example (x_0) into the input layer of the auto-associative component to infer its replication vector x_1; ii. reinjecting the replication vector x_1 in the input layer to infer the next replication vector x_2. The process of bringing the replication vector in the input layer is illustrated

by the dot arrow in Figure 4 and it is referred to as *reinjection*. Therefore, a sequence of length one consists in $((x_0) \rightarrow [x_1, y_0])$, a sequence of length two consists in $((x_0) \rightarrow [x_1, y_0]\ ;\ (x_1) \rightarrow [x_2, y_1])$ and so on. Note that the hetero-associative output provides only the label of each input sample. For each reinjection, we gather three vectors: the starting point, its corresponding label and the replication of the starting point. The reinjection sampling procedure mimics a non-conditional generative process where the samples are not conditioned by the labels but by the starting point of the generated sequence. After each reinjection, the replication function corresponds, at first order, to a small displacement towards higher densities in the training distribution [49, 50].

In the Data-free implementation, we burn-in the first samples of the beginning of the sequence because the starting points are sampled from an isotropic Gaussian distribution. Therefore, we discard the first point because it is not approximated by the autoencoder to any interesting area.

Alternatively for Combined replay, we do not burn-in (i.e discard iterations) the first samples of the beginning of the sequence because the starting points are samples from the tiny memory buffers instead of random points. Thus, all the generated pseudo-samples are gathered, hence generating a pseudo-sample sequence.

3.3 Knowledge transfer

We employ the terminology introduced in [94] where the knowledge of a trained ANN is defined by the mapping from input vectors to output vectors. This abstract view of the knowledge is free from any particular ANN implementation. For instance, when an ANN classifier infers the soft label of an example, the classifier delivers probabilities for all the classes. The information delivered by the probabilities of all the classes is useful because a new classifier can build similar decision boundaries by learning the real samples and their corresponding soft labels.

The knowledge transfer procedure is called *distillation* and it was originally proposed to transfer the mapping function between different neural networks [40, 41, 94]. Distillation in continual learning is a common practice that ensures that the information previously learned is maintained during a new learning step. In both approaches, Data-free and Combined replay, the set of generated inputs (x_i) and outputs $[x_{i+1}, y_i]$ defines the knowledge of a trained AH ANN (see Figure 4). This knowledge is used to reduce forgetting when learning a new task as it is described in Subsection 3.5.

3.4 Fixed feature extractor

In this work, we use a pre-trained fixed feature extractor to extract features from class images. It should be noted that this is only necessary for images with more than one channel (e.g. CIFAR-10 and CIFAR-100). Instead of learning to replicate and classify high-resolution images, the double memory system receives features that are low-dimensional vectors. A fixed feature extractor

allows us to obtain optimal representations of classes and it helps us to avoid training a feature extractor incrementally as in [14, 17, 46, 50, 78, 91, 95, 96]. We employ a convolutional neural network named Resnet-50 [97] that extracts the feature vectors. These features are transferred to NET_1 which infers the class of the images. As it is illustrated in Figure 2, Net_1 is trained with the features of the new class and pseudo-samples generated by Net_2. Once Net_1 learns the features of the new class, its parameters are duplicated into Net_2 . In this way, the proposed double memory system allows incremental learning without catastrophic forgetting in the fully connected layers benefiting from a fixed feature extractor.

Algorithm 1 Continual learning algorithm

Require: $x^s, ..., x^t$ // training image of classes s,...,t
Require: M // small memory buffer
Require: θ_1 // NET_1 model parameter
Require: θ_2 // NET_2 model parameter

for $u = 0$ to *learning_steps* **do**
  $New\_task \leftarrow \cup_{y=s,...,t} \{ (x_i, y_i) : x_i \in x^y \}$
  if *mode* == 'Data-free' **then**
    $x_{old} \leftarrow \sim \mathcal{N}(0, I)$
  end if
  if *mode* == 'Combined replay' **then**
    $x_{old} \leftarrow \cup_{y=1,...,s-1} \{ (x_i, \cdot) : x_i \in x^y \}$
    $x_{old} = x_{old} + noise\_strength * \mathcal{N}(0, I)$
  end if
  X = $[x_{new}]$
  XY = $[(x_{new}, y_{new})]$
  for $r = 0$ to number of reinjections **do**
    $[x_{n+1}, y_n] \leftarrow AH(\theta_2, x_n)$ for all $(x_i, \cdot) \in$x$_{old}$
    // store pseudo-samples and their outputs
    $X = X \cup [x_n]$
    $XY = XY \cup [(x_{n+1}, y_n)]$
  end for
    // run network training with total loss function
(eq.2 and distillation loss)
    $\theta_1 \leftarrow backprop(X, XY, \theta_1, lr)$
end for

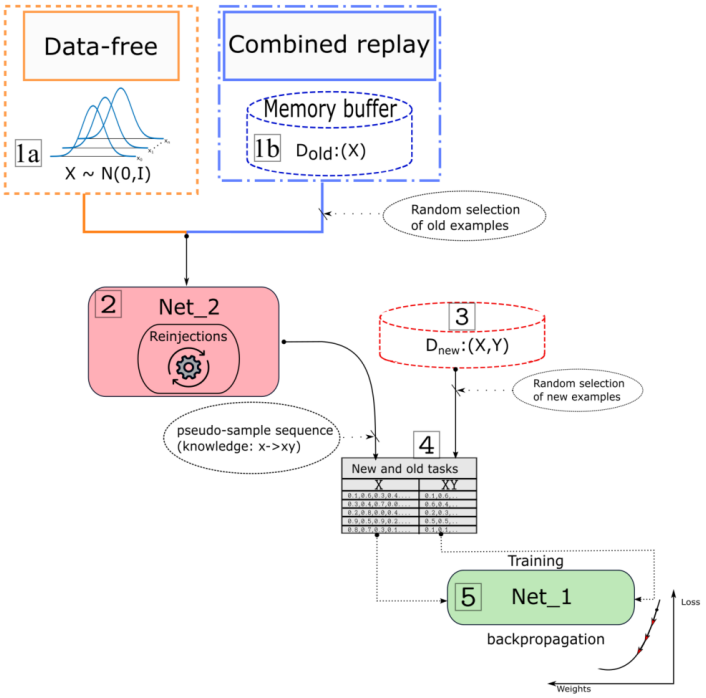

Fig. 5 Data-free and Combined replay workflow

3.5 Consolidation step

In the proposed Data-free and Combined replay solutions, Net_1 learns a new set of classes and its previous knowledge which is captured by Net_2 through reinjections. Algorithm 1 lists the steps behind data-free and combined replay, as shown in Figure 5. We consider that ("initially") Net_2 has already been trained on previous classes and we provide tiny memory buffer and the samples of the new classes.

Data-free:

For Data-free $\boxed{1a}$ (*mode* == 'Data-free' in Algorithm 1), we randomly draw samples from an isotropic Gaussian distribution $x_{old} \sim \mathcal{N}(0, I)$. That is, each value of the sample input vector is drawn from a Gaussian distribution of mean 0 and variance 1 which is classically used to generate synthetic samples [42, 49, 90, 98].

Combined replay:

For Combined replay $\boxed{1b}$ (*mode* == 'Combined replay' in Algorithm 1), we randomly draw samples from the tiny memory buffer x_{old}. Before performing the reinjections, random noise is added to the selected old samples (x_{old}).

Whether they come from, a Gaussian distribution (1*a*) or a memory buffer (1*b*), the samples are reinjected several times to generate the sequence of pseudo-samples 2. That is, Net_2 is evaluated in each reinjection for all the samples in x_{old} by delivering the "auto-hetero" output $[x_{i+1}, y_i]$. Then, we randomly draw samples from the currently available training set New_task 3. The soft-labeled pseudo-samples are merged with the labeled real samples D_{new} resulting in an enhanced dataset 4. Finally, the Auto-Hetero (Net_1) parameters are updated by minimizing the total loss $\ell(\theta_1)$ 5 that encourages to learn the auto-hetero mapping for the new set of classes and to consolidate the auto-hetero output of the previously learned classes (distillation loss). It should be noted that the same total loss is employed to update the Net_1 parameters in the Data-free and Combined replay approaches.

