Continual Learning: Comparison and New Strategies based on Data Distillation

Francisco de Sales Javier Ruiz Baquedano

Abstract

This thesis presents an improved variation of Distilled Replay for continual learning, incorporating distribution matching to enhance computational efficiency while preserving model performance. By refining the data distillation process, the proposed method significantly reduces training time without compromising accuracy. The study also provides a broader examination of continual learning strategies, including regularization techniques, dynamic architectures and memory replay, and introduces the main dataset distillation approaches. Experimental evaluations on the MNIST dataset demonstrate that the proposed method effectively mitigates catastrophic forgetting while optimizing resource utilization. These findings offer valuable insights for developing more efficient neural network training strategies in non-stationary learning environments.

Index Terms

Continual Learning, Catastrophic Forgetting, Dataset Distillation, Neural Networks, Lifelong Learning, Memory Replay, Distribution Matching.

I. Introduction

As the availability and size of the data increases rapidly, as do the time and power needed to train the models. Data not only grows in size, but also changes its distribution. Labels may vary, and some isolated events may only reappear after extended periods of time.

These kinds of problems, which strongly affect deep neural networks, are present in many not-fixed real-world scenarios, and the efforts to address them are usually attributed to the field of *Continual Learning*, also known as *Lifelong Learning*.

Related to the field, due to its intention of reducing computing power, time and memory when training and using neural networks, *model distillation* and *dataset distillation* try to compress either the model, or the dataset, respectively, seeking to achieve a similar performance to the original model.

A. Learning without forgetting

As with many other learning systems, humans tend to forget information over time if they are not exposed to it and do not work towards retain it. Modeled in 1885 by Ebbinghaus with the *exponential forgetting curve* [1], and later by many others [2], [3], [4], data shows that recalling the learned concepts with some spacing minimizes its loss in humans memory.

Deep neural networks learn by training, and when they do, are very effective for retaining huge amounts of information and accessing it almost instantly, but this training phase is computationally intensive and time-consuming. In non-stationary environments, where scenarios change drastically, a model, for example, for a classification problem, could be trained on a task A with a dataset $D_A = (x_i, y_i)$, minimizing a loss function as $\min_x \sum_{x_i,y_i \in D_A} L(\hat{y}_\theta(x_i), y_i)$ with some random initialization θ, but a new task B, previously unknown, with $D_B = (x_i, y_i)$ may appear. A natural thought could be to train the previous model on the new task, minimizing a loss function $\min_x \sum_{x_i,y_i \in D_B} L(\hat{y}_\theta(x_i), y_i)$, initializing with the solution to task A. This whole process generally ends with good performance at task B, but worse or even poor performance at task A, as the prior learning has

Author: Francisco de Sales Javier Ruiz Baquedano, fcosalesjavier@gmail.com
Advisor: Javier del Ser Lorente, Chief AI Scientist and Research Professor at TECNALIA Research & Innovation
Thesis dissertation submitted: February 2025

been interfered by the subsequent one. This phenomenon is known as *Catastrophic Forgetting* (or *Catastrophic Interference*), and is one of the main problems that DNNs face.

The two first ways of mitigating forgetting that could come to mind are usually to train the whole model from scratch, using D_A and D_B, or even training the already initialized model with a replay of D_A, along with D_B, but these methods may not always be possible due to time constraints or storage limits, as it is time-consuming and all data should be saved, which increases costs, among other things. The challenge that *Continual Learning* faces when developing models to mitigate *Catastrophic Forgetting* is usually generalized with the *Stability-Plasticity Dilemma* [5], that refers to the inability of a model to, both, be stable, not forgetting old information, and plastic, that is, learning and adapting to new data.

II. State of the art

Although the general concept of machines learning over time without forgetting is commonly known as *Lifelong Learning* (LL) [6], [7], its application to DNNs [8] is also referred to as *continual learning* [9], where, as already mentioned, one of its main focuses is dealing with *catastrophic forgetting* [10].

In *Chen et al.* [11], five key characteristics of lifelong learning systems are presented:

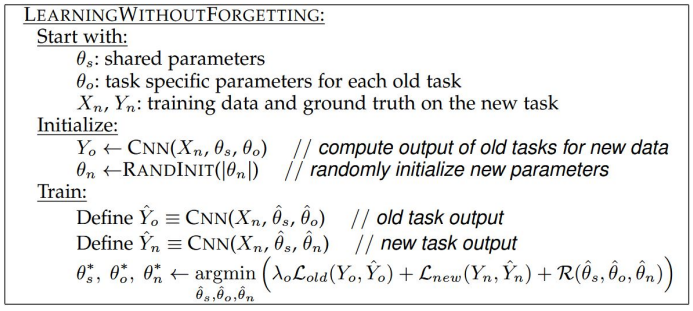
1. Continuous learning process.
2. Knowledge accumulation and maintenance of it.
3. The ability to use past knowledge to enhance future learning.
4. The ability to discover new tasks.
5. The ability to learn while working (treating a past task as a new one, and the rest of the tasks as previous ones.)

A. Continual Learning Approaches on NN

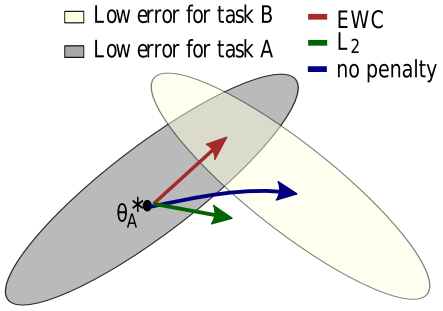
Many different approaches have been studied on neural networks to alleviate catastrophic forgetting. Although this field is still in an early stage of progress, in *Parisi et al.* [8] the authors divide these approaches into three main types:

1) Regularization approaches: The strategy is to train the whole network when new data arrive, but with a factor that penalizes changes to attenuate forgetting. These methods primarily focus on influencing the loss function with regularization terms, and tend to be more effective in domain-incremental scenarios.

One such approach is presented in *Learning without Forgetting (LwF)* [12], see Figure 1a, on a CNN with shared, θ_s, and task-specific, θ_o, parameters, that only trains on new data X_n, but takes into account the loss of the output of old tasks with new data Y_o at a rate λ_o (*stability-plasticity parameter*).



(a) Learning without Forgetting



(b) Elastic Weight Consolidation

Fig. 1: (a) LwF training procedure when new training data and new task appears. (b) Training trajectories in a schematic parameter space (EWC).

Another influential regularization approach is *Elastic Weight Consolidation (EWC)* [13], which, assuming over-parameterization [14], [15], tries to obtain a solution for task B, θ^*_B, near the previous solution for task A, θ^*_A, that reduces interference (see Figure 1b), and it does so by penalizing the deviations from θ^*_A based on the diagonal entry of the *Fisher Information Matrix* of the learned predictor evaluated at θ^*_A, F_i, and, as in the previous approach, with λ as the *stability-plasticity* parameter.

$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{1}$$

A way of computing the importance of different weights was presented in *Zenke et al.* through the *Synaptic Intelligence* approach [16], based on the logic idea that a parameter's importance is proportional to its contribution to the loss decrease over time.

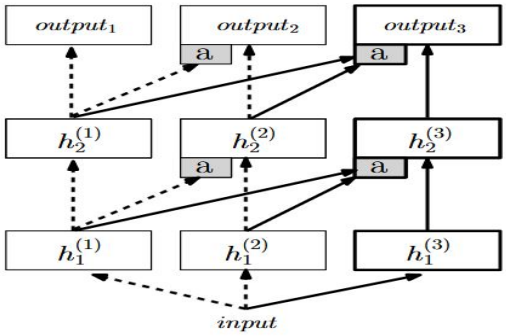


Fig. 2: A progressive network already trained on two tasks, 1 and 2, with hidden layers $h_i^{(1)}$ and $h_i^{(2)}$ respectively, and a third column created for task 3, being influenced also by the frozen columns through adapted layers a (see the paper for more)

2) Dynamic Architectures: Instead of tweaking the values of the weights and biases, dynamic architecture approaches modify the structure and resources of the network to adapt to new tasks, e.g., training again with a new layer and new connections. *Rusu et al.* [17] use what they called a *progressive network*, shown in Figure 2, where the DNN, after training at a task A with L layers, $h_i^{(A)} \in \mathbb{R}^{n_i}$ hidden activations, and n_i the number of units at layer $i \leq L$, and obtaining converged parameters θ^A, these last are *frozen* as task B training starts, adding new neurons with new parameters θ^B, randomly initialized, having h_i^B also depend on the representations learned by the previous tasks neurons, expecting that all the learning done so far could be helpful in learning new tasks.

