

Original Article

An Empirical Evaluation of Regularization, Replay, and Architectural Methods Employed in Continual Learning using Split MNIST, CIFAR10, CIFAR100

Neeraj¹, Poonam Nandal²

^{1,2}Department of Computer Science & Engineering, Manav Rachna International Institute of Research and Studies, Faridabad, Haryana, India.

¹Corresponding Author: goelneeru@gmail.com

Received: 25 March 2026

Revised: 24 July 2026

Accepted: 06 August 2026

Published: 30 September 2026

Abstract - Human beings can learn throughout their lives; they learn new concepts, refine their existing skills, employ their experiences in various situations, and build memories for prolonged retention. The ability to learn throughout one's lifetime is known as continual learning, and it is challenging to replicate this capacity in an artificial neural network. Deep models tend to overwrite previously learned information when exposed to sequential tasks, a phenomenon known as catastrophic forgetting. To address catastrophic forgetting, several methods have been proposed. In this study, we focus on a comprehensive empirical evaluation of three major categories of continual learning methods: regularization-based methods, replay-based methods, and architecture-based methods. The experiment uses multiple benchmark settings. For a simple image classification task, use Split-MNIST. For representing more complex tasks, use the Split CIFAR10 and Split CIFAR100 datasets (with and without pretraining). All datasets are used under class-incremental and Task-Incremental Learning scenarios. Each scenario faces distinct challenges. To ensure transparency, each method is evaluated using a shared backbone architecture. Performance of each method in both scenarios is reported using various standard continual learning metrics, such as Average Accuracy (ACC), Average Forgetting and Backward Transfer (BWT), along with measurements of total training time and CPU/GPU memory usage to show the resource efficacy. This paper presents a remarkable difference in terms of levels of difficulty and the relative efficacy of various tactics in both scenarios, task-incremental and Class-Incremental Learning. The performance of Task-Incremental Learning is better than Class-Incremental Learning for all continual learning methods.

Keywords - Continual Learning, Catastrophic Forgetting, Class-Incremental Learning, Task-Incremental Learning.

1. Introduction

Continual Learning (CL) aims to develop a model that sequentially learns new tasks while retaining details from earlier tasks [1, 2]. In recent years, this field of machine learning has been expanding continuously, and it has become a crucial area of research. It addresses the real-world situation where tasks and data are ever-changing, and a model adapts to these changes without losing track of prior knowledge. Traditional learning models assume static data distributions, which is problematic when tasks and data are changing and dynamic in nature. Here, CL comes into play, where a deep neural network model adapts to new tasks dynamically while retaining the previous information. It is also known as incremental learning, since the tasks are given incrementally over time. One of the significant challenges in CL is Catastrophic Forgetting (CF) [3, 4], where a model forgets the previously acquired knowledge while learning a new task. Various CL approaches have been proposed to address the issue of forgetting. These include (a) Replay-based methods,

where a replay buffer is used to improve knowledge consolidation [5, 6] by storing previously trained experiences to rehearse the memory of previously learned tasks with newly arrived tasks. Based on how previous memories are utilized, replay-based methods are divided into 2 groups: (i) Experience replay, where a limited memory buffer is used to store raw training examples [7]. (ii) Generative replay, where synthetic samples for previously trained tasks are generated by training a different generative model [8]. Although replay methods have significant computational and storage costs, they frequently yield good empirical results [9]. (b) Regularization-based methods place restrictions on parameter updates. These methods typically penalize changes to parameters important for previous tasks, as in Elastic Weight Consolidation (EWC) [1], Synaptic Intelligence (SI) [10], and Memory Aware Synapses (MAS) [11]. Another approach regularizes learning by distilling knowledge from previously trained models [12]. Although regularization methods are computationally effective, they often fail when applied to



longer task sequences or extremely complex data. (c) Architecture methods expand the network to allocate capacity for each new task. These methods reduce interference and isolate knowledge, such as PackNet [13], Progressive Neural Networks (PNNs) [14], and Hard Attention to the Task (HAT) [15]. Despite being successful in decreasing forgetting, these methods often compromise scalability and memory consumption due to network expansion or additional parameters.

Although growing research has provided several methods to tackle catastrophic forgetting, comparing these methods is still difficult because of variations in experimental setups, problem formulations, and evaluation metrics. To provide structure to the field, Continual Learning is categorized into three scenarios: task-incremental, domain-incremental, and Class-Incremental Learning. Each scenario introduces specific challenges and degrees of interference. There is also a lack of systematic empirical benchmarks that simultaneously measure accuracy, forgetting, and resource efficiency in this field. Beyond accuracy, careful consideration of resource constraints is essential for the real-world deployment of continual learning systems. The viability of these methods depends on training time, GPU memory, and storage constraints in robotics, embedded devices, and edge computing applications [16, 17]. However, only a small number of studies systematically evaluate continual learning algorithms in terms of both efficiency and performance. This motivates the present study, which aims to give a controlled empirical evaluation/comparison of representative replay, regularization, and architecture-based CL methods, under consistent protocols and evaluation metrics. This work addresses the following research questions such as how do replay-based, regularization-based, and architecture-based continual learning methods compare in terms of Average Accuracy (ACC), average forgetting / Backward Transfer (BWT), training time, and GPU memory usage, how consistent are these trade-offs across datasets of increasing complexity, ranging from Split-MNIST to CIFAR10 and CIFAR100, what is the impact of pretraining (on CIFAR10 before learning CIFAR100) on mitigating forgetting and improving efficiency.

To answer these questions, this paper presents a systematic empirical study with the following contributions:

- An experiment is conducted on two representative benchmarks. For preliminary assessment, Split-MNIST is adapted as a lightweight vision benchmark well used in CL literature. The experiment is then extended to more complex datasets, CIFAR10 and CIFAR100. An experiment on CIFAR100 is conducted both with and without pretraining to evaluate the effects of initialization and dataset complexity on continuous performance.
- Several representative algorithms from three major categories of CL are evaluated, including Elastic Weight Consolidation (EWC) [1], Synaptic Intelligence (SI) [10],

Learning without Forgetting (LwF) [12], Deep Generative Replay (DGR) [8], Incremental Classifier And Representation Learning (iCaRL) [18], Averaged Gradient Episodic Memory (A-GEM) [19], Context-Dependent Gating (XDG) [20], Brain-Inspired Replay (BI-R) [21], Experience Replay [23], and Functional Regularization of the Memorable Past (FROMP) [33].

- Performance is measured using widely adopted metrics such as Average Accuracy (ACC), Average Forgetting and Backward Transfer (BWT) [16, 19], with practical measures of training time, CPU, and GPU memory consumption. This offers practical insights for real-world deployment.
- Beyond accuracy and forgetting, this study measures training time and the memory requirements of each method, highlighting the trade-offs under various continuous learning scenarios.

This paper offers a better understanding of the strengths and limitations of existing continual learning methods by combining resource-aware profiling with thorough evaluation across benchmark datasets. The findings provide practical guidance to researchers and practitioners for selecting appropriate strategies under various performance and resource constraints, promoting the development of scalable lifelong learning in AI. The remainder of this paper is organized as follows: a summary of related work is presented in Section 2. Various CL methods are discussed in Section 3. The experimental setup is given in Section 4. Results are described in Section 5. Section 6 describes the conclusion and future work.

2. Related Work

This section provides an overview of existing research on CL methods to tackle catastrophic forgetting. Regularization methods mitigate forgetting by penalizing changes in the critical weights of the previously learned tasks. Parameter regularization [1, 22, 10, 11] constrains the amount that model parameters can change when learning new tasks, according to their estimated significance to the previous task. The most prominent method, Elastic Weight Consolidation (EWC), was proposed by Kirkpatrick et al. [1]. It leverages the Fisher Information Matrix to measure the importance of each parameter and penalizes modifications to crucial parameters from old tasks that significantly affect previous performance. Online EWC [22] extended the original EWC by maintaining a single, decayed moving average of the Fisher Information Matrix rather than storing separate matrices for each task, thereby making it more scalable to long task sequences. Instead of using explicit Fisher computation, Synaptic Intelligence (SI) [10] estimates the importance of each parameter by tracking its contribution to the loss trajectory during training. Memory Aware Synapses (MAS) [11] further extended this concept by leveraging the sensitivity of the model's output function to each parameter as a significance

signal, making it effective even in unsupervised environments. Function regularization constrains the model's output function by transferring knowledge from the previous model to a new one to preserve information from earlier tasks. Learning without Forgetting (LwF) [12] preserves prior knowledge by distilling outputs from an older large model to a new smaller model, ensuring that current task predictions remain aligned with those of previous tasks. Replay methods [7, 9, 16, 19, 23] store samples from previous tasks as exemplars or generate pseudo-samples through a generative model. These exemplars are replayed when training a new task to preserve stability without reducing plasticity. Experience Replay (ER) [23] interleaves current task data with a memory buffer of previous samples. Despite its effectiveness, ER faces challenges related to memory scalability and privacy.