Net_2 retains the previous model parameters which are not updated during this phase. Note that the distillation loss is the same loss formalized above in Equation (1) but the pseudo-samples and their inferred labels (i.e. from Net_2) are employed instead of the true samples and their corresponding ground-truth labels as in Equation 2.

$$\ell_{total} = -\Big(\underbrace{\sum_{(x_i,y_i)\in D_{new}} \ell_{AH}([x_i,y_i]_p, [\hat{x}_i,\hat{y}_i]_p)}_{New\_task\ Loss} + \underbrace{\sum_{(\bar{x}_i,\bar{y}_i)\in Net\_2(X_{seed})} \ell_{AH}([\bar{x}_i,\bar{y}_i]_p, [\hat{x}_i,\hat{y}_i]_p)}_{Distillation\ Loss} \Big) \tag{2}$$

where $(x_i, y_i) \in D_{new}$ are the examples and labels of the new task; $(\bar{x}_i, \bar{y}_i) \in D_{X_{seed}}$ are the synthetic samples and labels generated by Net_2 through reinjections; X_{seed} refers to the fact that it is possible to perform reinjections with Gaussian noise (data-free) or examples from a memory buffer (Combined replay) and $[\hat{x}_i, \hat{y}]$ is the predicted vector of dimension P delivered by Net_1 for a given input (e.g. x_i or $\bar{x}_i$).

Basically, the workflow of Figure 5 is similar to that in [26] where a buffer and a pre-updated classifier are used to perform distillation to capture previous knowledge. There, the samples of the buffer and their distilled outputs are jointly learned with the new samples and their ground-truth labels. The classification loss encourages the classification of the newly observed classes whereas the distillation loss ensures that the previously learned information is not lost. The model architecture and the way the knowledge is captured is different since we do no train a classifier; instead, we train an Auto-Hetero associative ANN and we perform reinjections to capture previous knowledge using random noise or samples from a memory buffer.

In this way, during the consolidation step of Figure 2, two losses are employed to update the parameters of Net_1 through backpropogation. The standard classification and the replication loss for the new samples of

Equation (1) encourage classifying and replicating the new set of classes. The distillation loss for the pseudo-samples and their corresponding soft-labels (logits) ensure that the information previously learned is not lost during the new learning stage.

4 EXPERIMENTS

This section describes the experiments carried out to evaluate the performance of our approaches (Data-free and Combined replay). First, we validate the generality of our approach and some optimization pathways with the Data-free approach. Second, we compare Combined replay with current state-of-the-art replay methods.

4.1 Datasets

We benchmark the beneficial effects of reinjections on three commonly used datasets that differ in the number of classes and features. First, we study the raw images from MNIST. Then, we extract the features from CIFAR-10 and CIFAR-100 using a Resnet50 [97] pre-trained on ImageNet [99] as in Figure **??**. The extracted features are also scaled between 0 and 1 using min-max normalization. It is worth noticing that the maximum accuracy of a classifier trained on the extracted features and their corresponding labels of CIFAR-10 and CIFAR-100 datasets is around 92% and 75% respectively (i.e. offline learning).

The MNIST and CIFAR-10 benchmarks consist of a total of 10 tasks where each task contains one class. For CIFAR-100, we split the original CIFAR-100 dataset into 20 disjoint subsets. Each subset is considered as a separate task and each contains 5 classes from the total of 100 classes. While in the literature dividing CIFAR-100 into 20 tasks with 5 classes is the standard way to benchmark this dataset; instead, we use 100 tasks each containing one class for the Data-free solution. We choose this way to see the limitations of the Data-free approach more clearly in a finer-grained benchmark. Let us note that T represents all the tasks to be learned. In this context, a *task* refers to an isolated training phase defined by (X^t, Y^t) such that X^t is a set of data samples for task t and Y^t the corresponding ground truth labels.

The three datasets, MNIST, CIFAR-10 and CIFAR-100, are complementary datasets widely used in the literature to compare continual learning approaches. These datasets allow studying a model with increasing complexity, from easiest to hardest. MNIST is a dataset with 60.000 samples of 784 features from 10 classes. CIFAR-10 is a dataset with 50.000 samples of 2048 features of 10 classes. The CIFAR-100 is a dataset with 50.000 samples of 2048 features from 100 classes. The main idea of using these datasets is to evaluate the model by increasing the number of features and classes. The larger the number of classes and features, the more difficult it is for the model to retain

old tasks. An additional complexity when evaluating the model with CIFAR-100 is the number of samples per class. For instance, MNIST and CIFAR-10 contain approximately 5000 training samples per class, while CIFAR-100 contains only 500 samples per class. The latter means that when we benchmark the model with all three datasets, we expose our approach to changes in the number of classes, features and available training samples. Thus, the three datasets are optimal choices for validating a continual learning approach.

4.2 Metrics

The performance of all our experiments are measured with a single-head evaluation metric. That is, we do not use a task identifier; instead, we identify the class to which a sample belongs among all the classes learned so far independently. We measure performance on the testing set using accuracy and forgetting, consistently with our domain's literature [22].

Accuracy: Let $a_{k,j} \in [0, 1]$ be the accuracy (fraction of correctly classified data from tasks 1 to k after learning the task j). The higher the value of a_k the better the model performance on the classification task.

$$A_T = \frac{1}{T}\sum_{j=1}^{T} a_{T,j} \tag{3}$$

Forgetting: Let $f_i \in [-1, 1]$ be the forgetting on task i. It measures the gap between the maximum accuracy obtained in the past and the current accuracy about the same task. The lower the forgetting, the better the model performance.

$$F_T = \frac{1}{T-1}\sum_{j=1}^{T-1} (max_{l\in 1,\ldots,i-1}\ a_{l,j}) - a_{i,j} \tag{4}$$

4.3 Architectures

We performed all experiments of the Data-free and Combined replay solutions with the hyperparameters presented in Table 1 (see Table 1). For classifiers and for Auto-Hetero ANNs, we maintain constant the parameters to compare the outcomes of the model under test. We employ the Mish activation function for the hidden layer because it has proved to be more robust than the relu activation function for classification tasks [100]. The models are trained using the adam optimizer [101] with beta1=0.9 and beta2=0.5 and the learning rates of Table 1. When learning CIFAR-100 dataset, we only change two hyperparameters: the epochs and the learning rate. This is due to the fact that the AH architecture needs more learning steps and a smaller learning rate to replicate and to classify CIFAR-100 correctly.

For the Data-free solution the size of the mini-batch for old samples (before reinjections) is set to 256 and for new examples is set to 64 regardless the dataset. Also, we employ 8 epochs per class in the 100 task benchmark (i.e.

Data-free CIFAR-100). The values 64 and 256 were found empirically and they are valid for the 3 benchmarked datasets. In general, the Data-free model needs a large batch of old task samples to consolidate correctly. For Combined replay, the size of the mini-batch of old and new samples is set to 10 irrespective of the memory buffer size. The mini-batch of old samples is copied as described below whenever reinjections are performed.

Table 1 MODEL HYPERPARAMETERS

Models	#units/hidden layer	activ. function	epochs /class	Opt	lr
MNIST					
Auto-Hetero	[784,200,200,794]	Mish	5	Adam	0.0001
Classifier	[784,200,200,10]	Mish	5	Adam	0.0001
CIFAR-10/100					
Auto-Hetero	[2048,1000,1000,2058/2148]	Mish	30	Adam	0.0001
Classifier	[2048,1000,1000,10/100]	Mish	2	Adam	0.001

4.4 EXPERIMENTS: Data-Free approach

The following experiments show that the Data-free solution works and learns classification tasks incrementally without using any information from the external world. They also aim to show that the auto-hetero model can consolidate previous knowledge thanks to the properties of autoencoders and the reinjections.

4.4.1 Baselines

We illustrate the Data-free approach under the following references:

- All of the examples (upper-bound reference): An Auto-Hetero model without reinjections trained with all the old examples of the dataset (i.e. the old examples and their corresponding labels) allows us to observe the maximum performance of our hybrid model when relying on an "infinite" memory buffer. Examples of new and old tasks are employed to minimize a single loss; the binary-cross entropy.
- Without dual memory systems (lower-bound reference): An Auto-Hetero model without reinjections trained only with the examples from the new task allows us to observe the minimum performance of our hybrid model when performing incremental tasks. That is, this baseline does nothing to alleviate catastrophic forgetting.
- Data-free (last R): An Auto-Hetero model that performs 5 reinjections but it only takes the last generated samples to capture the knowledge of Net_2. This experiment allows us to observe the importance of using all reinjections to capture previous knowledge.

- Data-free (0 R): An Auto-Hetero model that performs a single inference with Gaussian noise to capture Net_2 knowledge. It does not burn-in the Gaussian noise, nor does it perform reinjections to improve the samples. This experiment allows us to observe the importance of making reinjections when performing incremental tasks.
- Data-free (2-net) without regularization: An Auto-Hetero model that performs reinjections and that captures the knowledge of Net_2 with the generated sequence. However, during the first learning task, it does not perform reinjections to learn the initial state of Net_2. That is, the Data-free solution always performs reinjections and captures the knowledge of Net_2 even when it learns the first task.
- Data-free (3-Nets): We observed that the Data-free solution alleviates catastrophic forgetting but its effectiveness decreases when many tasks have to be learned incrementally. So, we integrated a third auto-hetero association model into the system (Net_3).