One of the main drawbacks of this approach is the obvious increase in storage costs if new tasks keep appearing. *Mallya et al.* presented, initially with *PackNet* [18], and later refined with *Piggyback* [19], *weight mask* approaches, a very efficient way of achieving zero-forgetting, both in terms of computation and storage (especially if the mask values are transformed to binary values, as shown in Figure 3) with the drawback of not having a real notion of knowledge transfer over time, and the necessity of task labels. It starts from a pre-trained model (backbone) and adds a mask for each weight of every particular task.

3) Complementary Learning Systems and Memory Replay: Also known as *Rehearsal strategies*, these kinds of approaches take inspiration from CLS theory [20], [21], where the hippocampus is responsible for fast learning of arbitrary information and the neocortex for generalizing and giving a structure to the knowledge, as in *Shin et al* [22] with the *Deep Generative Replay*, where a deep generative model is complemented with a task solver model. The first is trained to output synthetic data with the same distribution as new data x and replayed data x' from prior generators, and the last trains on the new task pairs (x, y) and artificial (x', y') pairs, where y' is obtained by feeding x' to the previous solver (see Figure 4). Note that, ensemble model's common notation of teacher-student is replaced by *scholar*, as each model can both learn and 'teach' others.

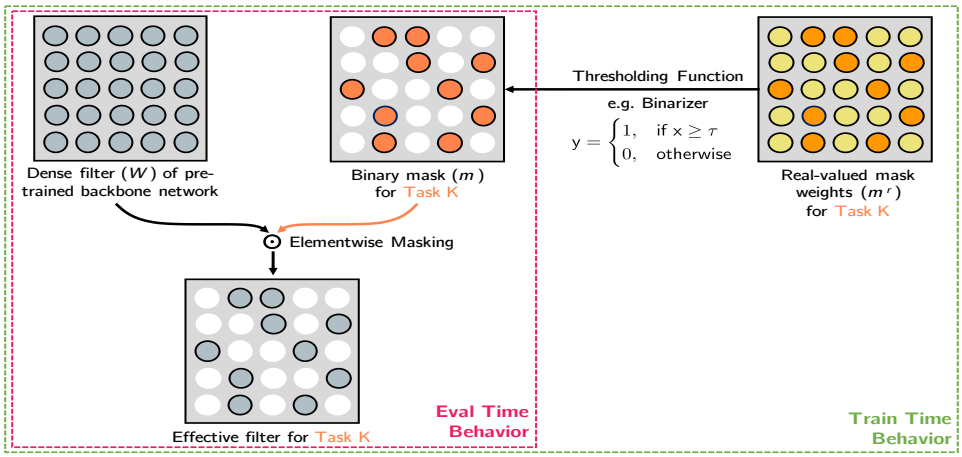


Fig. 3: Piggyback mask approach. Overview of the method applied to learn binary masks for fixed backbone networks

Some months prior to *Shin et al.* model, *Rebuffi et al.* presented *iCaRL* (*incremental Classifier and Representation Learning*) [23]; a *class incremental* strategy that simultaneously learns classifiers and feature representations, with a fixed memory limit. Each time a new image appears, it is classified using the *mean-of-examplars* classifier. *Examplars* consists of sets of dynamically selected images that belong to the same class, with a constant K as the total maximum of images stored. During the training phase, iCaRL's update routine is called, where it modifies its parameters and exemplars based on the current training data, learning at the same time if a new class is present (see the paper for a more in-depth explanation).

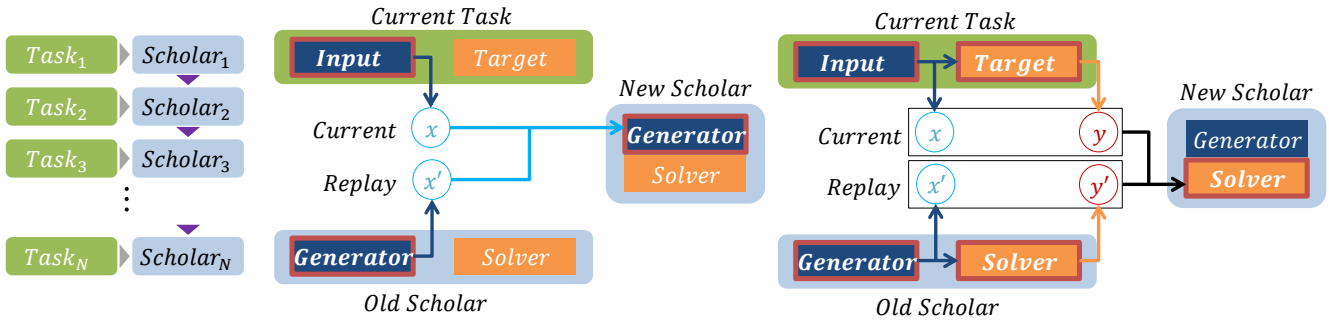
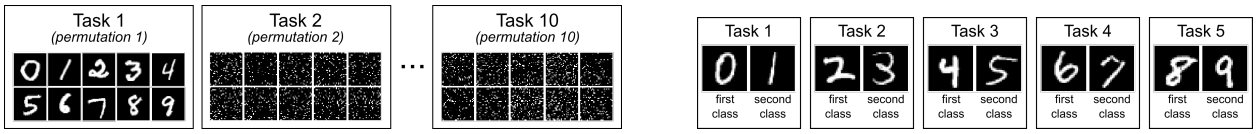


Fig. 4: Incremental Classifier and Representation Learning (iCaRL) schematic.

B. Evaluation Methods and Metrics

Although there is no consensus on a unique group of methodologies and metrics to evaluate lifelong learning strategies, some of the most common ones are presented. In terms of **benchmarks**, in *Kemker et al.* [24] the authors divide them into three families of scenarios:

1) *Data Permutation Experiment*: With an even split of data, each feature vector is permuted randomly, keeping the same permutation order on each task; that is, each task's features are transformed equally vector-wise, and different tasks have different transformations (Figure 5a).
2) *Incremental Class Learning*: A base task set is learned, and is followed by experiences with n new classes (Figure 5b).
3) *Multi-Modal Learning*: Tasks consist of different types of data, e.g., first learn image classification and then audio classification (Figure 6).