Variants such as Gradient Episodic Memory (GEM) [16] and A-GEM [19] enhanced stability by constraining gradient updates to avoid conflict with previously learned tasks and to balance the plasticity and stability. Tiny Episodic Memory [7] demonstrated that even very small memories can significantly reduce forgetting, but at the expense of performance degradation when tasks are highly dissimilar. By replaying input samples and their corresponding past logits, Dark Experience Replay (DER) [9] effectively preserves the model output distributions. iCaRL [18] addresses the class-incremental setting by storing class exemplars and combining them with a nearest-mean-of-exemplars classifier. Synthesized pseudo-samples from previous tasks are generated by a generative model (GAN or VAE) [8] and replayed with current data to avoid privacy issues and reduce the need for storing raw data, although this increases training complexity and may result in lower-quality samples over long sequences. Brain-Inspired Replay (BI-R) [21] was proposed to combine replayed data and actual data in a hippocampus-like manner, similar to how the human brain integrates episodic memories. Replay methods provide backward transfer and strong retention, but at the cost of computation,

additional memory storage, or generative quality issues. Architecture methods [13, 14, 15, 20] allocate separate sub-networks or adaptive modules for each task, thereby reducing the probability of forgetting. In Progressive Neural Networks (PNNs) [14], catastrophic forgetting is addressed by adding a new column for each new task while freezing previously learned parameters. Knowledge transfer from old to new tasks is achieved through lateral connections. However, scalability is the main issue in long-term learning scenarios. PackNet [13] introduced an efficient parameter reuse strategy by employing iterative pruning and retraining. Task-relevant neurons are dynamically isolated through attention-based mechanisms; Hard Attention to Task (HAT) [15] and Context-Dependent Gating (XDG) [20] balance memory retention and scalability by activating task-specific subnetworks. Architecture-based methods significantly reduce interference, but their applicability is constrained by the cost of training, size of the model, and the complexity of architectural management [17, 21].

2.1. Evaluation Frameworks

In recent works [1, 17, 21], the standardized evaluation metrics such as Average Accuracy (AA), Backward Transfer (BWT), and Average Forgetting (AF) are required to rate the performance consistency in continual learning models. These metrics show the insights into how well models balance learning new tasks while retaining old knowledge. Many comparative studies still lack standard protocols across datasets, leading to inconsistent evaluations. Moreover, factors like memory use, computational cost, and training efficiency are frequently overlooked, which limits the understanding of resource-performance trade-offs. Thus, a resource-aware and comprehensive evaluation framework is required for fair and scalable benchmarking of continual learning methods.

3. Continual Learning Methods

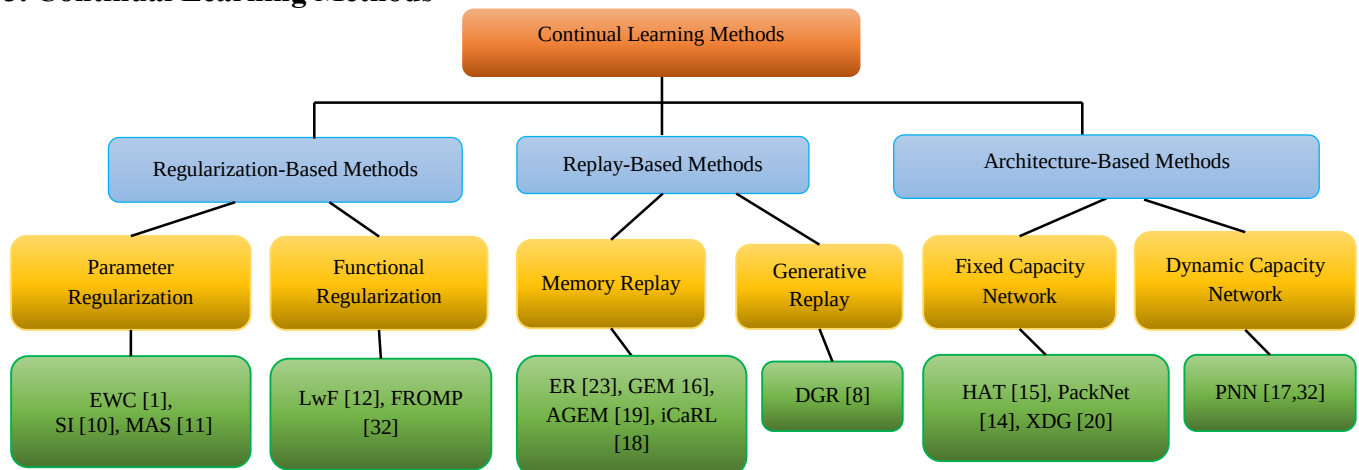


Fig. 1 Continual learning methods

The aim of Continuous Learning (CL) [1] is to enable a neural network to learn from a sequence of tasks without losing the information from previously learned tasks. Continual learning methods are broadly categorized into regularization, replay, and architecture-based methods, as shown in Figure 1. Each method has a different way to alleviate catastrophic forgetting while maintaining efficiency and scalability across tasks.

3.1. Regularization-Based Methods

These methods use penalty terms in the loss function to constrain parameter changes and preserve information from previously learned tasks. They operate in a fixed capacity model and are categorized into knowledge distillation (or function regularization) and parameter regularization methods.

3.1.1. Parameter Regularization

Parameter regularization [1, 10, 11] constrains the amount that model parameters can change when learning new tasks, according to their estimated significance to the previous task.

Elastic Weight Consolidation (EWC) [1], the Importance of previous tasks' parameters is estimated by using the Fisher Information Matrix, and penalizes deviations from them. The loss function is given in Equation (1), and EWC minimizes the L:

$$L_{EWC} = L_{curr}(\theta) + \frac{\lambda}{2} \sum_i F_i(\theta_i - \theta_i^*)^2 \quad (1)$$

Where θ_i current model parameter, θ_i^* parameter learned from the previous task, λ , control regularization strength, and F_i measure the importance of θ_i for the previous task.

Synaptic Intelligence (SI) [10], each parameter's relevance score is computed during training based on the contribution to reducing the previous task loss. The relevance score is given in Equation (2) below:

$$\Omega_i = \sum_t \frac{\Delta \theta_i g_i}{(\Delta \theta_i)^2 + \xi} \quad (2)$$

The importance Ω_i is accumulated, for every parameter θ_i as the product of the parameter update $\Delta \theta_i$, and to ensure stability g_i be the corresponding gradient normalized by the squared update magnitude. ξ The small damping term prevents division by zero.

Memory Aware Synapses (MAS) [11] determines the parameter relevance through the sensitivity of model outputs to parameter changes, and parameter relevance is calculated as given in Equation (3),

$$\Omega_i = E_x \left[\left\| \frac{\partial f(x; \theta)}{\partial \theta_i} \right\| \right] \quad (3)$$

Ω_i The importance of each parameter is computed based on the sensitivity of the model's output $F(x; \theta)$ with respect to changes in the parameter. θ_i . The computed Ω_i values are used to preserve important parameters and mitigate catastrophic forgetting by regularizing future updates.

Variational Continual Learning (VCL) [2] adopts a Bayesian framework, in which the posterior distribution of model parameters after each task serves as the prior for the next. Variational Continual Learning loss function is given in Equation (4),

$$L_{VCL} = E_{q(\theta)}[L_t(\theta)] + D_{KL}[q(\theta)||p(\theta|D_{1:t-1})] \quad (4)$$

$q(\theta)$ is the variational posterior, and $p(\theta)$ is the prior from previous tasks.

3.1.2. Knowledge Distillation/Function Regularization

Function Regularization [12, 33] constrains the model's output function by transferring knowledge from the previous model (teacher model) to a new model (student model) to preserve knowledge of the old task.

Learning without Forgetting (LwF) [12], previous tasks are retained by adding a distillation loss to the objective function, and consistency is maintained in the output before and after learning a new task. Distillation loss is given in Equation (5),

$$L_{distil}(x) = \sum_{y \in oldTasks} P(y|x; \theta^{old}) \log P(y|x; \theta^{new}) \quad (5)$$

$L_{distil}(x)$ is the distillation loss, $P(y|x; \theta^{old})$ probability distribution from the old model and $P(y|x; \theta^{new})$ probability distribution from the new model. Total loss combines the task-specific loss and distillation loss weighted by a factor λ .

Functional Regularization of the Memorable Past (FROMP) [33], a functional regularization method designed for continual learning. It uses anchor points, or stored data from previous tasks called memorable inputs, to maintain the model's behaviour within a Bayesian framework over time.

Regularization methods are memory-efficient, but underperform in large-scale or highly non-stationary environments due to strict parameter constraints.