4.4.2 Methodology

The Algorithm 1 is used to compare the performance of the Data-free approach. Only three things are worth noting:

1. When the first task is learned, reinjections are performed on Net_2 to capture its knowledge. The latter is done always and mainly when Net_2 has not learned anything yet. Learning the initial state of Net_2 when Net_1 learns the first task helps to stabilize future learning. If this is not done, Net_1 overlearns the first task, which will never be forgotten because Net_2 generates samples of the first task primarily. To prevent Net_1 from overlearning the first task, we use Net_2 initialization knowledge as a spring from the initial state. That is, Net_2's knowledge prevents Net_1 from straying too far from its initial state and overlearning the first task.
2. When the Data-free approach takes a Gaussian noise batch of 256 samples and performs 5 reinjections, it will produce 1,280 samples generating a batch of significant size. This number of samples is necessary to capture correctly the knowledge and to alleviate forgetting. It is crucial to perform reinjections and to the generated sequence in order to consolidate.
   When learning large datasets such as Cifar-100, the Data-free approach performs poorly with the 2 networks. As our goal is to show the generality of our approach on small and large-scale datasets, we equip the 2 network framework with a third auto-hetero model called Net_3 as it is shown in Figure 6. The reason for adding this third network is related to a possible limitation of data-free when it learns more than fifteen classes incrementally. In this setting, Net_2 will store short-term learned knowledge while Net_3 will store long-term learned knowledge. Net_3 has the same role as Net_2 in the workflow (Figure 5) but it does not immediately maintain the previous state of Net_1; instead, it maintains the earlier states. More precisely, if Net_1 learns task 10, Net_2 contains tasks 1 to 9

and Net_3 will contain tasks 1 to 5. The weights of NET_1 are duplicated into NET_2 after learning each new task and the weights of NET_2 are duplicated into NET_3 every 5 new learning tasks. In this way, NET_3 maintains an early steady state of learning from distant tasks as it is illustrated in Figure 6 (i.e. it is updated sporadically in order to have a stable version of the past). We choose to update Net_3 every 5 learning stages because we have empirically found that it works well; however, weight duplication every 4,5,6,...,15 learning stages also works well. We further discuss about the third neural network in section 5.

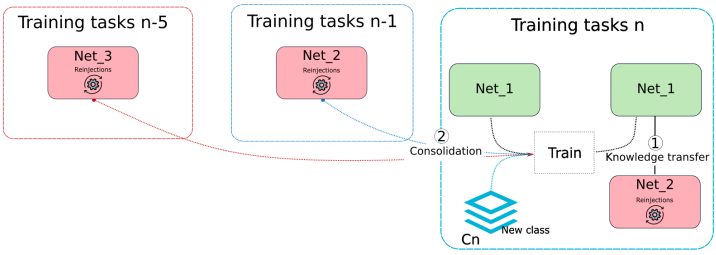

Fig. 6 Dual-memory system. Consolidation: Net_1 searches for a parameter set for new tasks and old tasks by replaying pseudo-samples from Net_2 and Net_3.

4.4.3 Results

Figures 7 (a/b) and Figure 8 summarize the results obtained for the Data-free approach. The following observations can be made.

First, Data-free solution alleviates catastrophic forgetting in all datasets and outperforms the models that do one inference or take the last reinjection to capture Net_2 knowledge (Data-free (0R)/(last R)) as shown in Figure 7(a/b). The Data-free curve in green is far superior to the catastrophic forgetting curve in orange (lower-bound reference); however, it shows a difference of 10% in MNIST, 15% in CIFAR-10 and 51% (2-Nets) in CIFAR-100 with respect to having an infinite buffer (upper-bound reference) plotted by the blue curve. Although the Data-free solution does not achieve a performance similar to one having an infinite buffer, it allows us to obtain an optimal result without storing any information about the dataset. These results show that our dual network framework with auto-hetero associative networks alleviates catastrophic forgetting with the pseudo-samples generated through reinjections and Gaussian noise.

Second, in Figure 7(a/b), the difference observed between the Data-free (0R) curve in black, the Data-free (last R) curve in yellow and the Data-free curve in green underlines the importance of reinjections to improve the samples. (0R) and (last R) Data-free curves are quantitatively comparable since they employ the same amount of pseudo-samples for the consolidation step. However, when the model does not perform reinjections, the Gaussian noise (0R) captures the knowledge of Net_2 less effectively, preventing Net_1

from retaining what it has learned. This highlights the importance of using pseudo-samples close to the learning distribution. The curves Data-free (0R) in Figure 7(a/b) are indistinguishable from the catastrophic forgetting curve in orange (lower-bound reference). Their overlap indicates that this amount of Gaussian noise is insufficient to consolidate previous tasks.
As already shown in [49, 50], reinjections have the property of displacing the sample towards areas of high example density. Therefore, through reinjections, we can approximately sample from the training distribution. The difference between performing no reinjections (Data-free 0R) and taking the last reinjection (Data-free last R) is considerable but is overcome by using the sequence of reinjections to consolidate, as the Data-free approach does. The sequence of pseudo-samples captures more knowledge about the replication and classification function than a single pseudo-sample, so consolidating the sequence of pseudo-sample alleviates forgetting efficiently.

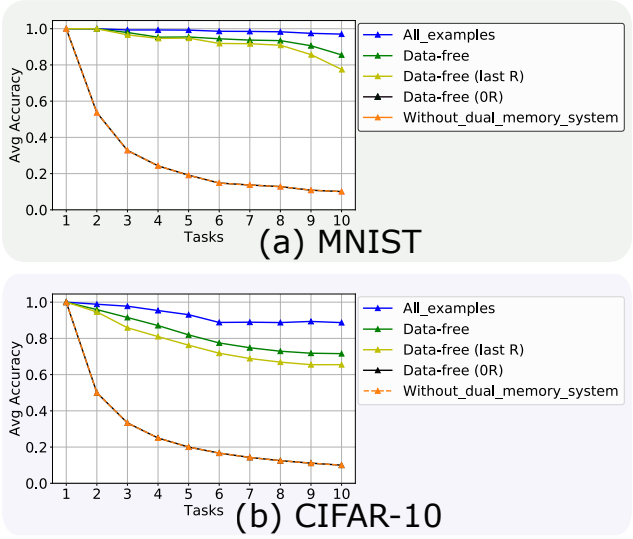

Fig. 7 Final avg.accuracy of the Data-free solutions

Third, learning the knowledge of Net_2 initialization regularizes the long-term learning in Net_1 as indicated by the Data-free curve (2-Nets) in green and the Data-free curve (2-Nets without regularization) in navy blue . A difference of up to 10 % is observed between 15 and 25 tasks, which is reduced to 7 % at the end of training. These two curves show that our regularization works when the Data-free solution works correctly (0-25 tasks) and when the Data-free approach works poorly (30-100 tasks). The gain obtained is always maintained. We can explain this effect intuitively through the stability-plasticity dilemma. If Net_1 overfits the first task, its parameters will adapt to replicate, classify and generate samples of that task. At

each consolidation stage, these parameters will be consolidated through distillation, which will cause Net_1 to maintain the parameters of the first task almost unchanged. We have empirically found that learning the initialization knowledge of Net_2 prevents Net_1 from overfitting the first task and from becoming too stable in it. Thus, we can make the neural network more plastic to learn future tasks with the initialization knowledge of Net_2. It is worth noting that there are other types of regularizations for deep learning such as dropout, L1 and L2 so as to prevent overfitting. However, we have found that the Net_2 initialization allows us to naturally regularize the incremental process on the Data-free solution without diminishing sample generation and knowledge consolidation. An easy way to avoid this step is to simplify the incremental learning problem by training the neural network with a couple of classes (e.g. half of the dataset) and then train the model class by class.