(a) Data Permutation Experiment (Permuted MNIST) (b) Incremental Class Learning (Split MNIST)

Fig. 5: Permuted and Split MNIST benchmarks

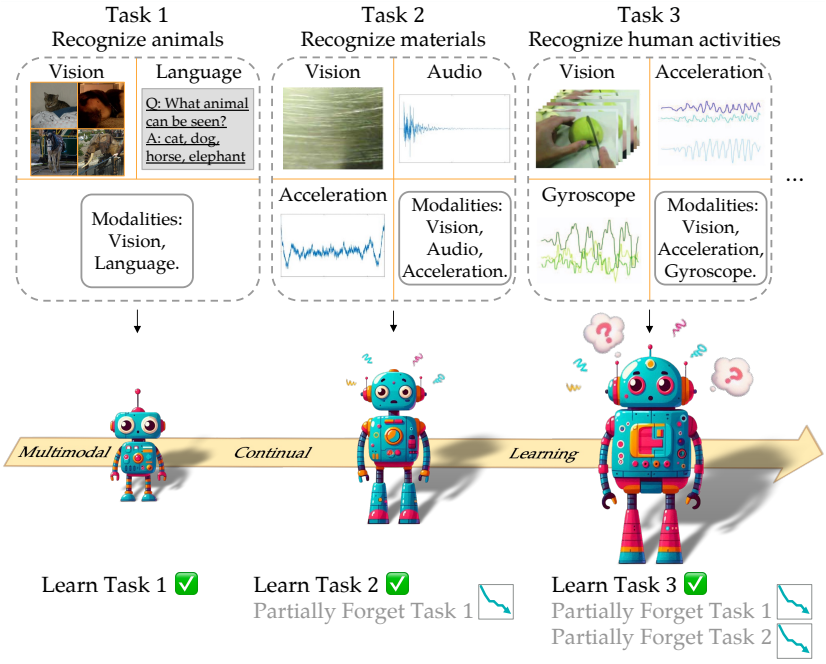


Fig. 6: Multi-Modal Learning example [25]

Depending on the availability of tasks at test time, a continual learning **scenario** [26] can be classified as:

1) *Task-incremental learning (Task-IL)*: The task identifier is always available for the model. Therefore, it is the easiest scenario, as it is possible to use architectures that benefit from this feature (usually with a "multi-headed" output layer).
2) *Domain-incremental learning (Domain-IL)*: The task identifier is neither available nor needed to infer, as the only objective to solve is the current task.
3) *Class-incremental learning (Class-IL)*: Without task identifiers, the model should be capable of solving both current and past tasks, plus inferring the tasks it encounters. It represents most real-world problems, as new classes incrementally appear and have to be learned.

Task-IL	Given permutation X, which digit?
Domain-IL	With permutation unknown, which digit?
Class-IL	Which digit *and* which permutation?

TABLE I: Permuted MNIST according to each scenario.

Task-IL	With task given, is it the 1^{st} or 2^{nd} class? (e.g., 0 or 1)
Domain-IL	With task unknown, is it a 1^{st} or 2^{nd} class? (e.g., in $[0, 2, 4, 6, 8]$ or in $[1, 3, 5, 7, 9]$)
Class-IL	With task unknown, which digit is it? (i.e., choice from 0 to 9)

TABLE II: Split MNIST according to each scenario.

Hyperparameter selection is another confronted topic. One way of operating would be to maximize the model performance based on the results of the stream of training experiences, or even doing multiple runs with a randomization in the experiences' order. Yet, this course of action would rarely be possible in many real-world continual learning scenarios. *De Lange et al.* [9] proposed a *Continual Hyperparameter Framework* that avoids the common practice of using grid search with validation data from all tasks, assuming that the only available information is the one contained in the actual experience. It starts with the initialization of all hyperparameters *H* forgetting-related to their most 'stable' tune possible (generally, its highest value), trying to avoid interference. If a predefined performance threshold is not surpassed, values are decreased until they do, pursuing the best *stability-plasticity trade-off*.

Once models are tested, several factors must be considered to measure their viability in real world scenarios, both in terms of computational(and storage) efficiency and performance effectiveness. Three generally used **metrics** [27] are:

$$\text{Average Accuracy: } ACC = \frac{1}{T}\sum_{i=1}^{T} R_{T,i} \tag{2}$$

$$\text{Backward Transfer: } BWT = \frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i} \tag{3}$$

$$\text{Forward Transfer: } FWT = \frac{1}{T-1}\sum_{i=2}^{T} R_{i-1,i} - \bar{b}_i. \tag{4}$$

with T as the number of tasks, t_i, $R \in \mathbb{R}^{T\times T}$ as the matrix of accuracies, where $R_{i,j}$ is the accuracy of the model on t_j test set after being trained on t_i, and $\bar{b}$ the vector for test accuracies for each task on random initialization. Equations 3 and 4 show the influence that learning a new task t_i has on the performance of a previous task t_{i-1} (BWT), and a future task t_{i+1} (FWT). Note that if $FWT > 0$, the model would be able to perform *zero-shot learning* [28]. In *Rodriguez-Diaz et al.* [29], greater granularity is introduced to these metrics, measuring performance at every *timestep i in time*, and other metrics are added, like *model size efficiency*, *samples storage size efficiency* and *computational efficiency*.

C. Dataset Distillation

In *Wang et al.* [30], *dataset distillation* is introduced as an alternative formulation of the already widely used *network distillation* [31], where a less computing-demanding and much easier-to-deploy model is obtained by compressing the knowledge of an ensemble. *Dataset distillation* (DD), sometimes also called *dataset condensation* (DC), as its name suggests, changes the objective of compression to the dataset, aiming for the performance of training with the original. The compressed dataset is usually formed by *synthetic samples*, sometimes with as few as a single instance per class.

Generally, distilled data do not bear much resemblance to real data of the same class (see Figure 7, 10 and 12), and it is shown to decrease performance if mixed during training with real data [32], or if used with a different network architecture than the one it was distilled with [33] (although the authors present some interesting and effective approaches to mitigate it).

Formally, a dataset $D = (x_i, y_i)_{i=1}^{m}$ is reduced to a *data summary* $D_{syn} = (\tilde{x}_i, \tilde{y}_i)_{j=1}^{n}$, with $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, as the input features and its labels respectively. DD aims to attain similar performance with $n << m$, where:

$$\sup \quad \Big\{ \mid l(\Phi_{\theta^D}(x), y) - l(\Phi_{\theta^{D_{syn}}}(x), y) \mid \Big\}_{\substack{x\sim\mathcal{X} \\ y\sim\mathcal{Y}}} \leq \epsilon \tag{5}$$

considering $\Phi_{\theta^K} : \mathcal{X}_\mathcal{K} \rightarrow \mathcal{Y}_\mathcal{K}$ a learning algorithm dependent on the optimal set of parameters θ^K on a dataset K, and $\epsilon \in \mathbb{R}^+$ as a threshold of acceptable discrepancy between the training dynamics of D_D and D_{syn}, and data distillation defined as:

$$\underset{D_{syn},n}{\arg\min} \left(\sup \quad \{ \mid l(\Phi_{\theta^D}(x), y) - l(\Phi_{\theta^{D_{syn}}}(x), y) \mid \}_{\substack{x\sim\mathcal{X} \\ y\sim\mathcal{Y}}} \right) \tag{6}$$

To improve robustness and generalization, several DD algorithms take expectation over different initializations $\theta^{(0)}$ [34].

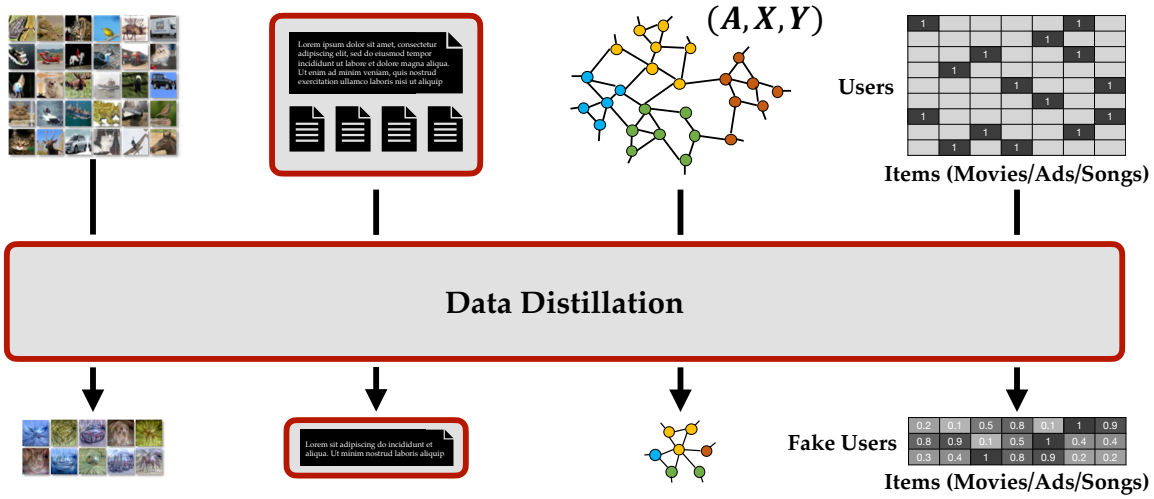


Fig. 7: Examples of how distilled datasets may outcome in different data modalities [35]