3.2. Replay-Based Methods

These methods store samples of previous tasks as exemplars or generate pseudo-samples using a generative model. These exemplars are replayed when training a new task to maintain stability without reducing plasticity. The work of [16, 19, 23-27] presents various replay-based methods to mitigate forgetting.

Experience Replay (ER) stores past examples in a small memory buffer and replays them during current task training [23]. The total loss calculation is given by Equation (6),

$$L_{ER} = \alpha L_t(\theta) + (1 - \alpha) L_{replay}(\theta) \quad (6)$$

L_t and L_{replay} are the current task loss and loss over replayed samples, respectively, and α controls their balance.

Gradient Episodic Memory (GEM) [16] maintains a small episodic memory for past tasks. It ensures backward transfer by projecting gradients from the current task so that the loss on stored samples from previous tasks does not increase. It is a constraint-based optimization technique. A-GEM [19] improves upon Gem [16] by projecting the current gradient to prevent an increase in the average loss over stored samples. It reduces computational complexity and maintains performance in long task sequences. Both GEM [16] and A-GEM [19] extend Experience Replay (ER) by constraining gradient updates to prevent loss on past stored samples.

Memory Sample Selection (MSS), samples for memory buffer storage are selected using heuristic methods based on certain rules [24, 25] such as selecting samples near to cluster centroid (e.g. iCaRL [18]), samples with higher diversity [26, 27], or samples near the decision boundary of each class. Replay loss is defined similarly to the RSR loss, as given by Equation (7),

$$L_{replay} = L_{current} + \sum_{i \in \text{selected samples}} L_{past}(x_i, y_i) \quad (7)$$

Deep Generative Replay (GR) [8] avoids storing real data. Instead, a generator (GAN or VAE) produces pseudo-samples that represents the past task. These pseudo-samples are replayed along with the current task. The generative models used include GAN-based [28, 29], Auto Encoder-based [21], Diffusion-based [30], and Model-inversion [31] approaches.

The generative loss is calculated by Equation (8),

$$L_{GR} = L_{current} + \sum_i L_{past}(G(z_i), y_i) \quad (8)$$

$G(z_i)$ is a generative model that produces synthetic data, and y_i corresponding label.

Although these methods require additional memory or generative capacity, they remain effective in maintaining performance.

3.3. Architecture Methods

Architecture-based methods isolate or expand the parts of the network to prevent interference between tasks. These methods are categorized as fixed capacity and dynamic capacity networks.

Progressive Neural Networks (PNN) [32, 33] reduce catastrophic forgetting by expanding the network and adding a new column for newly arrived tasks while freezing the old column. The combined hidden representation for the current task is defined by Equation (9),

$$h_t(x) = [h_{prev}(x), h_{new}(x)] \quad (9)$$

- $h_{prev}(x)$: hidden activations from frozen columns
- $h_{new}(x)$: new task-specific columns

PackNet [13] retained the important weights of earlier tasks by iteratively pruning and retraining the network weights to free up capacity for new tasks. The method maintained the network's capacity fixed through pruning.

Hard Attention to Task (HAT) [15] protected the crucial parameter of previous tasks by using task-specific attention masks. It maintained a fixed network capacity by selectively gating parameters to prevent interference while learning new tasks.

Context-Dependent Gating (XDG) [20] also employed a random gating mechanism to assign disjoint subsets of neurons to each task, thereby preventing interference and mitigating forgetting in fixed-capacity networks.

Architectural methods ensured strong retention, though they increased computational cost and memory requirement.

3.4. Continual Learning Scenarios

It was extremely challenging to evaluate the performance of various CL methods due to wide variety of datasets, variation in the experimental setting, and differing protocols. To tackle this problem, [32] proposed three scenarios for incremental learning, as shown in Figure 2. These scenarios were based on the Task-ID requirement at test time.

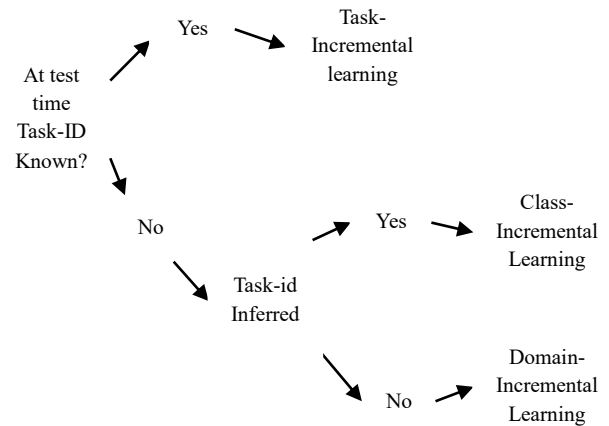


Fig. 2 Decision tree for three scenarios of Continual Learning

3.4.1. Task-Incremental Learning (Task-IL)

In Task-IL [17], Task-ID is provided to the model both during training and testing, which allows the model to select the correct output head for the current task from a multi-headed output layer. This setup simplifies predictions by restricting output to the known tasks.

3.4.2. Class-Incremental Learning (Class-IL)

In Class-IL, the model must classify among all previously learned classes without task identity information while incrementally learning new classes over time. This setting is the most challenging, as it requires maintaining a balance between stability and plasticity simultaneously [17].

3.4.3. Domain-Incremental Learning (Domain-IL) [17]

Task-ID is not available at test time. The input distribution varies constantly, while the task structure remains the same. All tasks are solved by the model using a single output head.

4. Experimental Setup

4.1. Datasets

Three standard benchmark image datasets were used to comprehensively evaluate the CL methods. The Split MNIST dataset contained 10 classes of handwritten digits (0-9); these were divided into 5 tasks with 2 classes per task, as shown in Figure 3. It used the standard 60000/10000 train/test split and 28×28 grayscale images.

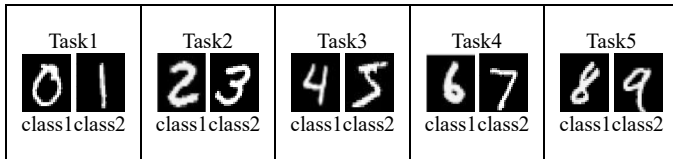


Fig. 3 Split MNIST: continual lifelong learning example

The Split CIFAR10 dataset contained images from 10 classes, divided into 5 tasks, each having 2 classes of images, as shown in Figure 4 and used the standard 50000/10000 train/test split with 32×32 RGB images. The images were normalized using channel-wise mean and standard deviation statistics, and no data augmentation was applied.

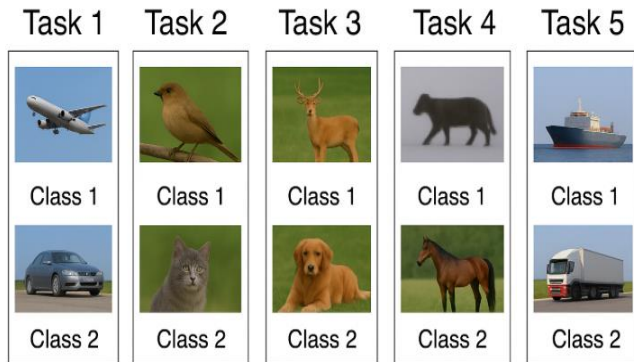


Fig. 4 Split CIFAR10: continual lifelong learning example

The CIFAR100 dataset consisted of images from 100 object classes, divided into 10 tasks with 10 classes per task. The dataset contained 32×32 RGB images normalized by channel-wise mean and standard deviation, and used the standard 50000/10000 train/test split. CIFAR100 (without pretraining) represented a more complex and challenging setting, while CIFAR100 (with CIFAR10 pretraining) was initialized using pretrained weights obtained from CIFAR10 to analyze the benefits of transfer learning in continual learning.

4.2. Continual Learning Scenarios

Two scenarios Task-Incremental Learning and Class-Incremental Learning has been used to evaluate the performance of the CL methods, as shown in Table 1. In both scenarios, the Task-ID was available during training time; however, at test time, the Task-ID was provided only in Task-Incremental Learning, not in Class-Incremental Learning, which made Class-Incremental Learning more challenging.

Table 1. Two scenarios of incremental learning

Scenario	Test time Requirement
Task-Incremental Learning	Task-ID is required
Class-Incremental Learning	Task-ID is not required

4.3. Baseline Configurations

Two baselines were used to compare the performance of various CL methods to reduce forgetting. In Lower Bound (“None”), the model was incrementally trained on all tasks sequentially in the standard way without any specific continual learning strategy. In Upper Bound (“Joint”), the model was trained jointly on all tasks' data at once, representing the best possible performance in the absence of catastrophic forgetting.