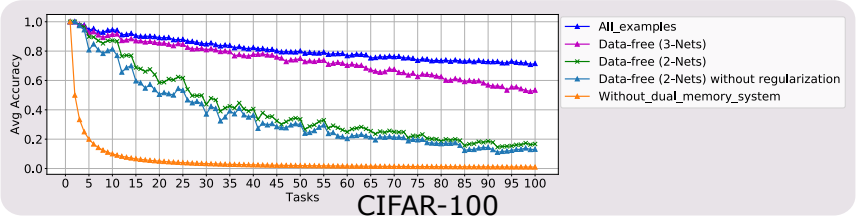

Fig. 8 Final avg.accuracy of Data-free

Fourth, for CIFAR-100 the Data-free solution provides a suboptimal result (19%). Despite the model performing satisfactorily for smaller datasets (83% MNIST and 77% CIFAR-10), it has difficulties when scaling in CIFAR-100. Advantageously, when the model is equipped with a third auto-hetero model, the system becomes more stable and it reaches a final accuracy of 54% as shown by the Data-free curve (3-Nets) in figure 8. Due to the fact that, during the consolidation process the generated pseudo-samples transmit knowledge in an approximate form, this knowledge is not completely transferred, creating a difference between NET_1 and NET_2. This difference causes long-term forgetting and it prevents the correct retention of certain tasks. Although long-term forgetting can be beneficial for learning new tasks because it provides constant plasticity, it is detrimental for the CIFAR-100 benchmark. One way to avoid this long-term forgetting is to reduce the error during the consolidation process by means of a third neural network. Thus, NET_3 will store a stable state from the earlier tasks that will be revisited several times by NET_1, decreasing the error during the consolidation process.

In summary, the Data-free solution, employing Gaussian noise and reinjections, alleviates catastrophic forgetting in all datasets. For the dual network system, our solution shows an optimal alternative for small datasets (10 classes) but it shows long-term forgetting as the information is lost for larger

datasets. In this case, the knowledge captured with Gaussian noise and reinjections is suboptimal for consolidation. We observe that when Net_1 revisits earlier knowledge several times through a third auto-hetero network (Net_3) it is capable of scaling up. While more knowledge can be captured using a third network, the computational cost and memory footprint increase significantly. In terms of memory, a 2-layer auto-hetero model of 1,000 neurons needs approximately 40 MB to be stored. We employ 3; so, 120 MB of memory are required for the 3-Net solution. In terms of computation, Net_2 performs 5 reinjections with a batch size of 256 samples which produces a final batch of 1,280 old samples. When the third network is used, another batch of 1,280 old samples is produced. This means that the model with 3 networks learns 64 examples of the new task and consolidates with 2,560 samples per learning batch. The computational cost for sample generation doubles, as does the computational cost of learning. If we extend this approach to even larger datasets, the memory and the amount of computation will likely continue to increase, making Data-free solutions very expensive method to be implemented

As a result of these observations, we decided to exchange complexity and memory for information from the environment. Combined replay stores real examples of previous tasks in a small memory buffer, allowing us to perform reinjection with real examples instead of random noise. In this way, the consolidation process can be stabilized at a very low cost in terms of resources. We present the experiences carried out with Combined replay in the following.

4.5 EXPERIMENTS: Combined Replay

This section describes the experiments carried out to evaluate the performance of our approach against the current state-of-the-art replay methods.

4.5.1 Baselines

We compare our approach with the following references:

- Auto-Hetero with buffer (AHB): An Auto-Hetero model without reinjections, which is trained with copied mini-batches of old samples (i.e. the old samples and the corresponding labels) to reveal whether the observed beneficial effects are due to the hybrid architecture.
- Auto-Hetero with buffer noise (AHBN): An Auto-Hetero model without reinjections, which is trained with copied mini-batches of noised old samples (i.e. the noised old samples and the corresponding ground-truth labels) to reveal if the observed beneficial effects are due to the added noise in the hybrid architecture. Note that AHBN is very much akin to a denoising autoencoder implementation with extra neurons for classification. That is, the inputs are noised samples while the outputs are the true samples with the corresponding ground-truth labels.

Forgetting			
Method	MNIST	CIFAR-10	CIFAR-100
CR(our)	**0.7080**	**0.2603**	**0.3902**
AHB	0.8137	0.4906	0.6957
AHBN	0.8044	0.5716	**0.055**
ICARL	0.8664	0.5040	0.7044
ER	0.8738	0.8664	0.7438

Table 2 Forgetting when using a tiny memory buffer of one sample per dataset class taken. Table taken from [50].

- ICARL (Classifier-based distillation): A rehearsal method that saves a pre-updated version of a classifier to capture previous knowledge by employing a memory buffer. We implement the fully-connected version of this method [91], which employs two classifiers with sigmoidal outputs and binary cross-entropy loss for distillation. In the experiments, we take into consideration this method due to its superior performance at the same amount of available memory in a comparison to other CL methods [3].
- Episodic Replay (ER): A classifier that uses a tiny memory buffer as a constraint to avoid catastrophic forgetting. It was recently stated that CNN classifiers employing tiny memory buffers are less prone to catastrophic forgetting than other popular rehearsal methods [52]. In this work, a fully-connected version of this method is implemented.

4.5.2 Methodology

We adapted the experimental setting proposed in Experience Replay [52] (Alg. 1.) originally designed to benchmark rehearsal methods. The original algorithm compares the final performance of continual learning approaches by carrying out four main operations: i. the samples of the new set of classes are learned only once. ii. the memory buffer of old samples is updated at every learning step. iii. the mini-batch of new classes (new samples) is merged with the mini-batch of old classes (old samples) randomly selected from the memory buffer. iv. the parameters of the models under test are updated by backpropagating the loss of the merged mini-batches.
We made three adaptations to this algorithm:

- The new samples from the new set of classes are learned several times.
- The memory buffer is updated after learning a new task to ensure that it always contains only old samples.
- The mini-batch of old samples is copied as many times as reinjections are performed to update all the methods with the same amount of old data.

In this way, after n reinjections, we obtain $n{*}$*mini-batch_size* pseudo-samples. *Mini-batch_size* refers to the size of the mini-batch of old samples. To obtain the same number of old true samples and pseudo-samples for a fair comparison, we copied the true samples n times. Therefore, if no reinjection is performed, no compensation is needed – so the mini-batch is not copied. If one reinjection is performed, the mini-batch of old samples is duplicated

to obtain the same mini-batch size, and so on. Note that the interest behind copying the old mini-batches is to update all the rehearsal methods with the equivalent amount of old data employed by Combined replay.

4.5.3 Results

In all our Combined replay experiments, we perform 5 reinjections; thus, the mini-batch is quintupled to update all the models with an equal number of samples. Furthermore, forgetting is not reduced and no detriment is observed in the generalization ability when doing so. We employed the reser-

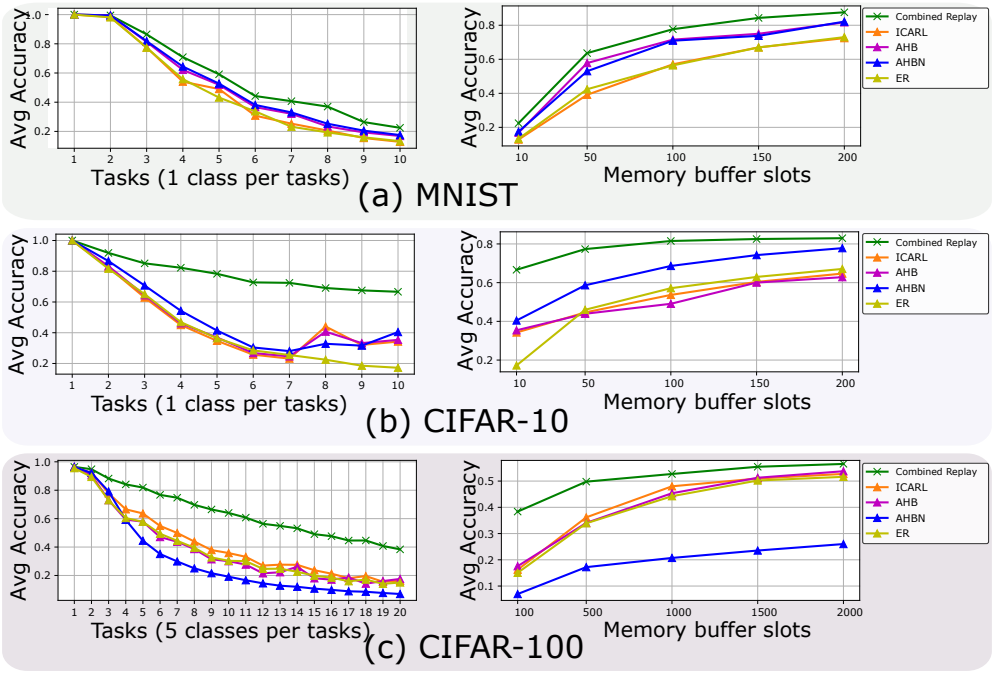

Fig. 9 On the left side, the average accuracy over tasks when only 1 sample per class is used. On the right side, the final average accuracy as a function of the buffer size.