Sachdeva and McAuley [35] divide DD approaches into four categories (see the paper for more details, such as method variants, assumptions taken in each framework, and main drawbacks of each approach):

1) *Meta-Model Matching* : Where *Meta-learning* (*learning to learn* [36]) denotes ML algorithms that learn from the output of other ML algorithms [37], these approaches perform a bi-level optimization problem (see Eq. 7), using the distilled data as hyperparameters, where the outer loop updates the data summary for transferability, and the inner loop performs the supervised learning problem based on D_{syn}, generally using gradient descent (GD) for NN or regression for kernel method until convergence. Based on this inner loop, [34] further classify meta-learning frameworks in two subgroups; *Backpropagation Through Time*(BPTT) approaches [30], [38], [39] , and *Kernel Ridge Regression*(KRR) approaches [40], [41].

$$\underset{D_{syn}}{\arg\min} \quad \mathcal{L}_\mathcal{D}(\theta^{D_{syn}}) \quad s.t. \quad \theta^{D_{syn}} = \underset{\theta}{\arg\min} \quad \mathcal{L}_{D_{syn}}(\theta) \tag{7}$$

A common flaw of these class of methods (much less noticeable in KRR than in BPTT, due to the replacement of the NN with a kernel model in the inner loop) is that they are both computationally expensive and memory demanding [22].

2) *Gradient Matching*: Originally introduced by *Zhao et al.* (DC) [42], this method rests on the insight that if two datasets produce similar gradient signals during training, then the models trained on these datasets are likely to generalize similarly. By matching gradients, this method implicitly captures both class-level and instance-level information without requiring a full forward and backward pass over the entire original dataset at each step. A crucial advantage of this approach lies in its memory efficiency: the optimization leverages mini-batch gradient descent, which reduces the computational overhead compared to bi-level optimization schemes.

However, despite these advantages, Gradient Matching has its own limitations. One significant challenge is the potential for gradient variance over training steps, particularly when using deep neural networks with non-convex loss surfaces. This could lead to instability during the optimization process, as small perturbations in the synthetic dataset might amplify over time, resulting in suboptimal convergence or overfitting to the training dynamics of the original dataset rather than generalizable patterns. The optimization problem is expressed as:

$$\underset{D_{syn},\eta}{\arg\min} \quad \mathbb{E}_{\substack{\theta_0 \sim \mathbf{P}_\theta \\ c \sim \mathcal{C}}} \left[\sum_{t=0}^{T} \mathbf{D}\left(\nabla_\theta \mathcal{L}_{D^c}(\theta_t), \nabla_\theta \mathcal{L}_{D^c_{syn}}(\theta_t) \right) \right] \quad \text{s.t.} \quad \theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla_\theta \mathcal{L}_{D_{syn}}(\theta_t), \tag{8}$$

where T reflects the model's similarity projected T-steps forward, η the learning rate, C the set of unique classes in $\mathcal{Y}$, and $D : \mathbb{R}^{|\theta|} \times \mathbb{R}^{|\theta|} \rightarrow \mathbb{R}$ denotes a chosen distance metric (commonly the cosine distance).

3) *Trajectory Matching*: Presented by *Cazenavette et al.* [43], TM approaches build upon the foundations of GM but introduces a temporal dimension to the alignment process. Rather than focusing solely on matching gradients at a fixed training step, TM seeks to align the entire sequence of model states as they evolve through training on the original and synthetic datasets. This approach is grounded in the observation that a synthetic dataset that produces a similar sequence of model updates to the original dataset is more likely to capture its essential characteristics. To achieve this, it minimizes the discrepancy between the parameter trajectories of models trained on the original dataset and those trained on the synthetic dataset, normalized by the distance the model travels through the parameter space over a fixed number of steps. Formally, the method seek matching D and D_{syn} training trajectories, $\{\theta_t^D\}_{t=0}^T$ and $\{\theta_t^{D_{syn}}\}_{t=0}^T$, respectively, represented in:

$$\begin{gathered} \underset{D_{syn},\eta}{\arg\min} \quad \mathbb{E}_{\theta_0 \sim \mathbf{P}_\theta} \left[\sum_{t=0}^{T-M} \frac{\mathbf{D}\left(\theta_{t+M}^D, \theta_{t+N}^{D_{syn}}\right)}{\mathbf{D}\left(\theta_{t+M}^D, \theta_t^D\right)} \right] \\ \text{s.t.} \quad \theta_{t+i+1}^{D_{syn}} \leftarrow \theta_{t+i}^{D_{syn}} - \eta \cdot \nabla_\theta \mathcal{L}_{D_{syn}}(\theta_{t+i}^{D_{syn}}) \quad ; \quad \theta_{t+1}^{D_{syn}} \leftarrow \theta_t^D - \eta \cdot \nabla_\theta \mathcal{L}_{D_{syn}}(\theta_t^D), \end{gathered} \tag{9}$$

with $D : \mathbb{R}^{|\theta|} \times \mathbb{R}^{|\theta|} \rightarrow \mathbb{R}$ as a distance metric (usually L2). The normalization term is essential, as it prevents trivial solutions where the synthetic dataset might match the trajectory over short horizons but diverge over longer ones. This methodology introduces greater robustness and generalization capacity compared to single-step GM, as it encourages the synthetic dataset to reflect not just the static information present in the original data but also its dynamic properties throughout the training process, although this comes with a noticeable increase in computational demands.

4) *Distribution Matching*: DM offers a more scalable alternative to Meta-Model Matching, GM, and TM by shifting the optimization paradigm. Instead of aligning gradients or training trajectories, DM focuses on minimizing the statistical discrepancy between the original dataset, D, and the synthetic dataset, D_{syn}, in a learned, low-dimensional representation space, with the assumption that two alike datasets based on a specific distribution divergence metric yield similarly trained models. *Zhao et al.* introduced DM [44] as a method that employs both parametric encoders to map high-dimensional data into low-dimensional spaces and Maximum Mean Discrepancy to evaluate the distribution mismatch between D and D_{syn} in these spaces. The objective function, given a set of k encoders $\mathcal{E} \triangleq \{\psi_i : \mathcal{X} \mapsto \mathcal{X}_i\}_{i=1}^k$ and the $\mathcal{C}$ set of unique classes in $\mathcal{Y}$, can be formulated as:

$$\underset{D_{syn}}{\arg\min} \quad \mathbb{E}_{\substack{\psi \sim \mathbb{E} \\ c \sim \mathcal{C}}} \left[\left|\left| \mathbb{E}_{x \sim D^c}[\psi(x)] - \mathbb{E}_{x \sim D^c_{syn}}[\psi(x)] \right|\right|^2 \right] \tag{10}$$

While this improves scalability and training efficiency, the quality of the distilled dataset depends heavily on the choice of encoder and distribution divergence metric. An insufficiently expressive encoder may fail to

capture important data variations, while an inappropriate divergence measure could lead to poor alignment, especially when dealing with complex, multi-modal datasets.

Dataset Distillation has increasingly found its place within the field CL, as the ability to store compact yet representative synthetic datasets offers a natural solution to many open-world constraint scenarios, fitting specially well with replay strategies. One such example is Distilled Replay (DR), introduced by Rosasco et al. (2022), where instead of storing raw samples from past tasks, DR leverages a dataset distillation technique to construct small but informative synthetic datasets, which are then replayed during training to maintain performance on previously seen data. This approach not only mitigates storage constraints but also provides a more efficient and targeted memory mechanism, as the distilled datasets are optimized to capture the essential training dynamics required for effective knowledge retention.