4.4. Model Architectures

To make the comparison fair and informative, the same base neural network architecture was employed for all CL methods wherever possible. For the Split MNIST benchmark, the base network consisted of two fully connected hidden layers with 400 ReLU-activated nodes each, followed by a softmax output layer with configuration described in [17]. For Split CIFAR10 and Split CIFAR100, the base network comprised five convolutional layers with 16, 32, 64, 128, and 256 channels, respectively. These were followed by two fully connected layers with 3000 ReLU units each, followed by a softmax classification layer. Each convolutional layer used a 3×3 kernel with a padding of 1, a stride of 1 in the first layer, and a stride of 2 in the remaining layers.

Batch normalization was applied after every convolutional layer, followed by a ReLU activation, and no pooling was used. For the CIFAR100 dataset with CIFAR10 pretraining, the same base network architecture was adopted as in the non-pretrained case; however, the five convolutional layers (with 16-256 channels) were initialised with weights

pretrained on CIFAR10. This pretraining enabled the network to capture generalizable low-level image features, useful for CIFAR100, thereby improving the performance. The convolutional layers were pretrained on CIFAR10 for 150 epochs using the Adam optimizer ($\beta_1=0.9$, $\beta_2=0.999$) with a learning rate of 0.0001 and a mini-batch size of 64. During pretraining, all images were normalized and augmented using horizontal flipping and random cropping.

4.5. Training Protocols

All methods were trained using the same hyperparameters and optimization schemes to ensure fair comparison. The batch size was set to 64, the number of training epochs was 5 for Split MNIST, and 10 for both Split CIFAR10 and CIFAR100. The learning rate for Split MNIST was 0.001 and 0.0001 for both Split CIFAR10 and CIFAR100. The Adam optimizer was used in all experiments. For memory-based methods, a fixed replay buffer size of 100 samples per class was used, and the experimental setting closely followed those in [17], with few modifications. Each experiment was repeated with 5 different random seeds, and the results were reported as mean $\pm$ standard deviation. The total Iterations were calculated as

$$\text{Iterations} = \frac{\text{Dataset size}}{\text{Batch size}} \times \text{Epochs}$$

4.6. Evaluation Metrics

In this study, to quantitatively evaluate both the resource efficiency and performance of different CL methods, various standard metrics, defined in the literature [7, 16], were employed. The metrics used are described as follows.

4.6.1. Average Accuracy (AA)

The mean accuracy across all tasks is measured after the completion of the final training stage. It reflects the model's overall generalization and retention ability.

$$AA = \frac{1}{T} \sum_{t=1}^T a_{T,t}$$

where T represents the total number of tasks and $a_{T,t}$ denotes the test accuracy on the task t after the completion of the final task training.

4.6.2. Average Forgetting (AF)

AF quantifies the average performance degradation on previous tasks after learning new ones.

$$AF = \frac{1}{T-1} \sum_{t=1}^{T-1} \max_{i < t} (a_{i,t} - a_{T,t})$$

where $a_{i,t}$ measure the accuracy on the task t right after the learning task i , $a_{T,t}$ represent the final accuracy on task t after all tasks have been trained.

4.6.3 Backward Transfer (BWT)

BWT evaluates the influence of new tasks learning on the performance of previous tasks.

$$BWT = \frac{1}{T-1} \sum_{t=1}^{T-1} (a_{T,t} - a_{t,t})$$

Where $a_{t,t}$ is the accuracy on task t immediately after learning it, $a_{T,t}$ is the accuracy on the same task after learning all subsequent tasks.

4.6.4. Resource Metrics

These metrics include training time, the total time required to complete the sequential training of all tasks, CPU and GPU memory consumption, which refers to the maximum CPU and GPU memory utilised per method. These metrics provide insights into the computational efficiency and resource utilization of CL methods. In particular, a higher value of Accuracy and BWT metrics represents better performance in continual learning, whereas lower values of the Forgetting measure improve retention capability and overall performance.

5. Results and Discussion

We used several benchmark datasets, such as Split MNIST, Split CIFAR10, and Split CIFAR100, for comparison with various methods of continual learning. For Split MNIST (Table 2), Split CIFAR10 (Table 3), Split CIFAR100 (Table 4), and Split CIFAR100 with model pretraining with CIFAR10 (Table 5), we found a clear distinction between the performance of various CL methods in task-incremental and Class-Incremental Learning scenarios. When "none-lower bound" and "joint-upper bound" baselines results compared with the performance of various CL methods in task-incremental and Class-Incremental Learning, all examined. The performance of both scenarios in regularization methods is such that in EWC, LwF, and FROMP, performance is close to the upper bound, and for SI, performance was near to lower target; when task identity was known during test time (i.e., Task-Incremental Learning), however, in Class-Incremental Learning performance of all these regularization methods dropped, but above to lower target, which shows that these methods failed in this scenario.

Replay methods performed well in these scenarios, but performance degrades from Task-Incremental Learning to Class-Incremental Learning. Replay with exemplars and architecture methods also performed well in class incremental learning and Task-Incremental Learning scenarios, respectively. For Class-Incremental Learning, methods which use a generative model (DGR) or a memory buffer for storing past samples perform well as compared to regularization methods. We can also observe from Table 4 and Table 5, methods that uses a pretrained model have better performance (average accuracy) compared to those methods that have not used a pretrained model. We also find the memory

requirement and computation time of various methods with the same benchmark dataset. All experiments were performed on Google Colab with GPU NVIDIA Tesla T4. There are 100

examples per class to be stored in the memory buffer. For BI-R and DGR, a generative model was used, due to which additional network capacity is consumed.

Table 2. Average test accuracy and Average Forgetting on the split MNIST dataset

Split MNIST		Task-Incremental Learning			Class-Incremental Learning		
Strategy	Method	Avg. Accuracy	Avg. Forgetting	BWT	Avg. Accuracy	Avg. Forgetting	BWT
Baselines	None-Lower Bound	84.38 ± 2.8	6.2 ± 2.5	-5.2	19.83 ± 0.07	98.05 ± 0.0	-98
	Joint-Upper Bound	99.75 ± 0.02	NA	NA	97.94 ± 0.11	NA	NA
Regularization	EWC	99.53 ± 0.00	0.10 ± 0.0	-0.13	19.78 ± 0.11	99.56 ± 0.30	-99.56
	SI	88.99 ± 0.69	10.47 ± 0.29	-10.14	22.59 ± 2.72	75.48 ± 3.33	-75.48
	LwF	99.56 ± 0.19	0.12 ± 0.08	0.1	20.25 ± 0.37	99.36 ± 0.52	-99.36
	FROMP	98.54 ± 1.17	5.3 ± 6.5	-5.3	58.7 ± 0.041	8.8 ± 0.11	-6.8
Replay	DGR	99.45 ± 0.15	9.33 ± 6.96	-9	88.37 ± 1.70	18.29 ± 1.68	-18.29
	ER	99.23 ± 0.32	-15.02 ± 9.87	7.8	90.15 ± 01.3	11.92 ± 1.67	-11.92
	A-GEM	99.05 ± 0.23	-36.84 ± 0.90	-37	65.87 ± 15.8	42.42 ± 19.7	-42.42
	BI-R	99.51 ± 0.15	18.72 ± 2.90	-18.72	85.46 ± 1.65	17.75 ± 2.01	-17.75
Replay with Exemplars	iCaRL	-----	-----	-----	88.99 ± 0.69	10.47 ± 0.29	-10.14
Architecture Based	XDG	99.05 ± 0.03	-----	-----	-----	-----	-----
	Separate Network	99.14 ± 0.10	-----	-----	-----	-----	-----

Continual learning results for Split MNIST (final test accuracy and forgetting as a percentage, averaged over all tasks) on two incremental learning scenarios (Task-Incremental Learning and Class-Incremental Learning). We reported three metrics mentioned as: Average Accuracy (Avg. Accuracy (higher is better)), Average Forgetting (Avg. Forgetting (lower is better)), and backward transfer (BWT (higher is better)).

The information about task identity is available during training time in all scenarios in the experiment of continual learning. The experiment was performed on each method five times with different random seeds, and the reported value is the mean ($\pm$ s.e.m.) over these runs. methods perform well in task-incremental, whereas with Class-Incremental Learning, performance decreased drastically.