voir sampling routine [52] to update the memory buffer since any sample seen is equally likely to be stored. We consider that the buffer size is bounded at (20 * #classes). For instance, on CIFAR-100, the largest buffer size is equal to 2000, which is also a size utilized in the literature [4, 26, 27, 32].
We average accuracy over 3 runs on test sets during the learning steps. Figure 9 and Table 2 summarize the results of the comparison with state-of-the-art approaches. The following observations can be made.
First, Combined replay greatly outperforms all the hybrid architectures that do not perform reinjections (i.e. AHB and AHBN). Also, our approach outperforms state-of-the-art replay methods relying on the same size of

the memory buffers. Furthermore, for very tiny memory buffers, Combined replay yields a higher performance at all benchmarks presented in Figure 9. On CIFAR-100 (Figure 9(c)(right), for a memory buffer of size 100, the accuracy of Combined replay is about 20% higher than ER and ICARL. This result is interesting considering that the performance of the CNN classifiers in ER seems to be much higher than that of fully connected classifiers [52]. We explain this result as follows: i. the test set is drawn from already seen examples of the training set in the original ER paper [52]; ii. the CNN used for feature extraction might help retain previous knowledge avoiding catastrophic forgetting, iii. the main reason for this difference is perhaps that ER evaluates its approach with an oracle and task identifiers while in this work we employ global accuracy on classes. The difference in performance between the methods gets smaller when the memory buffer size becomes larger. For a memory buffer of 2000 samples (Figure 9(c)(right), the curves meet showing a comparable performance. Moreover, ICARL delivers a slightly better performance (52%) than the performance obtained in [26, 91]. This difference might be due to the cloned mini-batches.

Second, it has already been observed that, often, the reservoir sampling routine can completely dislodge samples of the older classes when the memory buffer is very small [52]. Even though representative memory samples and balanced training set are not guaranteed with the reservoir update routine, Combined replay can capture a considerable amount of knowledge from most previously learned classes. The reinjections considerably alleviate the lack of previous samples while other methods experiment higher forgetting as it is shown in Table 2.

Third, the lowest value of forgetting for AHBN in Table 2, on CIFAR-100 dataset, suggests that the denoising implementation allows remembering some classes quite well. However, the low values in the average accuracy of Figure 9(left)(c) suggests that the denoising implementation struggles to learn new tasks. Hence, the forgetting and the accuracy results taken together indicate that AHBN suffers from lack of plasticity (i.e. the inability to update its knowledge) after learning some tasks.

In summary, Combined replay, employing reservoir sampling and very small memory buffers, outperforms all the presented replay methods in terms of accuracy and forgetting. For the selected hyper-parameters (i.e. noise strength and number of reinjections), our solution shows a less pronounced slope as the memory becomes larger. This suggests that the knowledge captured through reinjections is mostly beneficial when a reduced set of samples is available. We observe that the knowledge captured with our architecture reaches an optimal performance when a memory buffer size of 10 samples per class is employed. While more knowledge can be captured using larger memory buffers, the performance gain gradually decreases. This finding can be further confirmed in Figure 9(c)(right) where ICARL, EM and our solution yield a similar performance when a memory buffer of size 2000 is employed. These observations are discussed in more detail in the following subsection,

showing that the Combined replay suffers from a bias problem like most replay-based methods [4, 26–28, 31, 32, 34, 38, 39].

4.5.4 Bias correction

When incrementally training a neural network, a major problem is the drift of the weights from old to new tasks. This causes previously learned classes to be biased towards the classes of the most recently learned tasks. This phenomenon occurs because the model has seen many different samples of the new task but none or a few from previous tasks. This detrimental effect is also present in the classification layers (i.e. the fully connected layers after the feature extractor) [32]. Combined replay is not exempt from this problem, as can be seen in the confusion matrix of Figure 10(a).

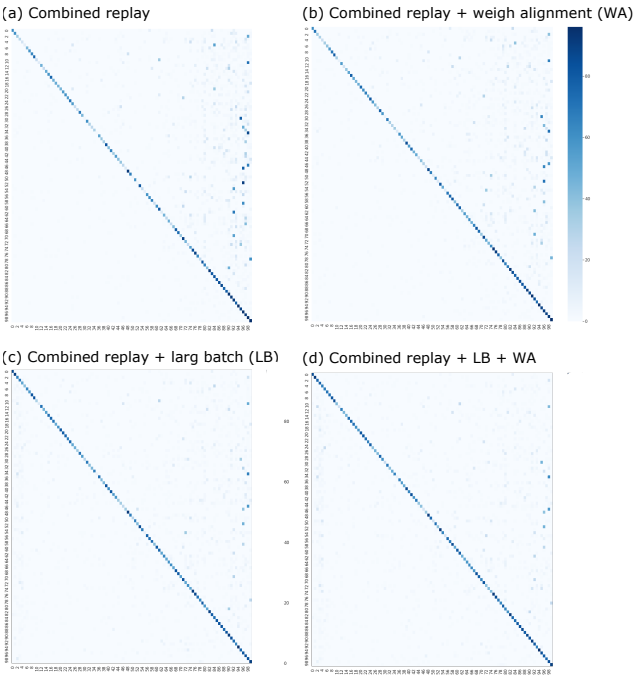

Fig. 10 Confusion matrix of CIFAR-100 after learning the 100 classes in 20 learning steps.

Several continual learning approaches employ different techniques to prevent such drift to new tasks [26–28, 31]. This work is interested in solving the bias problem without involving additional memory [28, 38], fine-tuning techniques [27] or extra loss functions [29, 30]. Weight Aligning (WA) [32] is an approach that meets these requirements and gives results comparable to state-of-the-art alternatives. In WA, it is observed that the norms of the weight vectors of the new classes are more salient, leading to higher predictions towards the new classes. As a result, the trained model predicts an

input image as belonging to a new class instead of its corresponding class. The paper concludes that the associated weights must have a similar norm for the classes to be treated without bias and proposes a solution to normalize such weights. WA consists in correcting the weights after learning each task in the fully-connected layer with . To do so, WA aligns the norm of the vectors of the new classes with those of the old classes as follows: $\widehat{W}_{new} = \gamma * \cdot W_{new}$ where $\gamma = \frac{Mean(\|W_{old}\|_2)}{Mean(\|W_{new}\|_2)}$ and $\widehat{W}_{new}$ are the corrected weights, W_{new} are the new task weights after training and W_{old} are the old task weight. In this way, the average norm of the weight vectors of the new tasks is the same as that of the old ones. Thus, the final effect of WA is that the γ coefficient rescales the weights of the new classes, reducing bias shift, as it is shown in the Figure 10(b) and Figure 11. WA adjusts the bias towards the new classes, providing a 4% accuracy gain and reducing catastrophic forgetting by 4%, as shown in Figure 11 and Table 3. WA alleviates the problems of bias shift by simply modifying the weights of the new classes without changing the ratio between the pseudo-samples and the new examples. However, we have empirically found that Combined replay can also reduce the bias shift towards new classes by increasing the size batch (LB for large batch) of old tasks to capture Net_2 knowledge, as shown in Figure 11 and Table 3. Similarly to the Data-free solution, where the batch of new tasks was smaller than the batch of old tasks, Combined replay also benefits from this arrangement. So, we have employed a batch size of 100 examples instead of 10 to perform the reinjections. This batch configuration provides a gain of 10% over previous Combined replay results and 6% over WA. We also reduced forgetting by 21% over previous results and by 17% over WA, as shown in Figure 11 and Table 3. Finally, as WA and LB are different solutions, one changes the parameters of the new classes while the other increases the batch of old tasks; we could think that they can complement each other. We therefore combine the two solutions and obtain results similar to those obtained when only one large batch is employed for the old tasks. WA is Combined with the batch size of 100 examples showing a similar performance to that obtained using only the large batch, as it is shown in Figure 11 and Table 3.

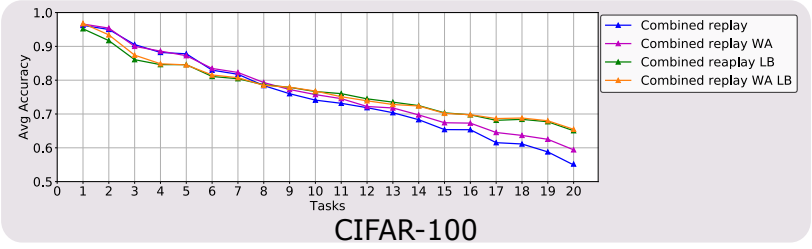

Fig. 11 Final average accuracy of the Combined replay in the CIFAR-100 dataset when using a memory buffer size of 2000 examples.

The present results indicate that Combined replay tends to be biased toward the new class when it consolidates prior knowledge in an unbalanced way. By providing more examples to consolidate through reinjections, Combined replay is able to find a solution that comprises old and new tasks without this issue. Indeed, Net_1 will be able to adapt the parameters correctly if the pseudo-samples represent the overall knowledge previously learned. We have empirically found that a batch size of 100 examples correctly represents the learned knowledge and that the number of examples also corresponds to the number of classes in the dataset. Note that 5 reinjections are performed; the model learns 10 examples of the new task and consolidates with 500 pseudo-samples at each learning stage. These sizes are static and are maintained during the whole training.