In more detail, DR falls under the family of dual memory strategies, similar to the previously mentioned *iCaRL*. In this case, it is formed by a small buffer of synthetic samples and a simple replay policy. The distillation process is based on the original Dataset Distillation by *Wang et al.*, with two main differences:

- The learning rate η is fixed, as distilled examples generated to work with a specific learning rate could diminish their effectiveness when used with a different one. Thus, for consistency, the distillation process employs the same learning rate that the model uses during continual learning.
- It uses a different loss function. Instead of evaluating the loss on a minibatch of training data at the final inner step, the loss is computed at each step, and the sum of these losses is backpropagated. As a result, buffer distillation is equivalent to backpropagating the gradient at each inner step, with the distilled images being updated once the inner training process is complete. (See Figure 8)

A significant advantage of DR is its adaptability to the continual learning scenario, where both the model and the distilled dataset evolve dynamically. As new tasks arrive, the synthetic dataset can be incrementally updated, ensuring it remains representative of the shifting data distribution. This dynamic memory buffer allows models to mitigate the adverse effects of concept drift and task interference.

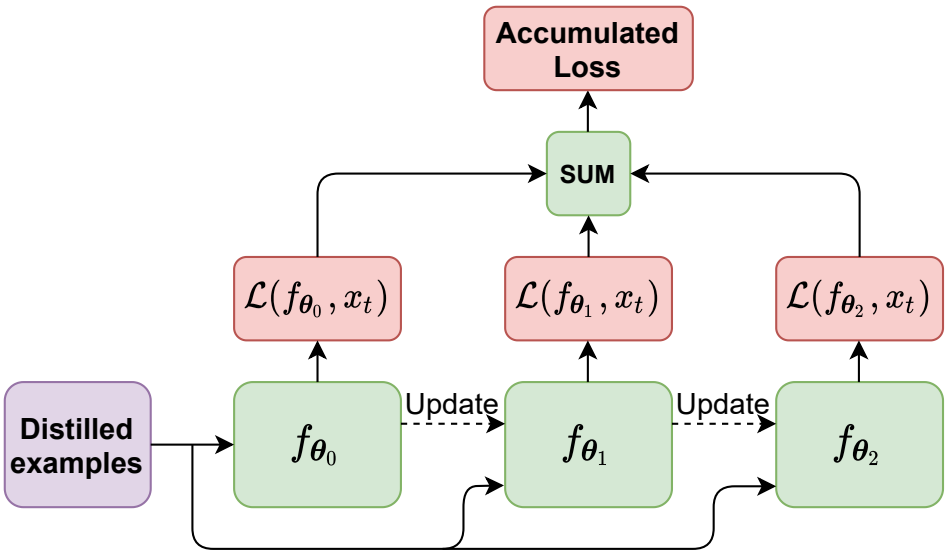


Fig. 8: Buffer Distillation update in Distilled Replay. The gradient is backpropagated to the distilled examples along the computation path defined by the solid arrows. The accumulated loss represents the sum of the losses computed at each step.

III. METHOD

A. A more efficient Distilled Replay

DR obtained state-of-the-art accuracies in a class-incremental scenario with a replay buffer in which only 1 sample (distilled) per class is allowed.

We employ a distributed matching technique rather than the approach originally used in Distilled Replay, seeking computational efficiency.

In our implementation, we begin by computing the gradients of the loss with respect to the model parameters for both real data and synthetic data stored in the replay buffer. The gradients obtained from real data reflect the true learning signal, while those computed from synthetic data provide a proxy that we wish to align with the true gradients. This alignment is achieved by evaluating the Euclidean distance between the gradient vectors of corresponding layers, thus quantifying the discrepancy between the two sets of gradients without resorting to any discrete or enumerated metrics. The individual layer discrepancies are then summed to form a single, aggregated gradient matching loss. Furthermore, this process is repeated across multiple model initializations, with the resulting discrepancies averaged to ensure that the synthetic data consistently approximates the behaviour of the real data over a range of initial conditions. The resulting total gradient difference is then backpropagated to update the synthetic data, driving the synthetic samples to induce gradients that closely match those derived from real data.

The theoretical background of our implementation is as follows:

We compare the gradient on a per-layer basis. For each corresponding layer i, the Euclidean (L2) norm of the difference between the gradients is calculated:

$$\mathcal{D}_i = \left\| \nabla_{\theta_i} \mathcal{L} - \nabla_{\theta_i} \mathcal{L}_{\text{syn}} \right\|_2. \tag{11}$$

where $\nabla_\theta \mathcal{L}$ and $\nabla_\theta \mathcal{L}_{\text{syn}}$ are the gradients with respect to the model parameters for a given mini-batch, of real data and synthetic data, respectively.

The total gradient matching loss is then obtained by summing the per-layer discrepancies:

$$\mathcal{L}_{\text{m}} = \sum_{i=1}^{L} \mathcal{D}_i. \tag{12}$$

This cumulative loss serves as an objective function to minimize during the optimization of the synthetic data. Minimizing $\mathcal{L}_m$ ensures that the synthetic data is refined to produce gradient dynamics similar to those of the real data.

To enhance robustness, our method computes the gradient matching loss over multiple model initializations. For each initialization θ_0 drawn from a distribution $P(\theta_0)$, the gradient discrepancy is evaluated and then averaged. This process can be mathematically expressed as:

$$\arg\min_{S} \quad \mathbb{E}_{\theta_0 \sim P(\theta_0)} \left[\mathcal{L}_{\text{m}}(\theta_0, D_{syn}) \right]. \tag{13}$$

Finally, the computed gradient matching loss is backpropagated to update the synthetic data S in the replay buffer. This update process ensures that the gradients induced by S converge toward those induced by the real data, thus effectively encapsulating the essential learning dynamics required for continual learning.

A highly beneficial key that our method shares with DR is that, unlike similar continual learning approaches, such as GEM [27], where most of the computational overhead occurs during training, the distillation process operates separately from the main learning phase. This separation allows Buffer Distillation to run in parallel with training as soon as a new experience is introduced.

B. Experiments

All experiments were conducted on the MNIST dataset, a well-established benchmark consisting of 60.000 grayscale 28x28 pixel images of handwritten digits. This dataset is typically divided into training and testing subsets. In this work, it is further split into 5 distinct experiences to simulate a class-incremental scenario, with 2 classes in each experience, as previously shown.

Additionally, our proposed method was also tested with CIFAR-10, another very popular dataset of 60,000 32x32 color images labeled into 10 classes, used for image classification and deep learning benchmarks, which was also used as a class-incremental scenario with 2 classes per experience.

In Appendix A, some initial experiments were conducted to compare the effectiveness of regularization and replay approaches. Regularization-based methods fail to mitigate catastrophic forgetting in strict class-incremental learning. Replay methods, even with small buffers, significantly improve accuracy. Adaptive buffers perform better than fixed ones, and larger buffers approximate cumulative training, making them preferable when storage allows.

Following the methodology described in DR's paper, we use a LeNet5, a modified version of the LeNet by *LeCun et al.* [45], that incorporates two convolutional layers with 5×5 kernels, each followed by subsampling layers to reduce spatial dimensions. The extracted features are processed by fully connected layers before classification through a softmax output.

The six approaches of the Distilled Replay paper plus our modified version considered are: a cumulative and a naive approach, serving as the upper and lower bounds; both the original distilled replay and our modified version with distribution matching; a simple replay method (similar to the ones used in Appendix A); the incremental Classifier and Representation Learning (iCaRL) described on previous sections; and the Maximal Interfered Retrieval (MIR) [46] method, a more refined memory replay technique that prioritizes storing and revisiting examples that are most susceptible to being forgotten. Instead of sampling past data randomly, MIR selects examples that exhibit the highest degree of interference from newly learned information, ensuring that the model reinforces the most at-risk knowledge while learning new tasks.