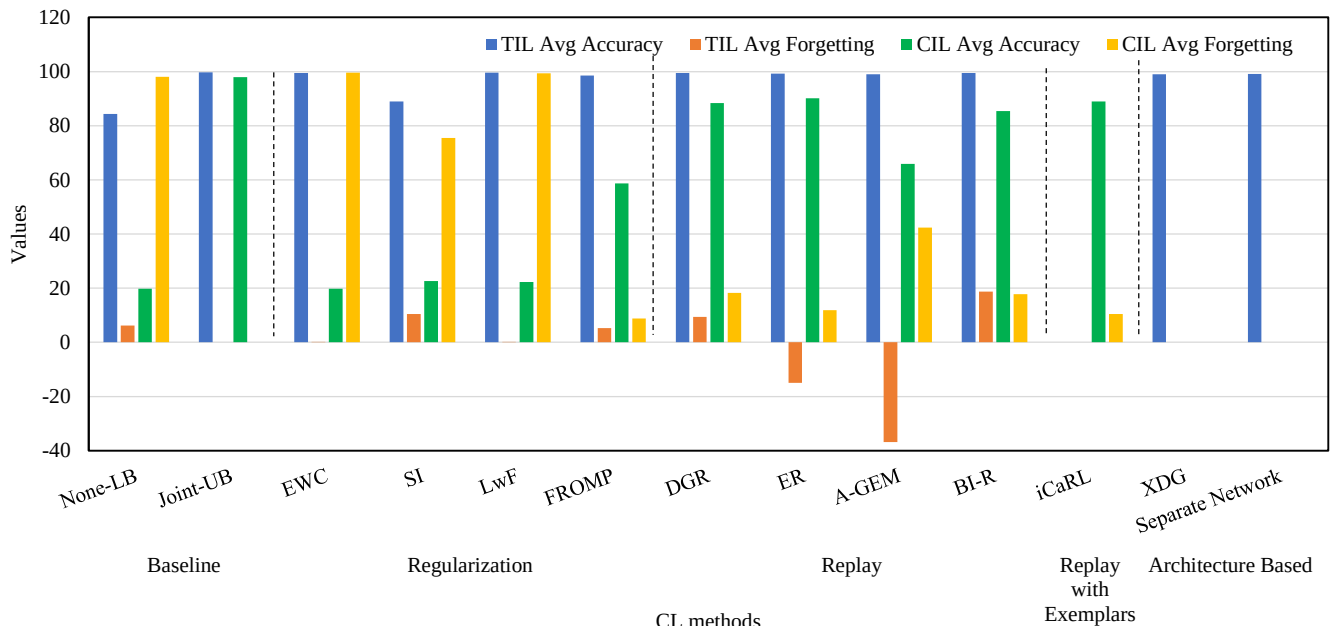


Fig. 5 Average test accuracy and forgetting on the split MNIST under Task-Incremental and Class-Incremental Learning setting

Table 3. Average test accuracy and Average forgetting on Split CIFAR10 dataset

Split CIFAR10		Task Incremental Learning			Class Incremental Learning		
Strategy	Method	Avg. Accuracy	Avg. Forgetting	BWT	Avg. Accuracy	Avg. Forgetting	BWT
Baselines	None-Lower Bound	76.28 ± 2.48	57.33 ± 7.29	-57.3	18.20 ± 0.10	97.49 ± 0.77	-97.49
	Joint-Upper Bound	87.82 ± 2.03	NA	NA	56.30 ± 0.50	NA	NA
Regularization	EWC	82.20 ± 1.20	53.50 ± 6.32	-53.5	17.40 ± 0.05	74.07 ± 0.87	-92.93
	SI	77.75 ± 3.38	53.37 ± 8.03	-53.37	18.32 ± 0.21	97.15 ± 0.72	-97.15
	LwF	87.73 ± 1.26	38.28 ± 6.55	-42.41	18.16 ± 0.17	97.72 ± 0.42	-98
Replay	DGR	69.28 ± 2.87	11.31 ± 2.27	2.34	17.60 ± 0.43	72.59 ± 1.72	-90.73
	ER	86.27 ± 1.01	18.26 ± 3.17	-18.26	26.26 ± 1.07	67.95 ± 1.55	-83.14
	A-GEM	85.34 ± 1.13	4.07 ± 1.65	23.44	18.26 ± 0.16	96.67 ± 1.12	-96.92
	BI-R	85.45 ± 1.10	20.72 ± 2.90	-20.72	55.08 ± 0.57	45.11 ± 0.95	-45.11
Replay with Exemplars	iCaRL	-----	-----	-----	42.97 ± 2.08	46.63 ± 1.13	-47
Architecture Based	XDG	82.72 ± 1.82	15.45 ± 2.09	-17.63	-----	-----	-----

Continual learning results for Split CIFAR10 (final test accuracy, and forgetting as a percentage, averaged of all tasks) on two incremental learning scenarios (Task-Incremental Learning and Class-Incremental Learning). We reported three metrics mentioned as: Average Accuracy (Avg. Accuracy (higher is better)), Average Forgetting (Avg. Forgetting (lower

is better)), and BWT (backward transfer (higher is better)). The information about task identity is available during training time in all scenarios in the experiment of continual learning. The experiment was performed on each method five times with different random seeds, and the reported is the mean ($\pm$ s.e.m.) over these runs.

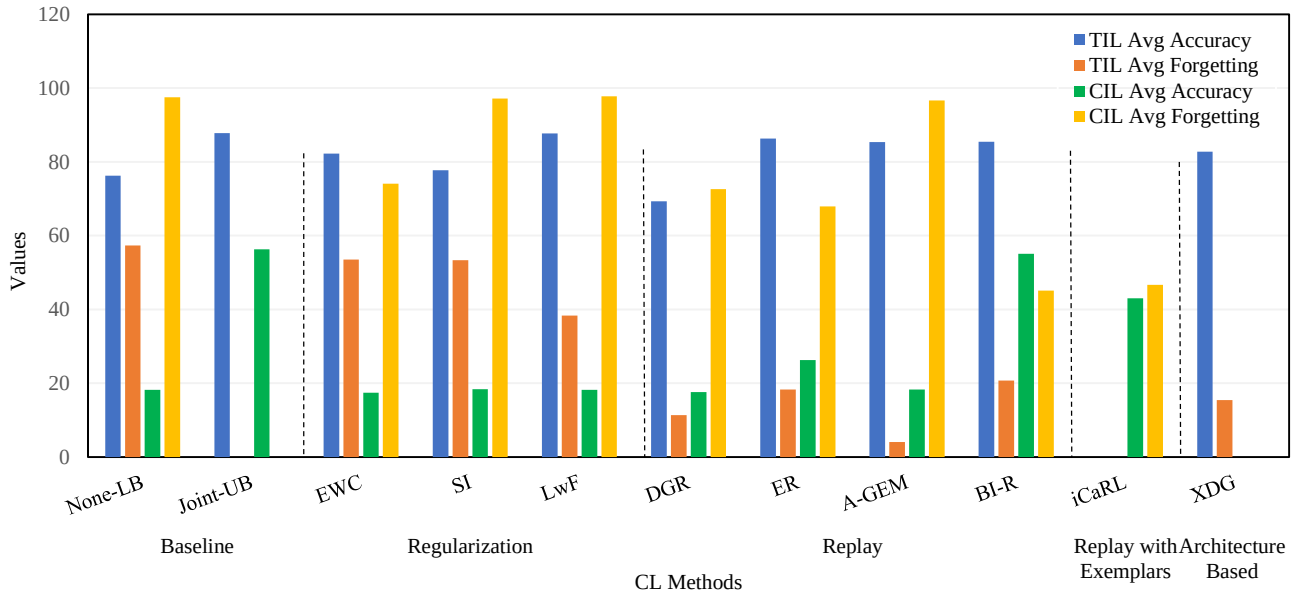


Fig. 6 Average test accuracy and forgetting on the split CIFAR10 under Task-Incremental and Class-Incremental Learning setting

Table 4. Average test accuracy and Average Forgetting on the split CIFAR100 dataset

Split CIFAR100		Task-Incremental Learning			Class-Incremental Learning		
Strategy	Method	Avg. Accuracy	Avg. Forgetting	BWT	Avg. Accuracy	Avg. Forgetting	BWT
Baselines	None-Lower Bound	52.51 ± 0.63	41.02 ± 3.33	-41.02	6.59 ± 0.31	79.15 ± 5.90	-79.22
	Joint-Upper Bound	68.97 ± 0.60	NA	NA	38.42 ± 0.13	NA	NA

Regularization	EWC	51.43 ± 0.96	2.72 ± 0.32	-2.72	$6.70(\pm 0.19)$	69.57 ± 0.85	-69.57
	SI	57.30 ± 0.50	14.50 ± 1.39	-14.50	6.63 ± 0.24	77.3 ± 0.94	-75
	LwF	63.13 ± 1.17	42.32 ± 0.08	42.29	7.21 ± 0.41	83.85 ± 0.35	-83.85
Replay	DGR	26.81 ± 1.21	21.93 ± 0.65	-21.93	5.35 ± 0.52	79.0 ± 0.1	-79.0
	ER	58.04 ± 0.66	3.17 ± 0.27	-3.17	19.98 ± 0.70	46.97 ± 1.71	-46.97
	A-GEM	60.22 ± 0.18	5.28 ± 0.91	-4.74	6.61 ± 0.47	79.27 ± 4.79	-79.27
	BI-R	69.58 ± 0.63	2.33 ± 0.25	-2.33	9.95 ± 0.47	81.46 ± 1.96	-81.46
Replay with Exemplars	iCaRL	-----	-----	-----	20.22 ± 0.09	27.16 ± 0.63	-27.16
Architecture Based	XDG	52.39 ± 0.54	67.86 ± 0.64	-67.86	-----	-----	-----

Continual learning results for Split CIFAR100 (final test accuracy, and forgetting as a percentage, averaged all tasks) on two incremental learning scenarios (Task-Incremental Learning and Class-Incremental Learning). We reported three metrics mentioned as: Average Accuracy (Avg. Accuracy (higher is better)), Average Forgetting (Avg. Forgetting (lower

is better)), and BWT (backward transfer (higher is better)). The information about task identity is available during training time in all scenarios in the experiment or continual learning. The experiment was performed on each method five times with different random seeds, and the reported value is the mean ($\pm$ s.e.m.) over these runs.

