Continual Learning with Experience Replay

Andrii Krutsylo

PhD Dissertation

Supervisor:
Ph.D., D.Sc., Eng. Paweł Morawiecki

Institute of Computer Science
Polish Academy of Sciences

September 29, 2025

Acknowledgements

First and foremost, I am deeply thankful to my advisor, Paweł Morawiecki. His trust in my ideas and the freedom he granted me to decide and research by myself allowed me to develop my own interests, views and approaches. He was always there when I needed guidance, and I truly appreciate his support.

A special thank you goes to Jacek Tabor and Stanisław Jastrzębski. They were my Master's thesis advisors and introduced me to research, patiently explaining the basics of machine learning. Their early support and advice were crucial in launching my academic career.

I also owe a great deal to Oliver Kramer, whom I met at ESANN 2022 during a casual conference conversation. Oliver not only provided me with access to the GPUs in his lab, making it possible to run most of the experiments in this dissertation, but also treated me like a real scientist from the very beginning. His encouragement has had a lasting impact on how I see myself and how I hope to guide the next generation of researchers.

I am deeply grateful to Vincenzo Lomonaco for inviting me to Pisa for a three-month research visit. During that period, I wrote the central paper of this dissertation. The experience of working on an EU-funded project with his support gave me a huge boost and allowed me to secure many more grants and positions.

I could not have done this without the constant support of my parents, Tetiana and Oleksandr. They backed my decisions, even when they were difficult to understand, and encouraged me every step of the way.

Finally, my profound and eternal gratitude goes to the Ukrainian army. Their efforts in defending our home have given me the opportunity to work in safety and pursue my passion for research. I will never surrender.

The "Diverse Memory for Experience Replay in Continual Learning" was supported by the Polish National Science Center (NCN) under Grant 2018/31/B/ST6/03003.

The "Batch Sampling for Experience Replay" project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952215.

Contents

List of Publications

Chapters 2–6 include text and figures adapted from the following peer-reviewed articles, reproduced in accordance with publisher policies:

1. **A. Krutsylo**, P. Morawiecki. Diverse Memory for Experience Replay in Continual Learning. In *ESANN*, 2022. (70 points)
2. P. Morawiecki, **A. Krutsylo**, M. Wołczyk, M. Śmieja. Hebbian Continual Representation Learning. In *HICSS*, 2023. (140 points)
3. **A. Krutsylo**. Batch Sampling for Experience Replay. In *CODS-COMAD*, 2024. (Best Student Paper Award)
4. **A. Krutsylo**. Evaluating Knowledge Retention in Continual Learning. In *SAC*, 2024. (20 points)
5. **A. Krutsylo**. Merging versus Separating Replay Samples in Continual Learning. In *ICANN*, 2025. DOI: https://doi.org/10.1007/978-3-032-04558-4_48 (20 points)
6. C. Rebillard, J. Hurtado, **A. Krutsylo**, L. Passaro, V. Lomonaco. Continually Learn to Map Visual Concepts to Language Models in Resource-Constrained Environments. In *Neurocomputing*, 652 (2025) 131013. DOI: https://doi.org/10.1016/j.neucom.2025.131013 (140 points)

The following coauthored works were referenced in the dissertation:

1. M. Wołczyk, **A. Krutsylo**. Remember More by Recalling Less (Student Abstract). In *AAAI Doctoral Symposium*, 2021.
2. **A. Krutsylo**. The Inter-Batch Diversity of Samples in Experience Replay for Continual Learning (Student Abstract). In *AAAI Doctoral Symposium*, 2024.
3. **A. Krutsylo**. The Hebbian Forward-Forward Algorithm. In *Workshop on Optimization for Machine Learning at NeurIPS*, 2025.
4. **A. Krutsylo**. Non-Uniform Memory Sampling in Experience Replay. Preprint, 2025.
5. **A. Krutsylo**. Scalable Forward-Forward Algorithm. Preprint, 2025.

The point values of the publications are taken from the "List of scientific journals and reviewed materials from international conferences", announced by the Polish Ministry of Science and Higher Education in the communication of January 5, 2024.

Streszczenie

Uczenie ciągłe w kontekście uczenia maszynowego to umiejętność nabywania nowych zadań w czasie bez utraty wcześniej zdobytych umiejętności. Główną trudnością jest katastrofalne zapominanie, w którym nowe dane treningowe nadpisują wcześniejszą uzyskaną wiedzę. Niniejsza rozprawa bada w jaki sposób metody oparte na odtwarzaniu (replay) konstruują, wykorzystują i oceniają pamięć.