Forgetting				
Bias correction —— Dataset	Combined replay (no correction)	Combined replay WA (weight aligning)	Combined replay LB (large batch)	Combined replay WA LB (weight aligning + large batch)
CIFAR-100	0.21	0.17	0.09	0.08

Table 3 Forgetting score on the CIFAR-100 dataset when using a small memory buffer size of 2000 examples.

The low forgetting values observed in Table 3 are very convenient but not surprising. With larger batch sizes, the model becomes very stable, preventing old tasks from being forgotten.

As the model is less plastic when using large bath sizes, it may not learn new tasks with maximum precision. For example, when a batch of size 10 is used to perform reinjections, the new task usually shows a maximum accuracy of about 90%. Task performance degrades through incremental learning tasks that it generates a difference from the maximum value achieved. Since a batch size of 10 examples partially represents previous knowledge, the differences become more significant, which leads to a relatively high forgetting score. The high forgetting score is due to two factors: first, the model learns the first class very well (about 90% of accuracy) without considering the previous classes. Secondly, the model does not consolidate these classes sufficiently in the following incremental steps. Hence the precision of those classes decreases considerably and the model shows a high forgetting score. In contrast, when Combined replay LB learns new tasks, it shows decreasing accuracy (i.e. early tasks will reach a high accuracy of around 90% while more distant tasks will have a maximum accuracy of 65-70%). Thus, new classes are learned, but their performance is limited due to the consolidation of the previous class, indicating restricted plasticity. The decreasing inaccuracy is due to fact that the model correctly fits most of the classes without letting one class dominate over all the others. The difference between the

maximum achieved accuracy and the degraded accuracy will not be significant because the knowledge is well consolidated with a batch size of 100 examples, so the forgetting score is low. Probably, the major problem in using a large and constant batch over time is that there is a bias towards new classes (in Figure 11 c/d, see pale blue on the left corresponding to the first classes). Naturally, the first classes learned are revisited more often than the intermediate classes. Thus, it can be seen that some examples of intermediate and final classes are considered to be examples of initial classes. By using a large batch for consolidation, we trade stability for plasticity. For example, the model will learn the new task without any difficulty if it consolidates previous tasks with few or no samples of old tasks. But, the detrimental effect is that the model will forget the old task. Conversely, the model will hardly learn the new task if it consolidates previous tasks with several old task samples. In other words, the ratio between old and new task samples is related to the trade-off between stability and plasticity. We increase the number of samples the model replays for consolidating old tasks (i.e. higher stability). Thus, the model has less freedom to change weights and learn the new class (i.e. lower plasticity). In our model, the batch size regulates the balance between stability (i.e. retaining old tasks) and plasticity (i.e. learning new tasks). By reducing the plasticity of the model, we obtain in parallel the desired outcome of reducing the bias towards new tasks. In a nutshell, larger batch sizes allow the model to consolidate old tasks and avoid overlearning new ones. Therefore, it achieves the desired goal of reducing the bias toward new tasks.

Note that this finding is not a minor discovery since our model obtains a gain of 10% just by using a larger batch size to consolidate old tasks for the same memory buffer size. Although several approaches employ distillation to consolidate old tasks, to the best of our knowledge, there is no mention of using large batch sizes to increase stability. For instance, our finding does not appear in the well-known framework that determines the stability-plasticity trade-off of the continual learners [3].

Accuracy							
Methods	DMC	GDumb	EEIL	BiC	WA	CR	CR+LB
Cifar-100 (20 tasks)	56.8	45.2	63.4	63.8	62.6	55.5	**65.0**

Table 4 Cifar-100 state-of-the-art results. DMC:[67], GDumb: [102]; EEIL: [27], BiC: [28], WA: [32], CR: [50], CR+LB: (ours)

Earlier, we compare our solution against ICARL and ER at identical conditions (i.e. fixed feature extractor and similar architectures). Next, we compare our solution against published results obtained from several replay approaches on the same benchmark using the same memory buffer size. In particular, we compare the results obtained after adjusting the bias, as presented in Table 4. Please note that the results shown in the Table 4 were

extracted from the original works as in [32, 37, 78]. These baselines are intended to highlight the results obtained in CIFAR-100 against approaches that scale and perform well on the same dataset. We can observe that our approach yields better accuracy than the state-of-the-art in CIFAR-100 when a memory buffer of 2000 examples is employed.

5 Discussion

Data-free and Combined replay highlight the importance of reinjections to improve the information retrieval process for transferring knowledge between two ANNs when memory buffer sizes are constrained or when they are not available. Reinjections are performed in a hybrid architecture to generate pseudo-data that captures previously acquired knowledge. The pseudo-data can be generated through a reinjection sampling procedure with Gaussian noise or with noised examples from tiny memory buffers. When incrementally learning new tasks, the pseudo-data set is jointly learned with the samples of a new task to overcome catastrophic forgetting. Previously, in experiments without data, we have further investigated not performing reinjections and taking the last of the sequence. We have also analyzed the instability of the Data-free solutions and the third neural network needed to stabilize learning. Here, we discuss some common features of the Data-free and the Combined replay solutions as well as their advantages and disadvantages.

First, in this work, we have prioritized the average accuracy while targeting minimal memory buffer sizes for embedded applications. Embedded applications have limited available memory as well as a limited number of operations. Data-free does not store data, so this is a critical point in applications where memory constraints exist. In the case of Combined replay, some old examples are stored. However, the memory size is severely constrained and the number of operations is reduced. Ideally, Data-free and Combined replay are two different implementations that allow trading memory for operations or vice-versa while providing similar performance.

Second, we investigated the quality of the pseudo-data set in terms of the strength of added noise. Figure 12 presents the noise strength vs the average accuracy for Combined replay and AHBN. On MNIST dataset, for a minimal memory buffer of size 10, the more noise is added before performing reinjections, the better Combined replay captures previous knowledge. In this case, the added noise can also improve the final performance in AHBN as it is the case for a noise strength between 0.1 and 0.3. However, a negative effect of noise in AHBN reveals that the performance gain of Combined replay is not due to a denoising effect but to reinjections. This finding suggests that it might be possible to interpret it as a measure of the generative capacity of the model. The more noise we add, the closer it is to the Data-free option and, conversely, the less noise we add, the closer it is to a version similar to Combined replay, a study beyond the scope of the present work.

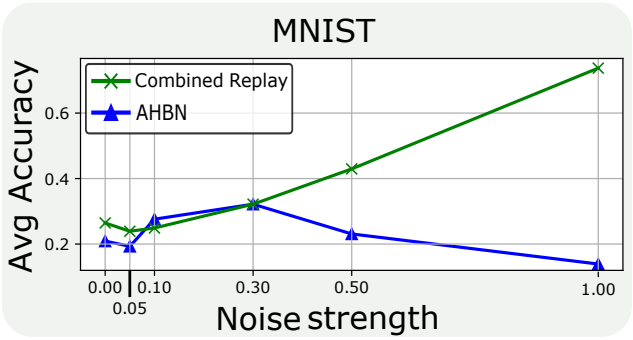

Fig. 12 Final avg.accuracy of Combined replay over noise strength for a memory buffer of 10 samples.

For simplicity, Combined replay employs an isotropic Gaussian noise $\mathcal{N}(0, I)$ that is pondered by a noise strength of 0.05 in all experiences of Figure 9.
Third, regardless of their final performance, the Data-free and the Combined replay solutions are conceived to alleviate catastrophic forgetting under different constraints. They are different because they capture the knowledge in two different ways, that is the reason why they are suitable for different continual learning applications. On the one hand, it is possible to employ our model for data-free applications where privacy issues are the primary concern. Thus, knowledge is captured with random noise and reinjections. On the other hand, we employ the hybrid model to improve the examples stored in the memory buffer. Thus, knowledge is captured with real examples and reinjections.
In a nutshell, the strength of the noise, the number of reinjections and the old sample buffer size play a crucial role in retrieving previously acquired knowledge. These parameters directly regulate the generation of pseudo-samples that influence the preservation of old knowledge. In our view, these two parameters can be considered as a function of the memory buffer size and the properties of the dataset (e.g. the distance between classes, the number of samples per class, etc.). However, in this work, we found it essential to balance training using a batch representing the previously learned knowledge. Further research on Combined replay could lead to an improved selection of examples for reinjections to consolidate correctly with even fewer pseudo-examples.
Fourth, there is little or no research effort on how to forget what is unnecessary to remember. Replay methods "self-regulate" the learning capacity of the model through the replay. If old memories are constantly replayed, meaningful activation patterns will naturally appear, while unnecessary patterns will be forgotten. The latter might be the case for our model that keeps only the crucial patterns that allow efficient consolidation and active forgetting. There are, to our knowledge, no proposals to model the forgetting mechanism even though it seems to be a crucial key to continual human learning.