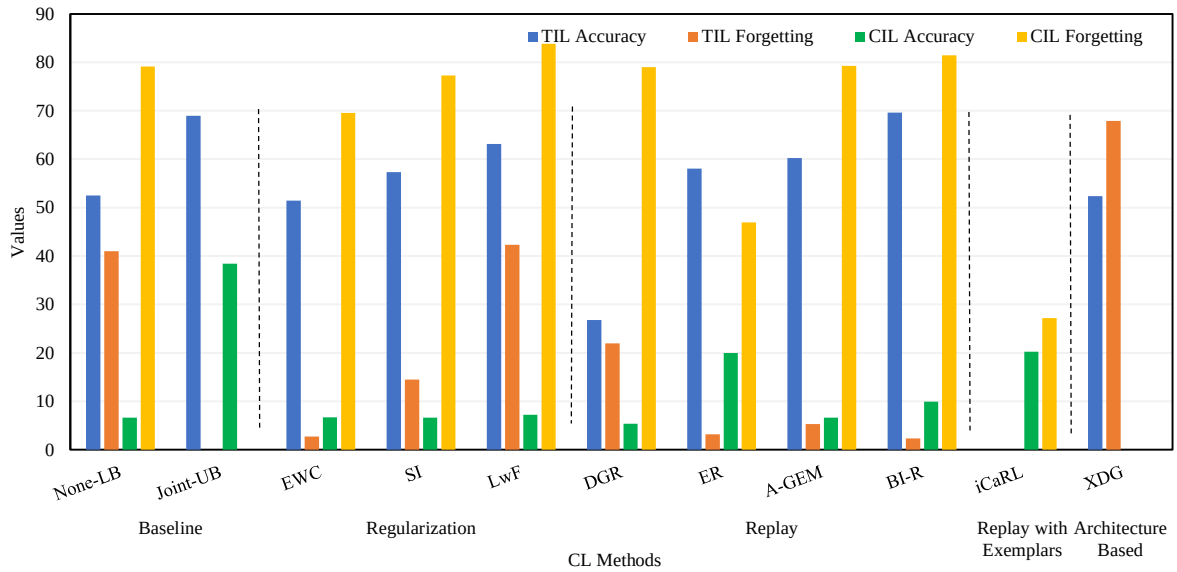


Fig. 7 Average test accuracy and forgetting on split CIFAR100 CIFAR10 under Task-Incremental and Class-Incremental Learning setting

Table 5. Average test accuracy and Average Forgetting on the CIFAR100(pre-trained) dataset

Split CIFAR100 Pretrained with CIFAR10		Task-Incremental Learning			Class-Incremental Learning		
Strategy	Method	Avg. Accuracy	Avg. Forgetting	BWT	Avg. Accuracy	Avg. Forgetting	BWT
Baselines	None-Lower Bound	64.50 ± 0.24	14.25 ± 8.23	-14.25	7.78 ± 0.17	100.00	-97.0
	Joint-Upper Bound	71.52 ± 0.29	NA	NA	40.29 ± 0.10	NA	NA
Regularization	EWC	66.72 ± 0.17	33.85 ± 0.15	-45.1	7.92 ± 0.09	95.19 ± 1.12	-95.19
	SI	68.32 ± 0.13	3.43 ± 0.79	36.3	7.77 ± 0.07	97.99 ± 1.60	-97.99
	LwF	71.50 ± 1.08	16.7 ± 0.71	-16.5	11.79 ± 0.59	94.09 ± 0.64	-94.09
Replay	DGR	47.91 ± 0.00	32.85 ± 1.77	-32.85	7.80 ± 0.21	95.09 ± 0.27	-95.09
	ER	69.88 ± 0.28	3.95 ± 1.61	40.76	29.85 ± 0.29	50.16 ± 3.74	-50.16
	A-GEM	67.49 ± 0.48	20.68 ± 0.77	-20.68	8.22 ± 0.02	97.70 ± 0.91	-97.70
	BI-R	70.42 ± 0.27	23.83 ± 4.23	-23.83	10.74 ± 0.39	99.51 ± 0.25	-99.51

Replay with Exemplars	iCaRL	-----	-----	-----	27.24 ± 0.59	43.05 ± 2.92	-43.05
Architecture Based	XDG	67.84 ± 0.31	0.495	3.67	-----	-----	-----

Continual learning results for Split CIFAR100 (final test accuracy, and forgetting as a percentage, averaged of all tasks) on two incremental learning scenarios (Task-Incremental Learning and Class-Incremental Learning). We reported three metrics mentioned as: Average Accuracy (Avg. Accuracy (higher is better)), Average Forgetting (Avg. Forgetting (lower is better)), and BWT (backward transfer (higher is better)).

The information about task identity is available during training time in all scenarios in the experiment or continual learning. The experiment was performed on each method five times with different random seeds, and the reported value is the mean ($\pm$ s.e.m.) over these runs. The convolutional layers that were used by all the methods compared in the above Table 5 were pre-trained on CIFAR10.

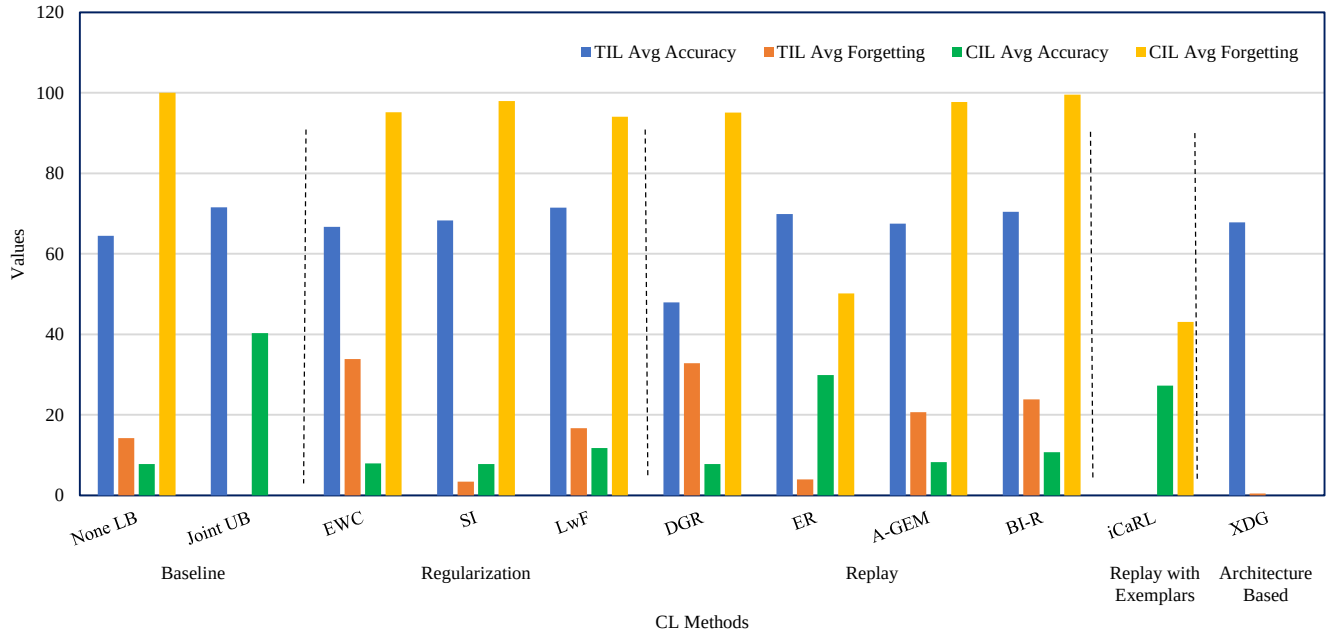


Fig. 8 Average test accuracy and forgetting on split CIFAR100 (pretrained with CIFAR10) under Task-Incremental and Class-Incremental Learning setting