Po pierwsze, badam konstruowanie pamięci. Wprowadzam Deep Features Buffer, który poprawia różnorodność przechowywanych przykładów, porównując reprezentacje cech zamiast polegać na losowym lub rezerwuarowym próbkowaniu. Proponuję także hebbowskie ciągłe uczenie reprezentacji, w którym aktualizacje wag synaptycznych działają jako bufor pośredni, pozwalając modelom zachować reprezentacje bez jawnego magazynowania. Dodatkowo wnoszę wkład w Continual Visual Mapping, które zastępuje dynamiczny bufor pamięci wstępnie wytrenowanym, zamrożonym dużym modelem językowym, dostarczającym osadzenia językowe jako stabilne zakotwiczenia dla każdej nowej klasy. Te podejścia ilustrują komplementarne rozwiązania problemu budowy pamięci.

Po drugie, analizuję wykorzystanie odtwarzania. Pokazuję, że protokół aktualizacji — czy odtwarzane dane są łączone z nowymi danymi, czy od nich oddzielane — zasadniczo zmienia dynamikę optymalizacji i może nawet odwrócić skuteczność poszczególnych heurystyk selekcji. Aby rozwiązać ograniczenia istniejących metod, opracowuję nową rodzinę strategii próbkowania na poziomie partii danych (ang. batch), które oceniają partie odtwarzania przy użyciu Batch Cosine Distance, zapewniając ich zbiorową informacyjność i brak redundancji.

Po trzecie, ponownie rozważam ewaluację. Standardowe benchmarki często opierają się na dokładności, co utrudnia odpowiedź czy modele rzeczywiście zachowują wiedzę, czy też jedynie ponownie się jej uczą z przechowywanych danych. Proponuję metrykę Knowledge Retention, która kwantyfikuje, ile wewnętrznej reprezentacji jest zachowywane w kolejnych zadaniach, oferując wyraźniejszą miarę długoterminowego uczenia.

Zaprezentowane metody i algorytmy traktują odtwarzanie doświadczeń (ang. replay experience) jako kontinuum — od ukrytego przechowywania w parametrach po jawny projekt bufora, protokoły odtwarzania i strategie na poziomie partii, kończąc na narzędziach do bardziej rzetelnej ewaluacji. Opisane rozwiązania wzmacniają powtarzalność i interpretowalność odtwarzania doświadczeń w badaniach nad uczeniem ciągłym.

Abstract

Continual learning addresses the challenge of enabling machine learning models to acquire new tasks over time without losing performance on previously learned ones. The central difficulty is catastrophic forgetting, where new training overwrites earlier knowledge. This dissertation advances continual learning by investigating how replay-based methods construct, use, and evaluate memory.

First, I study memory construction. I introduce the Deep Features Buffer, which improves the diversity of stored examples by comparing feature representations rather than relying on random or reservoir sampling. I also propose Hebbian continual representation learning, where synaptic weight updates form an implicit buffer that preserves representations without explicit storage. I further contributed to Continual Visual Mapping, which replaces a dynamic memory buffer with a pre-trained, frozen large language model that produces language embeddings as stable anchors for each new class. These approaches provide complementary solutions for building memory.

Second, I analyze how replay is used. I show that the update protocol, whether replayed data is merged with or kept separate from new data, significantly changes optimization dynamics and can reverse the effectiveness ranking of selection heuristics. To address limitations of existing methods, I develop a new family of batch-level sampling strategies that evaluate replay batches using Batch Cosine Distance, ensuring they are collectively informative and non-redundant.

Third, I examine evaluation. Standard benchmarks often rely on accuracy, which can obscure whether models truly retain knowledge or merely relearn it from stored data. I propose the Knowledge Retention metric, which quantifies how much internal representation is preserved across tasks, providing a clearer measure of long-term learning.

Together, these contributions present experience replay as a sequence of design choices, from implicit parameter storage to explicit buffer design, replay protocols, and batch-level strategies, while providing tools for more reliable evaluation. The dissertation delivers both algorithmic advances and methodological guidance that improve the reproducibility and interpretability of experience replay in continual learning research.

Chapter 1

Introduction

1.1 Motivation

1.1.1 The Need for Continual Learning

Classical machine learning assumes that training and testing data are drawn independently from the same stationary distribution. A model is trained once on a large dataset and then expected to generalize to unseen examples from that distribution. This assumption underlies the remarkable success of deep learning in computer vision, natural language processing, and related fields.

In many real-world applications, however, data is not stationary. New classes appear over time, the statistics of the environment evolve, and previously collected data may no longer be accessible due to privacy or storage constraints. Retraining a model from scratch whenever new data arrives is often infeasible in terms of both computation and data availability. What is required instead are systems that can integrate new information as it becomes available, while continuing to perform well on previously encountered tasks. This capability, known as *continual learning*, is crucial for domains such as robotics, autonomous driving, and personalized recommendation, where models must operate reliably in dynamic and unpredictable conditions.