IV. Results

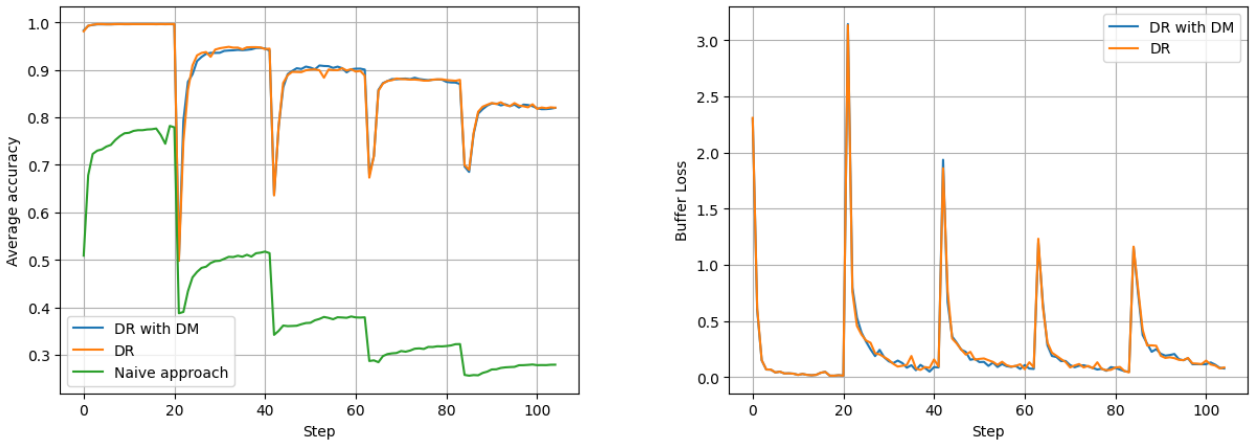
Results obtained replicating the experiments from the original paper produced the same results on the naive and the DR approach due to the use of the same seeds. All other methods (Cumulative, Simple Replay, iCaRL and MIR) accuracy results were copied from the same source (on CIFAR10, DR results were also obtained directly from the original paper).

On MNIST, our DR variation with distribution matching exhibits **similar performance** to the original approach (see Table III), while reducing the time to distill the data of one task of the experience (formed by 2 classes) from an average[1] of 1632 to 830 seconds, **an almost 50% decrease in computing time**.

Figure 9a shows the almost identical behaviour of DR with the DM version, and how the naive approach quickly falls due to the lack of stability. The sudden drops in accuracy, forming 'mountain-like' patterns, are caused by the new experiences arrival. In Figure 10, the images obtained from the two different dataset distillation approaches used can be visualized.

	UB	LB	DR	DR*	SR	iCaRL	MIR
E_2	.96	.49	.93	**.94**	.88	.91	.92
E_3	.94	.33	.89	**.90**	.70	.85	.78
E_4	.93	.25	**.87**	**.87**	.66	.81	.68
E_5	.91	.20	**.87**	**.82**	.61	.77	.59

TABLE III: Average accuracies over trained experiences on the split version of MNIST. UB as upperbound (Cumulative), LB as lowerbound (Naive), DR as Distilled Replay, DR* as our modified version of DR with distribution matching, SR as simple replay, iCaRL as incremental Classifier and Representation Learning, and MIR as Maximal Interfered Retrieval. In bold the top performing values, excluding the ones from the cumulative approach.



(a) Average Accuracy

(b) Buffer Loss

Fig. 9: (a) Comparison of accuracies between DR, the Naive approach and our DR with distribution matching on the split MNIST benchmark. (b) Comparison of the buffer loss between the distillation methods of DR and our approach on the MNIST benchmark

[1]Using a Ryzen 7600k, an Nvidia RTX 4070 Super and 32GB of DDR5 RAM

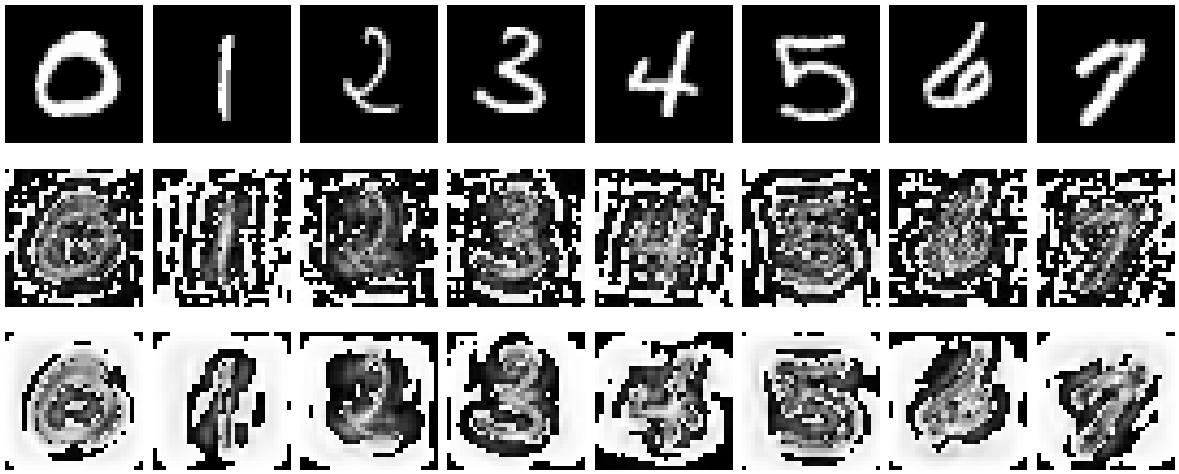


Fig. 10: Comparison of MNIST digit representations obtained during our experiments. The first row shows the original MNIST samples. The second row presents images generated using our distribution matching method, while the third row displays images obtained through the Distilled Replay approach.

On CIFAR10, our modified method obtained **higher accuracies** than its original approach (see Figure IV, indicating that it might mitigate catastrophic forgetting more effectively, while also being more time-efficient. Figure 11 compares the accuracy evolution over the whole benchmark with the Naive approach, which displays the same behaviour mentioned above, and Figure 12 shows some distillation examples obtained using the proposed method.

	UB	**LB**	**DR**	**DR***	**SR**	**iCaRL**	**MIR**
E_2	.56	.28	**.52**	.51	.43	.41	.49
E_3	.43	.21	.34	**.38**	.29	.29	.32
E_4	.38	.18	.28	**.32**	.21	.23	.18
E_5	.35	.14	.24	**.28**	.19	.21	.19

TABLE IV: Average accuracies over trained experiences on the split version of CIFAR10. UB as upperbound (Cumulative), LB as lowerbound (Naive), DR as Distilled Replay, DR* as our modified version of DR with distribution matching, SR as simple replay, iCaRL as incremental Classifier and Representation Learning, and MIR as Maximal Interfered Retrieval. In bold the top performing values, excluding the ones from the cumulative approach.

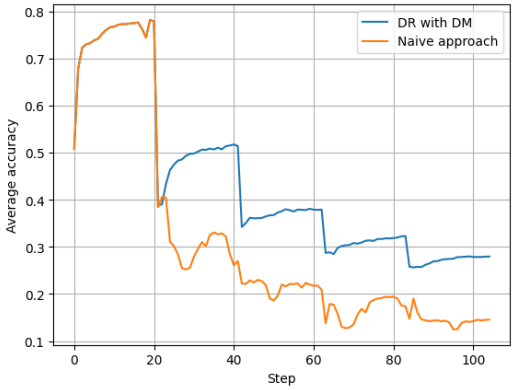


Fig. 11: Comparison of accuracies between the Naive approach and our DR with distribution matching on the split CIFAR10 benchmark

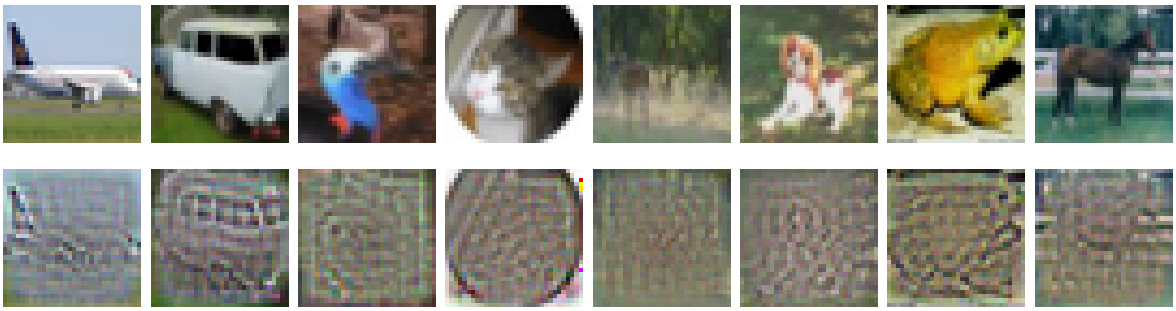


Fig. 12: Comparison of CIFAR10 images obtained during our experiments. The first row shows the original CIFAR10 samples. The second row presents images generated using our distribution matching method.