Table 6. Computation Time (TT), CPU, and GPU memory requirements for the split CIFAR100 dataset in two scenarios

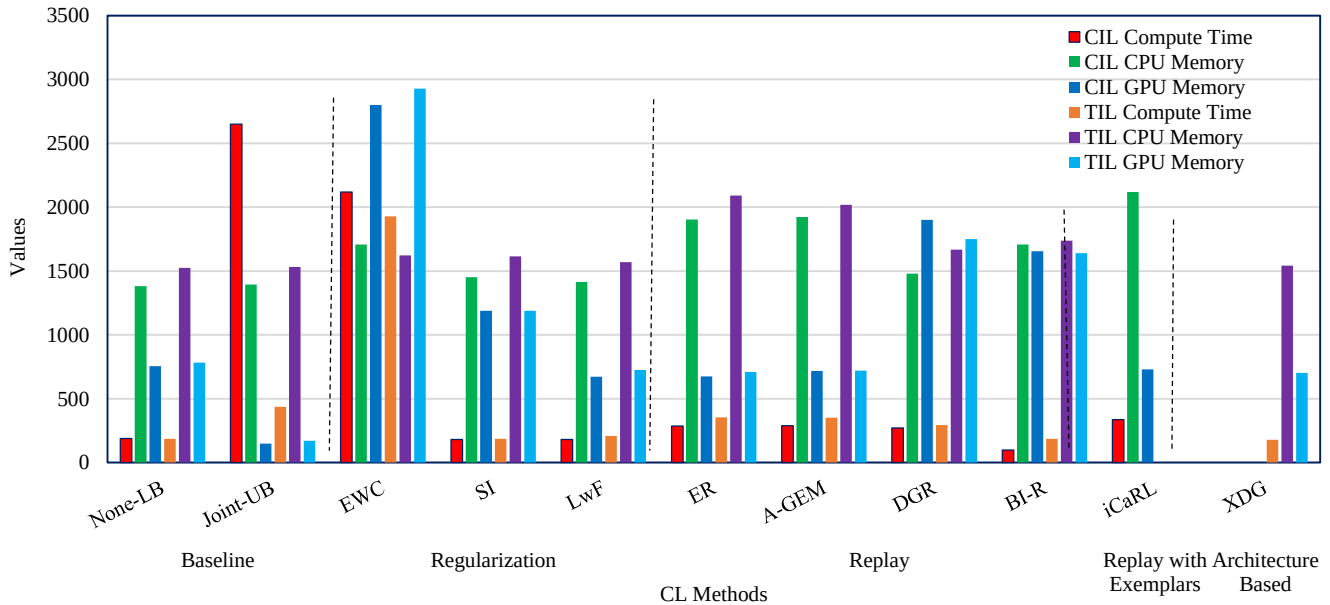
Split CIFAR100		Class-Incremental Learning			Task-Incremental Learning		
Strategy	Method	Compute Time (TT)	CPU Memory	GPU Memory	Compute Time (TT)	CPU Memory	GPU Memory
Baselines	None-Lower Bound	189.42	1382.44	755.61	186.37	1526.15	781.91
	Joint-Upper Bound	2651.54	1394.71	149.40	437.73	1533.73	170.57
Regularization	EWC	2118.78	1707.97	2801	1927.43	1621.97	2929.49
	SI	181.49	1451.65	1187.93	186.08	1616.16	1189.86
	LwF	182.39	1414.26	672.78	209.08	1569	725.64
Replay	DGR	287.38	1902.76	675.02	355.45	2090.79	710.33
	ER	290.24	1923.10	718.45	353.04	2018.87	720.42
	A-GEM	272.7	1480.8	1901.2	294.81	1667.26	1751.73
	BI-R	97.86	1707.46	1656.14	185.72	1738.23	1640.58
Replay with Exemplars	iCaRL	335.78	2117.86	731.20	X	X	X
Architecture Based	XDG	X	X	X	178.28	1542.71	702.71

Table 7. Computation Time (TT), CPU, and GPU memory requirements for split CIFAR100 with model pretraining with CIFAR10 in two scenarios

Pretrained (Split CIFAR100)		Class Incremental Learning			Task Incremental Learning		
Strategy	Method	Compute Time (TT)	CPU Memory	GPU Memory	Compute Time (TT)	CPU Memory	GPU Memory
Baseline	None-lower bound	75.80	1397.15	803.34	50.27	1552.36	656.76
	Joint-Upper bound	64.80	1424.61	141.01	161.90	1555.52	149.04
Regularization	EWC	983.67	1389.26	3023.15	973.33	1636.12	3152.20
	SI	80.07	1457.53	1390.74	81	1618	1333
	LwF	73.31	1432.07	816.24	96.79	1604.46	816.24
Replay	ER	80.65	1607.30	816.46	123.26	1761.46	816.60
	A-GEM	90.79	1619.99	880.84	135.17	1780.58	880.09
	DGR	408.98	1535.41	1636.67	428.43	1680.47	1636.70
	BI-R	166.14	1715.6	1656.14	348.25	1726.57	1640.58
Replay with Exemplars	iCaRL	358.98	1997.41	741.11	X	X	X
Architecture Based	XDG	X	X	X	65.59	1552.05	803.32

In the above tables, comparisons of various state-of-the-art methods on different metrics, their computation time and memory requirements on the benchmark dataset are represented.

The performance of replay-based methods has also been affected by the size of the replay buffer used to store samples from the previously trained tasks that are replayed with the current task training.

**Fig. 9 Compute time, CPU memory, and GPU memory required by various CL methods for Split CIFAR100 dataset (without pretraining) in Class-Incremental and Task-Incremental Learning setting**

Average test accuracy and average forgetting in both scenarios, Task-Incremental Learning and Class-Incremental Learning, on Split MNIST, Split CIFAR10, Split CIFAR100, and Split CIFAR100 with pretraining are represented in Figure 5, Figure 6, Figure 7, and Figure 8, respectively. In Figure 5-8, the x-axis represents CL methods, and the y-axis represents

average accuracy and average forgetting. Compute time, CPU memory, and GPU memory requirements of Split CIFAR100 with and without pretraining are represented in Figures 9 and 10, respectively, for various CL methods. Figures 9 and 10 shows the computation time and memory requirements of various CL methods in both scenarios-Task-Incremental

Learning and Class-Incremental Learning, where the x-axis represents the CL methods and the y-axis represents the computation time in seconds and memory requirements in MB's. Table 1 to Table 7 represents the average accuracy, average forgetting, backward transfer, computation time, and memory requirement of various CL methods on several

benchmark datasets. These tables help us to choose a particular method when resource constraints are present. We can improve these by combining several CL methods or generating hybrid methods. The above tables of Split CIFAR100, with and without pretraining, also show that pretraining improves the computation time.

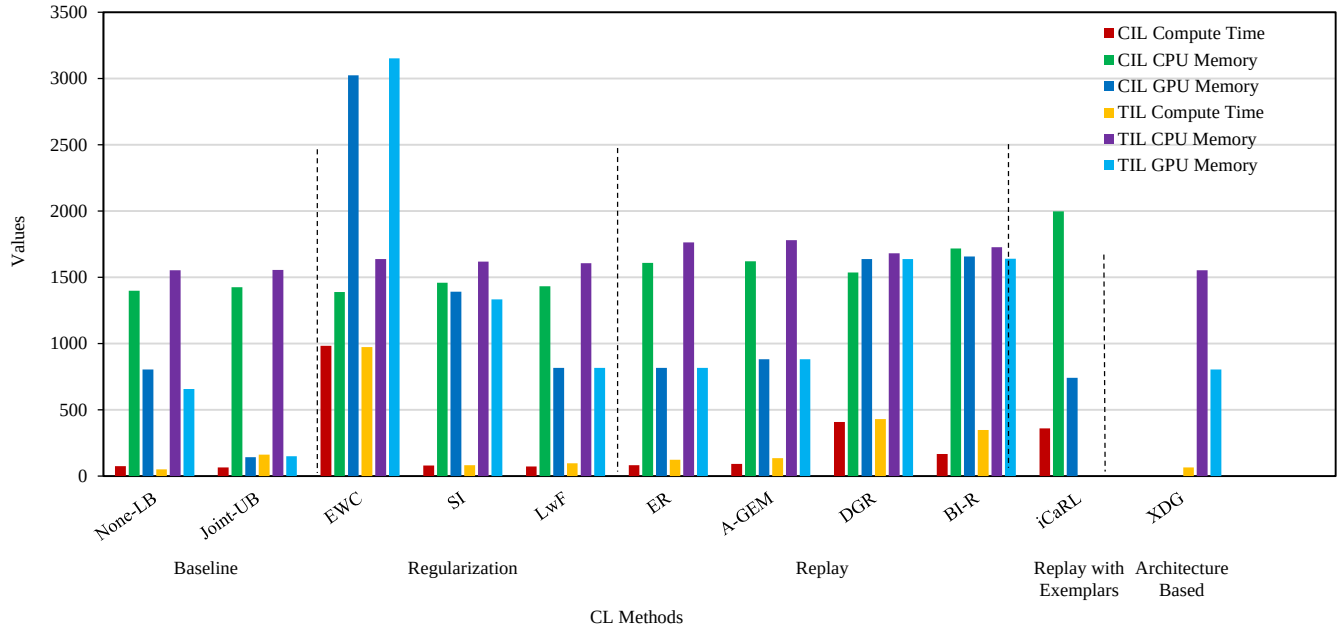


Fig. 10 Compute time, CPU memory, and GPU memory required by various CL methods for the Split CIFAR100 dataset (pretrained with CIFAR10) in Class-Incremental and Task-Incremental Learning setting

6. Analysis and Discussion

The comparative analysis highlights how different methods behave differently across the datasets. All methods perform well under Task-Incremental Learning as compared to Class-Incremental Learning; as Class-Incremental Learning needs a single output space, enhances catastrophic interference, while Task-Incremental Learning allows task-specific heads. EWC and LwF are static in nature, constraining their ability to adapt to new data distributions, especially in complicated datasets like Split CIFAR100, significant forgetting occurs in Class-Incremental Learning.