To endow a learner with active forgetting would probably require identifying what is necessary to consolidate and what makes sense to forget. In our opinion, the mechanism of reinjections can be a first exploratory clue to identify what needs to be remembered and forgotten in continual learning.
Fifth, the Data-free approach has the main characteristics of an ideal continual learning model since it is able to adapt to all circumstances. That is, it does not need external information to operate and it is, therefore, easy to deploy. Since this model does not store anything, it only needs to generate pseudo-samples through reinjection and random noise, regardless of what it has learned and what it will learn. These features make the Data-free model ideal for applications where privacy is essential. However, we note that it is unstable for large datasets and it requires generating large batches of pseudo-samples to consolidate. Combined replay sacrifices the main advantage of the Data-free solution by equipping the model with a memory buffer that makes the model stable at a very low resource cost. For example, the data-free version (3-Nets) employs an additional 40 MB neural network, while the 2,000 memory buffer size requires only 15 KB memory, which is 2,600 times smaller than the data-free (3-Net) version. In terms of computation, Data-free (3-Nets) model employs 2,560 pseudo-samples to consolidate while CR uses 50 and CR+LB 500, reducing 51.2 and 5.12 times the pseudo-samples needed for consolidation. All this makes Combined replay faster, more accurate and less costly to implement than Data-free alternatives.
The reason for adding this third network is related to a possible limitation of data-free when it performs more than fifteen consolidation steps. In this scenario, the performance of Data-free decreases, illustrating that it can forget old tasks in the long run. Random noise and reinjections generate functional samples for some consolidation steps; however, the quality of generated samples may vanish through consolidation steps. Combined replay prevents the examples vanish over updates because the model reviews faithful samples from the past. Data-free emulates this behavior by reviewing past samples generated with a third neural network having undergone less vanishing (i.e. performed fewer consolidation steps). Our implementation with three neural networks allows the model to review more representative samples of the past, thus reducing class forgetting. In our opinion, it would be like reviewing the original snapshot of a vanished personal memory. Implementing three neural networks is attractive because it allows our model to provide a data-free solution for large-scale datasets. However, it requires knowing the number of classes the model will learn and could require extra neural networks from even larger datasets to emulate that snapshot of the past.
The two models presented in this work are both based on the same fundamental ideas. However, they are two different models for different uses. Data-free model does not need to store the training set data and is suitable for privacy-preserving applications. Combined replay is a computationally lighter version that allows for higher performance using memory buffers.

Together, they show the generality of our approach and the beneficial effects of reinjections for continual learning.

6 Conclusion

This work presents a novel approach for retrieving previously learned information to reduce catastrophic forgetting in artificial neural networks. The experimental results on MNIST, CIFAR-10 and CIFAR-100 presented in this work lead to the following conclusions. First, our Data-free approach is a privacy-preserving solution that alleviates catastrophic forgetting in small datasets, extending to large datasets by duplicating the same principle. Second, the memory buffer employed by our Combined replay approach allows for robust solution that outperforms state-of-the-art replay methods when relying on a minimal memory buffer. Third, our approach does not require representative memory samples and a balanced training set to be efficient, two mandatory conditions for other replay methods. Future works will include determining the best examples to perform reinjections to reduce memory and complexity. We would like to enable Combined replay to provide better results in embedded applications.

Declarations

– Funding: This work has been partially supported by MIAI@Grenoble Alpes, (ANR-19-P3IA-0003).
– Conflict of interest/Competing interests: French Alternative Energies and Atomic Energy Commission (CEA).
– Ethics approval : Not applicable
– Consent to participate :Not applicable
– Consent for publication :Not applicable
– Availability of data and materials: Not applicable
– Code availability : Not applicable
– Authors' contributions: Not applicable

References

[1] Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C.: Measuring catastrophic forgetting in neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

[2] Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019)

[3] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. arXiv preprint arXiv:1909.08383 (2019)

[4] Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation. arXiv preprint arXiv:2010.15277 (2020)

[5] Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., Sanner, S.: Online continual learning in image classification: An empirical survey. arXiv preprint arXiv:2101.10423 (2021)

[6] McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of Learning and Motivation vol. 24, pp. 109–165. Elsevier, ??? (1989)

[7] Akers, K.G., Martinez-Canabal, A., Restivo, L., Yiu, A.P., De Cristofaro, A., Hsiang, H.-L.L., Wheeler, A.L., Guskjolen, A., Niibori, Y., Shoji, H., *et al.*: Hippocampal neurogenesis regulates forgetting during adulthood and infancy. Science **344**(6184), 598–602 (2014)

[8] Yang, G., Lai, C.S.W., Cichon, J., Ma, L., Li, W., Gan, W.-B.: Sleep promotes branch-specific formation of dendritic spines after learning. Science **344**(6188), 1173–1178 (2014)

[9] Barry, D.N., Maguire, E.A.: Remote memory and the hippocampus: A constructive critique. Trends in cognitive sciences **23**(2), 128–142 (2019)

[10] Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734 (2017)

[11] Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7765–7773 (2018)

[12] Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International Conference on Machine Learning, pp. 4548–4557 (2018)

[13] Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)

[14] Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong

learning with a network of experts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3366–3375 (2017)

[15] Xu, J., Zhu, Z.: Reinforced continual learning. In: Advances in Neural Information Processing Systems, pp. 899–908 (2018)

[16] Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: International Conference on Machine Learning, pp. 3925–3934 (2019)

[17] Verma, V.K., Liang, K.J., Mehta, N., Rai, P., Carin, L.: Efficient feature transformations for discriminative and generative continual learning. arXiv preprint arXiv:2103.13558 (2021)

[18] Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)

[19] 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. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)

[20] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 139–154 (2018)

[21] Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning, pp. 3987–3995 (2017)

[22] Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 532–547 (2018)

[23] Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International Conference on Machine Learning, pp. 4528–4537 (2018)

[24] Park, D., Hong, S., Han, B., Lee, K.M.: Continual learning by asymmetric loss approximation with single-side overestimation. In: Proceedings of the IEEE/CVF International Conference on Computer

Vision, pp. 3335–3344 (2019)

[25] Lee, J., Hong, H.G., Joo, D., Kim, J.: Continual learning with extended kronecker-factored approximate curvature. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9001–9010 (2020)

[26] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2001–2010 (2017)

[27] Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 233–248 (2018)

[28] Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 374–382 (2019)

[29] Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 831–839 (2019)

[30] Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning **12365**, 86–102 (2020). Springer

[31] He, J., Mao, R., Shao, Z., Zhu, F.: Incremental learning in online scenario. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13926–13935 (2020)

[32] Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S.-T.: Maintaining discrimination and fairness in class incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13208–13217 (2020)

[33] Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. arXiv preprint arXiv:2004.07211 (2020)

[34] Mittal, S., Galesso, S., Brox, T.: Essentials for class incremental learning. arXiv preprint arXiv:2102.09517 (2021)

[35] Hu, X., Tang, K., Miao, C., Hua, X.-S., Zhang, H.: Distilling

causal effect of data in class-incremental learning. arXiv preprint arXiv:2103.01737 (2021)

[36] Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486 (2019)

[37] Prabhu, A., Torr, P., Dokania, P.: Gdumb: A simple approach that questions our progress in continual learning. In: The European Conference on Computer Vision (ECCV) (2020)

[38] Belouadah, E., Popescu, A.: Il2m: Class incremental learning with dual memory. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 583–592 (2019)

[39] Thai, A., Stojanov, S., Rehg, I., Rehg, J.M.: Does continual learning= catastrophic forgetting? arXiv preprint arXiv:2101.07295 (2021)

[40] Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science **7**(2), 123–146 (1995)

[41] Ans, B., Rousset, S.: Avoiding catastrophic forgetting by coupling two reverberating neural networks. Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie **320**(12), 989–997 (1997)

[42] Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. arXiv preprint arXiv:1705.08690 (2017)

[43] Atkinson, C., McCane, B., Szymanski, L., Robins, A.: Pseudo-recursal: Solving the catastrophic forgetting problem in deep neural networks. arXiv preprint arXiv:1802.03875 (2018)

[44] Lavda, F., Ramapuram, J., Gregorova, M., Kalousis, A.: Continual classification learning using generative models. arXiv preprint arXiv:1810.10612 (2018)