V. Conclusions

While the results are encouraging, further evaluation is necessary to confirm the robustness and generalizability of the proposed method. Future work should include experiments on more complex datasets with higher visual detail and a greater number of classes. Additionally, testing across different continual learning scenarios, such as Domain-IL and Task-IL, would provide a more comprehensive assessment of its effectiveness compared to existing approaches.

The proposed approach and the original one present effective solutions to storage-constrained scenarios, although its computational requirements may not be viable in many real-world environments. Future advancements (and current ones, not used in this paper) in dataset distillation techniques could greatly enhance the landscape of continual learning. More efficient distillation methods would allow for even smaller and more informative synthetic datasets, reducing both memory usage and training time while preserving key data characteristics.

While continual learning is not yet on par with offline learning in terms of stability and performance, ongoing research will help bridge this gap. Advances in memory retention strategies, adaptive architectures, and optimization techniques are steadily improving the ability of neural networks to learn incrementally without catastrophic forgetting. As these methods evolve, continual learning will become more reliable and applicable to real-world scenarios, enabling AI systems to adapt dynamically to new information while preserving past knowledge.

APPENDIX A
REGULARIZATION AND SIMPLE REPLAY APPROACHES

A multilayer perceptron (MLP) model is employed, implemented with PyTorch and augmented using the Avalanche library [47]. The model initially projects the input features into a 200-dimensional hidden space using a linear transformation, followed by a ReLU activation and dropout for regularization. Additional hidden layers are then sequentially added, each following the same pattern of linear transformation, activation, and dropout.

The final classification layer is designed to adapt to different continual learning settings. It can either operate as a standard classifier for a fixed set of outputs or as an incremental classifier that accommodates evolving output dimensions as new experiences are introduced.

During the forward pass, the model reshapes and processes the input data through the feature extraction layers before generating predictions via the classifier. Optimization is achieved using a gradient-based optimizer (SGD with momentum), with a cross-entropy loss function guiding the learning process. The training process is set for a maximum of 20 epochs and incorporates early stopping [48] with a patience of 5 epochs to reduce overfitting.

The Naive approach is used to show the worst performing baseline. It incrementally fine-tunes a single model on each new task without implementing any mechanisms to mitigate catastrophic forgetting.

For the Learning without Forgetting (LwF) experiment, knowledge distillation was applied to preserve knowledge from previous tasks. The LwF plugin used alpha values of [0, 0.5, 1.33333, 2.25, 3.2] to control the weight of the distillation loss across experiences, and a temperature of 2 was used to smooth the output probabilities, enhancing the transfer of knowledge between tasks.

The Elastic Weight Consolidation (EWC) experiment introduced regularization with a lambda value of 1 to penalize changes to parameters that are critical for prior tasks. The 'separate' mode was employed to maintain individual regularization terms for each experience, ensuring distinct importance tracking for all tasks. No decay factor was applied, meaning the penalty remained constant throughout training.

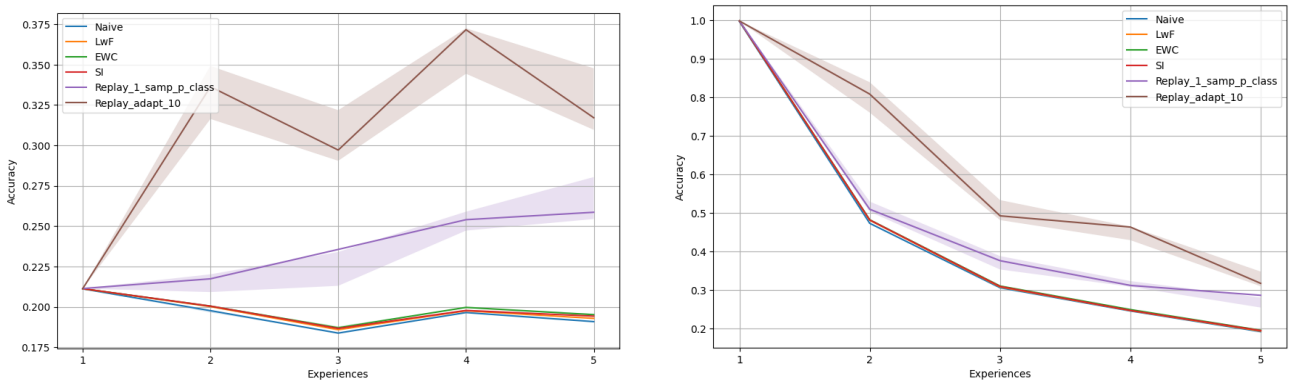
For the Synaptic Intelligence (SI) experiment, the regularization coefficient (λ) of 1 was used to determine how strongly parameter changes were constrained based on their historical contributions to task performance. An ϵ value of 0.1 was added to improve numerical stability during the importance calculations.

Finally, a simple Replay Method with a very small buffer (10 samples), with 2 variants. The first one keeps a strict sample per class. The second adapts the number of samples per class to the buffer size, proportionally (pe, during the second experience, when classes 2 and 3 appear, the buffer would be composed of 5 samples of class 0 and 5 of class 1 [0:5, 1:5]; in the next experience, [0:3, 1:3, 2:2, 3:2]; ...).

Figure 13a shows the overall test accuracy after each experience in the class-incremental setup. As new classes arrive, most methods undergo a substantial performance drop, illustrating the challenge of catastrophic forgetting. Indeed, by the final experience, the regularization approaches barely exceed 20% average accuracy on the full dataset, and do not make a difference with the naive approach, confirming prior findings [26] that regularization-based methods tend to fail in strict class-incremental settings.

In contrast, even a minimal replay buffer significantly mitigates forgetting, as seen in the replay curves. Although these curves exhibit some variability across experiences, the replay-based approaches consistently retain higher accuracy overall. Notably, they outperform the regularization-only methods by a clear margin in the final experiences.

Figure 13b offers a complementary view, plotting the average accuracy only on the classes encountered so far. Here, all methods begin with near-perfect accuracy on the first task (as it is a simple binary classification) but show pronounced declines as new experiences are introduced.



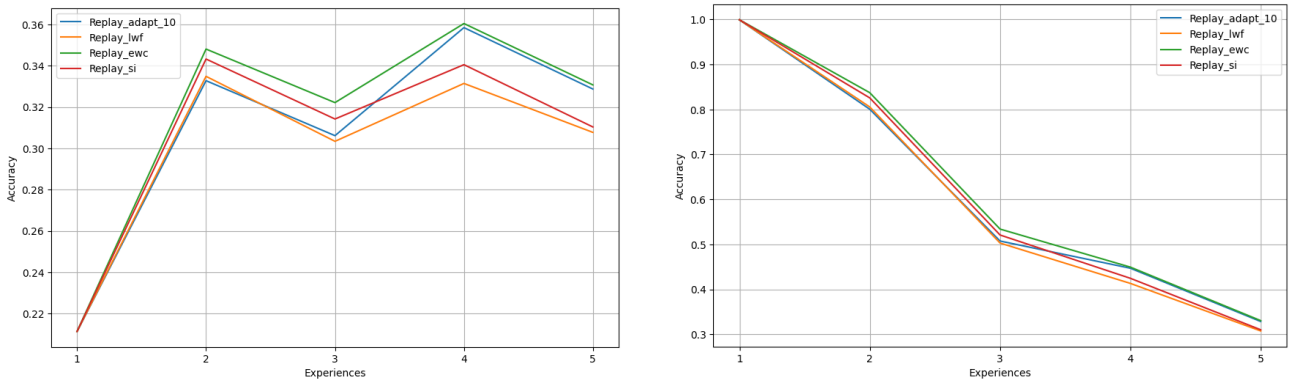
(a) Model average accuracies over the full stream of data

(b) Model average accuracies over trained data

Fig. 13: Comparison of the naive approach, regularization methods (LwF, EWC and SI) and simple replay (Replay_1_samp_p_class: Simple replay with strictly 1 sample per class; Replay_adapt_10: buffer of 10 samples, adaptive.)