Replay methods, which reuse exemplars or synthetic data, effectively mitigate forgetting. ER and BI-R perform better than others due to continual rehearsal of previous samples, balancing stability and plasticity. These methods require additional memory, and for large-scale datasets, need more computation. Strong task isolation is provided in XDG and Separate Network architectures that effectively reduce forgetting; however, these methods are less efficient and scalable while handling a large number of tasks. It is also shown that feature reuse across related domains significantly minimizes catastrophic forgetting, as in the pretrained Split CIFAR100. Representational generality enhanced by pretraining, which improves average accuracy by ~10-15% for

most methods. The computation cost is high for methods that have strong retention. Replay methods balance compute efficiency and retention.

7. Scope of future work

The present study provides several promising research directions. Stability and efficiency can be further improved via hybrid CL frameworks that combine adaptive regularization and replay. Another important direction is the development of memory-efficient, dynamic architectures that are capable of compressing or expanding in response to task complexity. Furthermore, the scalability and generalization of CL systems will be improved by advancing meta-CL techniques for automated parameter tuning and expanding evaluations to large-scale, cross-domain benchmarks.

8. Conclusion

This comparative empirical analysis of various CL methods showed that methods such as EWC, LwF, BI-R effectively handle Task-Incremental Learning with minimal forgetting, whereas Class-Incremental Learning poses challenges and there is significant performance degradation. This analysis also demonstrates that no single continual learning method performs well across diverse datasets and environments.

Regularization methods (EWC, LwF) perform well for simple domains but face challenges with complicated, high-dimensional data. Replay methods provide the best accuracy-forgetting balance for vision datasets, but resource requirements are more. Moreover, pretraining reduces catastrophic forgetting and significantly improves representation transfer. Architectural methods are impractical for long task sequences and resource-intensive, but ensure isolation.

For real-world implementation, selecting a continual learning method requires a balance between computational efficiency and accuracy.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

- [1] James Kirkpatrick et al., "Overcoming Catastrophic Forgetting in Neural Networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521-3526, 2017. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [2] Cuong V. Nguyen et al., "Variational Continual Learning," *arxiv*, pp. 1-18, 2018. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [3] Michael McCloskey, and Neal J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109-165, 1989. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [4] Ian J. Goodfellow et al., "An Empirical Investigation of Catastrophic Forgetting in Gradient-based Neural Networks," *arXiv*, pp. 1-9, 2013. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [5] Delphine Oudiette, and Ken A. Paller, "Upgrading the Sleeping Brain with Targeted Memory Reactivation," *Trends in Cognitive Sciences*, vol. 17, no. 3, pp. 142-149, 2013. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [6] Gido M. van de Ven et al., "Hippocampal Offline Reactivation Consolidates Recently Formed Cell Assembly Patterns during Sharp Wave-Ripples," *Neuron*, vol. 92, no. 5, pp. 968-974, 2016. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [7] Arslan Chaudhry et al., "On Tiny Episodic Memories in Continual Learning," *arXiv*, pp. 1-15, 2019. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [8] Hanul Shin, et al., "Continual Learning with Deep Generative Replay," *Advances in Neural Information Processing Systems*, vol. 30, 2017. [[Google Scholar](#)] [[Publisher Link](#)]
- [9] Pietro Buzzega et al., "Dark Experience for General Continual Learning: A Strong, Simple Baseline," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15920-15930, 2020. [[Google Scholar](#)] [[Publisher Link](#)]
- [10] Friedemann Zenke, Ben Poole, and Surya Ganguli, "Continual Learning through Synaptic Intelligence," *Proceedings of the 34th International Conference on Machine Learning (PMLR)*, vol. 70, pp. 3987-3995, 2017. [[Google Scholar](#)] [[Publisher Link](#)]
- [11] Rahaf Aljundi et al., "Memory Aware Synapses: Learning What (not) to Forget," *Computer Vision (ECCV): 15th European Conference, Munich, Germany*, vol. 11207, pp. 1-11, 2018. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [12] Zhizhong Li, and Derek Hoiem, "Learning without Forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935-2947, 2017. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [13] Arun Mallya, and Svetlana Lazebnik, "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 7765-7773, 2018. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [14] Andrei A. Rusu et al., "Progressive Neural Networks," *arXiv*, pp. 1-16, 2016. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [15] Joan Serra et al., "Overcoming Catastrophic Forgetting with Hard Attention to the Task," *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 4548-4557, 2018. [[Google Scholar](#)] [[Publisher Link](#)]
- [16] David Lopez-Paz, and Marc'Aurelio Ranzato, "Gradient Episodic Memory for Continual Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017. [[Google Scholar](#)] [[Publisher Link](#)]
- [17] Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias, "Three Types of Incremental Learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185-1197, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [18] Sylvestre-Alvise Rebuffi et al., "iCaRL: Incremental Classifier and Representation Learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 5533-5542, 2017. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [19] Arslan Chaudhry et al., "Efficient Lifelong Learning with A-GEM," *arxiv*, pp. 1-20, 2019. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [20] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman, "Alleviating Catastrophic Forgetting using Context-Dependent Gating and Synaptic Stabilization," *Proceedings of the National Academy of Sciences*, vol. 115, no. 4, pp. E10467-E10475, 2018. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [21] Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias, "Brain-Inspired Replay for Continual Learning with Artificial Neural Networks," *Nature Communications*, vol. 11, no. 1, pp. 1-14, 2020. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [22] Jonathan Schwarz et al., "Progress and Compress: A Scalable Framework for Continual Learning," *Proceedings of the 35th International Conference on Machine Learning*, pp. 4528-4537, 2018. [[Google Scholar](#)] [[Publisher Link](#)]

- [23] David Rolnick et al., “Experience Replay for Continual Learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019. [[Google Scholar](#)] [[Publisher Link](#)]
- [24] Arslan Chaudhry et al., “Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence,” *European Conference on Computer Vision*, Springer, Cham, vol. 11215, pp. 556-572, 2018. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [25] Rahaf Aljundi, et al., “Online Continual Learning with Maximal Interfered Retrieval,” *Advances in Neural Information Processing Systems*, vol. 32, 2019. [[Google Scholar](#)] [[Publisher Link](#)]
- [26] Rahaf Aljundi et al., “Gradient-based Sample Selection for Online Continual Learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019. [[Google Scholar](#)] [[Publisher Link](#)]
- [27] Jihwan Bang et al., “Rainbow Memory: Continual Learning with a Memory of Diverse Samples,” *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, pp. 8214-8223, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [28] Chenshen Wu et al., “Memory Replay GANs: Learning to Generate Images from New Categories without Forgetting,” *Advances in Neural Information Processing Systems*, vol. 31, 2018. [[Google Scholar](#)] [[Publisher Link](#)]
- [29] Ye Xiang et al., “Incremental Learning using Conditional Adversarial Networks,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), pp. 6618-6627, 2019. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [30] Quentin Jodelet et al., “Class-Incremental Learning using Diffusion Model for Distillation and Replay,” *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Paris, France, pp. 3417-3425, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [31] James Smith et al., “Always be Dreaming: A New Approach for Data-Free Class Incremental Learning,” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, pp. 9354-9364, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [32] Gido M. van de Ven, and Andreas S. Tolias, “Three Scenarios for Continual Learning,” *arXiv*, pp. 1-18, 2019. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [33] Pingbo Pan et al., “Continual Deep Learning by Functional Regularisation of Memorable Past,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 4453-4464, 2020. [[Google Scholar](#)] [[Publisher Link](#)]