[45] Lesort, T., Gepperth, A., Stoian, A., Filliat, D.: Marginal replay vs conditional replay for continual learning. In: International Conference on Artificial Neural Networks, pp. 466–480 (2019). Springer

[46] Van de Ven, G.M., Siegelmann, H.T., Tolias, A.S.: Brain-inspired replay for continual learning with artificial neural networks. Nature communications **11**(1), 1–14 (2020)

[47] Atkinson, C., McCane, B., Szymanski, L., Robins, A.: Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. Neurocomputing **428**, 291–307 (2021)

[48] Knoblauch, J., Husain, H., Diethe, T.: Optimal continual learning has perfect memory and is np-hard. In: International Conference on Machine Learning, pp. 5327–5337 (2020)

[49] Solinas, M., Galiez, C., Cohendet, R., Rousset, S., Reyboz, M., Mermillod, M.: Generalization of iterative sampling in autoencoders. In: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 877–882 (2020). IEEE

[50] Solinas, M., Rousset, S., Cohendet, R., Bourrier, Y., Mainsant, M., Molnos, A., Reyboz, M., Mermillod, M.: Beneficial effect of combined replay for continual learning. In: ICAART (2), pp. 205–217 (2021)

[51] Jiang, Y., Pehlevan, C.: Associative memory in iterated overparameterized sigmoid autoencoders. In: International Conference on Machine Learning, pp. 4828–4838 (2020)

[52] Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: Continual learning with tiny episodic memories (2019)

[53] LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist **2** (2010)

[54] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[55] Ratcliff, R.: Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. Psychological review **97**(2), 285 (1990)

[56] Lewandowsky, S., Li, S.-C.: Catastrophic interference in neural networks: Causes, solutions, and data, 329–361 (1995)

[57] McClelland, J.L., McNaughton, B.L., O'Reilly, R.C.: Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review **102**(3), 419 (1995)

[58] Ans, B., Rousset, S.: Avoiding catastrophic forgetting by coupling two reverberating neural networks. Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie **320**(12), 989–997 (1997)

[59] French, R.M.: Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference, 335–340 (1994)

[60] Abraham, W.C., Robins, A.: Memory retention–the synaptic stability versus plasticity dilemma. Trends in neurosciences **28**(2), 73–78 (2005)

[61] Aljundi, R., Rohrbach, M., Tuytelaars, T.: Selfless sequential learning. arXiv preprint arXiv:1806.05421 (2018)

[62] Rannen, A., Aljundi, R., Blaschko, M.B., Tuytelaars, T.: Encoder based lifelong learning. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1320–1328 (2017)

[63] Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong learning with a network of experts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3366–3375 (2017)

[64] Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018)

[65] Benna, M.K., Fusi, S.: Computational principles of synaptic memory consolidation. Nature neuroscience **19**(12), 1697–1706 (2016)

[66] Dhar, P., Singh, R.V., Peng, K.-C., Wu, Z., Chellappa, R.: Learning without memorizing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5138–5146 (2019)

[67] Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., Kuo, C.-C.J.: Class-incremental learning via deep model consolidation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1131–1140 (2020)

[68] Lan, J., Liu, R., Zhou, H., Yosinski, J.: Lca: Loss change allocation for neural network training. arXiv preprint arXiv:1909.01440 (2019)

[69] Benzing, F.: Understanding regularisation methods for continual learning. arXiv preprint arXiv:2006.06357 (2020)

[70] Farquhar, S., Gal, Y.: Towards robust evaluations of continual learning. arXiv preprint arXiv:1805.09733 (2018)

[71] Quiroga, R.Q., Reddy, L., Kreiman, G., Koch, C., Fried, I.: Invariant visual representation by single neurons in the human brain. Nature **435**(7045), 1102–1107 (2005)

[72] Frankland, P.W., Bontempi, B.: The organization of recent and remote memories. Nature reviews neuroscience **6**(2), 119–130 (2005)

[73] Sorrells, S.F., Paredes, M.F., Cebrian-Silla, A., Sandoval, K., Qi, D.,

Kelley, K.W., James, D., Mayer, S., Chang, J., Auguste, K.I., *et al.*: Human hippocampal neurogenesis drops sharply in children to undetectable levels in adults. Nature **555**(7696), 377–381 (2018)

[74] Boldrini, M., Fulmore, C.A., Tartt, A.N., Simeon, L.R., Pavlova, I., Poposka, V., Rosoklija, G.B., Stankov, A., Arango, V., Dwork, A.J., *et al.*: Human hippocampal neurogenesis persists throughout aging. Cell stem cell **22**(4), 589–599 (2018)

[75] Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., Bejnordi, B.E.: Conditional channel gated networks for task-aware continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3931–3940 (2020)

[76] Masana, M., Tuytelaars, T., van de Weijer, J.: Ternary feature masks: continual learning without any forgetting. arXiv preprint arXiv:2001.08714 **4**(5), 6 (2020)

[77] Lomonaco, V., Maltoni, D.: Core50: a new dataset and benchmark for continuous object recognition. In: Conference on Robot Learning, pp. 17–26 (2017)

[78] Hocquet, G., Bichler, O., Querlioz, D.: Ova-inn: Continual learning with invertible neural networks. arXiv preprint arXiv:2006.13772 (2020)

[79] Rajasegaran, J., Khan, S., Hayat, M., Khan, F.S., Shah, M.: itaml: An incremental task-agnostic meta-learning approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13588–13597 (2020)

[80] Caccia, M., Rodriguez, P., Ostapenko, O., Normandin, F., Lin, M., Page-Caccia, L., Laradji, I.H., Rish, I., Lacoste, A., Vázquez, D., et al.: Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. Advances in Neural Information Processing Systems **33** (2020)

[81] Farajtabar, M., Azizan, N., Mott, A., Li, A.: Orthogonal gradient descent for continual learning. In: International Conference on Artificial Intelligence and Statistics, pp. 3762–3773 (2020)

[82] Saha, G., Garg, I., Roy, K.: Gradient projection memory for continual learning. arXiv preprint arXiv:2103.09762 (2021)

[83] Iscen, A., Zhang, J., Lazebnik, S., Schmid, C.: Memory-efficient incremental learning through feature adaptation. In: European Conference on Computer Vision, pp. 699–715 (2020). Springer

[84] French, R.M.: Pseudo-recurrent connectionist networks: An approach to the'sensitivity-stability'dilemma. Connection Science **9**(4), 353–380 (1997)

[85] Wu, C., Herranz, L., Liu, X., Wang, Y., Van de Weijer, J., Raducanu, B.: Memory replay gans: learning to generate images from new categories without forgetting. arXiv preprint arXiv:1809.02058 (2018)

[86] Lesort, T., Caselles-Dupré, H., Garcia-Ortiz, M., Stoian, A., Filliat, D.: Generative models from the perspective of continual learning. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2019). IEEE

[87] Liu, Y., Su, Y., Liu, A.-A., Schiele, B., Sun, Q.: Mnemonics training: Multi-class incremental learning without forgetting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12245–12254 (2020)

[88] Chaudhry, A., Gordo, A., Dokania, P.K., Torr, P., Lopez-Paz, D.: Using hindsight to anchor past knowledge in continual learning. arXiv preprint arXiv:2002.08165 (2020)

[89] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv preprint arXiv:1406.2661 (2014)

[90] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

[91] Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. International Conference on Learning Representations (ICLR) (2018)

[92] Jeon, I.H., Shin, S.Y.: Continual representation learning for images with variational continual auto-encoder. In: ICAART (2), pp. 367–373 (2019)

[93] Mellado, D., Saavedra, C., Chabert, S., Torres, R., Salas, R.: Self-improving generative artificial neural network for pseudorehearsal incremental class learning. Algorithms **12**(10), 206 (2019)

[94] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning Workshop (2015)

[95] Belouadah, E., Popescu, A.: Deesil: Deep-shallow incremental learning. In: Proceedings of the European Conference on Computer Vision

(ECCV) Workshops, pp. 0–0 (2018)

[96] van de Ven, G.M., Li, Z., Tolias, A.S.: Class-incremental learning with generative classifiers. arXiv preprint arXiv:2104.10093 (2021)

[97] 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, pp. 770–778 (2016)

[98] Choi, Y., El-Khamy, M., Lee, J.: Dual-teacher class-incremental learning with data-free generative replay. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3543–3552 (2021)

[99] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)

[100] Misra, D.: Mish: A self regularized non-monotonic neural activation function. arXiv preprint arXiv:1908.08681 (2019)

[101] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. 3rd International Conference on Learning Representations ICLR (2014)

[102] Prabhu, A., Torr, P.H., Dokania, P.K.: Gdumb: A simple approach that questions our progress in continual learning. In: European Conference on Computer Vision, pp. 524–540 (2020). Springer