1.1.2 Catastrophic Forgetting

Standard neural networks, despite their success in static settings, are not equipped for continual learning. When trained sequentially on multiple tasks, they suffer from *catastrophic forgetting*—a rapid loss of performance on earlier tasks once new ones are introduced [61, 25]. This phenomenon is rooted in the mechanics of gradient descent: parameters are updated to minimize the current loss, regardless of their relevance to past knowledge, and as a result, features essential for earlier tasks are overwritten. Even simple benchmarks such as classical MNIST [50] dataset exposed to the model class by class, demonstrate the severity of this effect, with models collapsing from near-perfect accuracy on an initial class to near-random performance after exposure to subsequent classes.

Catastrophic forgetting is therefore a fundamental limitation that prevents neural networks from functioning in non-stationary environments.

1.1.3 Replay as a Dominant Paradigm

A variety of strategies have been proposed to mitigate forgetting. Regularization-based methods constrain parameter updates to preserve prior knowledge [36, 96]. Parameter isolation methods allocate separate subsets of the network to different tasks [78, 59]. Replay-based methods store a small set of past examples and interleave them with new data during training [75, 15].

Among these families, replay has emerged as the most widely adopted approach. It combines conceptual simplicity with empirical effectiveness and consistently establishes strong baselines across benchmarks. A compact replay buffer stabilizes training by reinforcing previously acquired representations while allowing adaptation to new tasks. This mechanism has proven robust across a wide range of continual learning scenarios. As a result, replay-based methods not only dominate empirical comparisons but also provide the foundation for many of the most competitive algorithms in the field.

1.2 Research Objectives

The overarching goal of this dissertation is to advance the understanding and effectiveness of replay-based approaches for continual learning in the supervised online class-incremental setting. While replay has become the dominant paradigm due to its empirical success and conceptual simplicity, many aspects of how memory should be constructed, maintained, and used remain underexplored. Furthermore, the evaluation of continual learning methods is still hindered by protocol inconsistencies and sometimes misleading metrics. This work addresses these gaps through three research objectives.

1.2.1 Memory Construction

The first objective is to investigate strategies for populating the replay buffer. Conventional reservoir sampling [90] provides a uniform but often redundant coverage of past data. This dissertation examines explicit memory construction, through sample selection strategies that promote diversity, implicit construction, through feature-level representations learned during training, and memory alternatives, which replaces the dynamic buffer with a fixed, pretrained language model that provides stable embeddings as anchors for each new class. The central challenge of explicit memory is to determine which samples from the incoming data are most valuable for preserving information about the current distribution. In the online setting, this task is particularly demanding, since only the current minibatch is available and decisions must be made without access to the full dataset.

1.2.2 Memory Usage and Protocols

The second objective is to analyze how replayed samples are best integrated with current data during training. Existing studies differ in whether replay samples are merged with the current minibatch or processed in a separate update step, and this seemingly minor implementation choice has been shown to significantly affect performance. Beyond such protocol differences, this work also investigates how replay batches should be formed and consumed. In particular, it explores batch-level replay strategies, which treat the minibatch as a collective unit rather than a collection of independent samples. This perspective allows for balancing informativeness, diversity, and representativeness at the batch level, and provides a path toward more effective utilization of limited memory.

1.2.3 Evaluation Metrics

Finally, the dissertation addresses the problem of evaluating continual learning systems. Standard metrics such as average accuracy and forgetting quantify overall performance but fail to distinguish true knowledge retention from mere relearning of stored data. To address this limitation, the dissertation introduces a novel *Knowledge Retention* metric that measures how well a model preserves useful internal representations during sequential learning. This contribution aims to make the evaluation of replay-based methods more reliable, interpretable, and comparable across studies.

Together, these objectives provide theoretical insights and practical methods to improve the reliability and interpretability of replay-based continual learning. The following section outlines the scope and limitations of this investigation, clarifying the boundaries within which the contributions are developed.

1.3 Scope and Limitations

The scope of this dissertation is restricted to *supervised online class-incremental continual learning*, with a few exceptions where additional experiments were performed. In this setting, data arrive as a sequence of tasks composed of disjoint sets of classes. During training, each sample is observed only once, except for those explicitly stored in a bounded replay buffer. The classifier is required to predict over all classes encountered so far, without access to task identifiers at inference time. This scenario is both practically relevant and methodologically challenging, as it combines the difficulty of incremental class expansion with the constraints of an online data stream.