The higher rebounds experienced in the adaptive-buffer replay method, in contrast with the fixed one, are probably due to its non-linear behaviour in sampling rehearsal. With that said, if in a real scenario the total storage capacity is fixed, it seems reasonable to apply an adaptive buffer over a fixed one, specially if storage is constrained and pre-known.

Due to its compatibility, more experiments (see Figures 14a and 14b) were conducted with combinations of the simple replay method (with an adaptive buffer) and the regularization approaches. The aim was to determine whether integrating mechanisms such as LwF, EWC, or SI could further boost the retention capabilities of the replay strategy. However, these additional experiments show the same conclusions as before, that the inclusion of regularization did not produce any significant improvement in mitigating forgetting for class-incremental scenarios. Even when paired with replay, the overall performance remained largely consistent with that of the replay-only approaches.



(a) Model average accuracies over the full stream of data

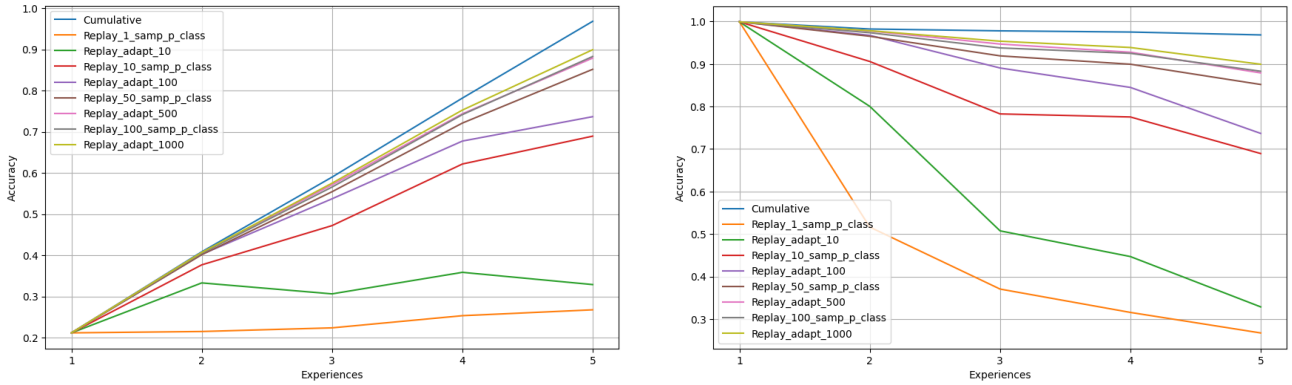
(b) Model average accuracies over trained data

Fig. 14: Comparison of the methods. Simple Replay method with an adaptive buffer of 10 samples compared with a combination of the same approach and regularization strategies (LwF, EWC and SI)

Focusing on replay methods, a comparison between different buffer sizes is shown in Figure 15.

The cumulative method, trained at each experience with all current and past data serves as the upper bound method, showing almost perfect scores (lowest average accuracy obtained at the 5th experience was 0.9672). Logically, the more data is replayed, the higher the accuracy. Nevertheless, the difference between the cumulative

version (with around 12.000 images per experience to train) and the replay methods with buffers of 100 samples per class performs relatively alike, taking into account the difference in storage and computing needed. Another important point to consider is the trajectory of the replay methods, pointing out that, in a real-world scenario where classes keep incrementing, higher buffers would be needed to approximate the performance to a cumulative approach.



(a) Model average accuracies over the full stream of data

(b) Model average accuracies over trained data

Fig. 15: Comparison of the cumulative approach with simple replay methods with different buffer configurations (Replay_x_samp_p_class: Simple replay with strictly x sample per class; Replay_adapt_y: buffer of y samples, adaptive.)

REFERENCES

[1] Hermann Ebbinghaus. *Memory: A contribution to experimental psychology*. Number 3. Teachers college, Columbia university, 1913.

[2] Michael C. Mozer, Harold Pashler, Nicholas Cepeda, Robert Lindsey, and Ed Vul. Predicting the optimal spacing of study: a multiscale context model of memory. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS'09, page 1321–1329, Red Hook, NY, USA, 2009. Curran Associates Inc.

[3] Behzad Tabibian, Utkarsh Upadhyay, Abir De, Ali Zarezade, Bernhard Schoelkopf, and Manuel Gomez-Rodriguez. Optimizing human learning, 2018.

[4] Siddharth Reddy, Igor Labutov, Siddhartha Banerjee, and Thorsten Joachims. Unbounded human learning: Optimal scheduling for spaced repetition. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016.

[5] Wickliffe C. Abraham and Anthony Robins. Memory retention – the synaptic stability versus plasticity dilemma. *Trends in Neurosciences*, 28(2):73–78, 2005.

[6] Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1):25–46, 1995. The Biology and Technology of Intelligent Autonomous Agents.

[7] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[8] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[9] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2021.

[10] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[11] Zhiyuan Chen and Bing Liu. *Lifelong machine learning*. Morgan & Claypool Publishers, 2018.

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017.

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017.

[14] ROBERT HECHT-NIELSEN. Iii.3 - theory of the backpropagation neural network**based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 ieee. In Harry Wechsler, editor, *Neural Networks for Perception*, pages 65–93. Academic Press, 1992.

[15] Héctor J. Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, 5(4):589–593, 1992.

[16] Friedemann Zenke, Ben Poole, and Surya Ganguli. Improved multitask learning through synaptic intelligence. *CoRR*, abs/1703.04200, 2017.

[17] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2022.

[18] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning, 2018.

[19] Arun Mallya and Svetlana Lazebnik. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. *CoRR*, abs/1801.06519, 2018.

[20] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 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.

[21] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[22] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay, 2017.

[23] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017.

[24] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks, 2017.

[25] Dianzhi Yu, Xinni Zhang, Yankai Chen, Aiwei Liu, Yifei Zhang, Philip S. Yu, and Irwin King. Recent advances of multimodal continual learning: A comprehensive survey, 2024.

[26] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *CoRR*, abs/1904.07734, 2019.

[27] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022.

[28] Bernardino Romera-Paredes and Philip H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, 2015.

[29] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning, 2018.

[30] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020.

[31] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[32] William Yang, Ye Zhu, Zhiwei Deng, and Olga Russakovsky. What is dataset distillation learning?, 2024.

[33] Xuyang Zhong and Chen Liu. Towards mitigating architecture overfitting on distilled datasets, 2025.

[34] Shiye Lei and Dacheng Tao. A comprehensive survey of dataset distillation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):17–32, January 2024.

[35] Noveen Sachdeva and Julian McAuley. Data distillation: A survey, 2023.

[36] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent, 2016.
[37] Jiahui Geng, Zongxiong Chen, Yuandou Wang, Herbert Woisetschlaeger, Sonja Schimmler, Ruben Mayer, Zhiming Zhao, and Chunming Rong. A survey on dataset distillation: Approaches, applications and future directions, 2023.
[38] Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, page 1–8. IEEE, July 2021.
[39] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks, 2022.
[40] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression, 2021.
[41] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression, 2022.
[42] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching, 2021.
[43] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories, 2022.
[44] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching, 2022.
[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[46] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *CoRR*, abs/1908.04742, 2019.
[47] Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. Avalanche: A pytorch library for deep continual learning. *Journal of Machine Learning Research*, 24(363):1–6, 2023.
[48] Lutz Prechelt. *Early Stopping - But When?*, pages 55–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.