Several important areas of continual learning fall outside the scope of this work. First, *generative replay* methods, which rely on synthetic data produced by generative models, are not considered here. While promising, such approaches introduce confounding factors related to the quality, cost, and stability of generative modeling. Second, methods based on *parameter isolation* or dynamic architecture expansion are used only as a baselines, as they pursue a different

line of inquiry that focuses on model capacity rather than memory efficiency. Third, large-scale unsupervised or self-supervised continual learning is beyond the intended scope. The focus is instead on supervised benchmarks, which provide a controlled environment for developing and evaluating memory-based methods.

By delineating these boundaries, the dissertation narrows attention to a well-defined problem: how to construct, use, and evaluate replay memory in supervised online class-incremental learning. This focus enables systematic analysis while leaving broader questions for future work.

1.4 Contributions

This dissertation makes four main contributions to the study of replay-based continual learning, organized across theoretical, algorithmic, methodological, and evaluation dimensions. Figure 1.1 illustrates where each proposed contribution fits within the replay pipeline.

1.4.1 Theoretical Contribution

A conceptual link is established between *explicit* replay strategies, which select and store individual samples, and *implicit* replay, which leverages feature-level representations to shape the memory buffer. By analyzing replay from the perspective of representation learning, the dissertation provides a unifying view that clarifies how different forms of memory construction affect stability–plasticity trade-offs in continual learning.

1.4.2 Algorithmic Contributions

Building on this theoretical foundation, several novel algorithms are introduced:

- **Deep Features Buffer (DFB)**: a memory construction strategy that increases within-class diversity by storing samples based on their feature representations.
- **HebbCL**: a biologically inspired replay algorithm that integrates Hebbian principles with continual learning.
- **Continual Visual Mapping (CVM)**: a memory buffer-free continual learner that replaces the classifier with fixed anchor vectors from a frozen language model, training a visual network to map images into this semantic space. This reduces forgetting and enables zero-shot transfer.
- **Batch Cosine Distance (BCD)**: a batch-level selection criterion that identifies replay minibatches based on collective representational change, providing a criterion for replay sample prioritization.

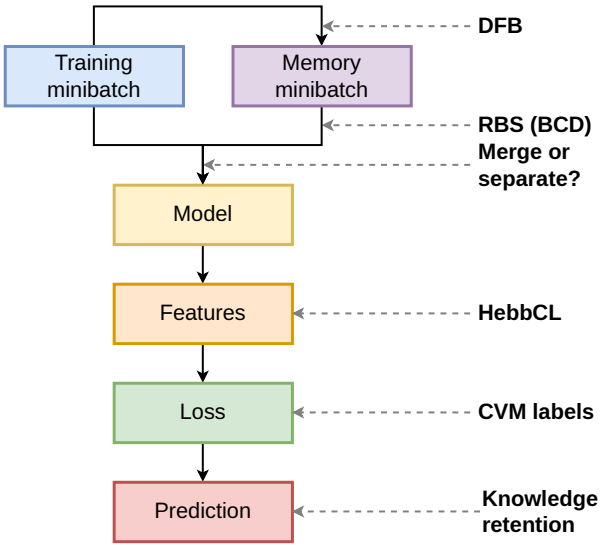


Figure 1.1: The figure maps where each contribution acts in the replay pipeline: memory construction (Deep Features Buffer, HebbCL, CVM anchors as class labels) and replay usage (protocol analysis, Batch Cosine Distance for batch sampling). Evaluation is covered by the Knowledge Retention metric.

- **Random Batch Sampling (RBS)**: a proof-of-concept batch-level strategy that leverages the BCD criterion to form replay batches with greater diversity and representational coverage.

These algorithms were evaluated against state-of-the-art baselines and were shown to improve performance in the supervised online class-incremental setting.

1.4.3 Methodological Contribution

A systematic analysis of replay protocols focuses on whether replay samples are merged with or separated from current data during training. The study demonstrates that this implementation choice can significantly alter experimental outcomes and, if unreported, introduces a confounding factor in empirical comparisons. By highlighting this issue and providing controlled benchmarks, the dissertation contributes to more transparent and reproducible practices in continual learning.

1.4.4 Evaluation Contribution

Finally, a novel metric, *Knowledge Retention*, is proposed to assess whether a model genuinely preserves internal representations over time, rather than simply relearning from stored data. This metric complements existing measures such as average accuracy and forgetting, offering a clearer view on the stability–plasticity

trade-off and the effectiveness of memory utilization. It provides a more reliable foundation for assessing replay-based continual learning methods.

1.5 Structure of the Dissertation

The remainder of this dissertation is organized into six chapters.

Chapter 2 provides the background and conceptual foundations of continual learning. It formalizes the problem setting, introduces notation and constraints, and surveys the continual learning scenarios. The chapter then reviews the principal continual learning method families—regularization, optimization-based, representation-based, parameter-isolation, and replay—with particular emphasis on replay-based approaches. Finally, it discusses connections to biological learning through Hebbian principles and the usage of multimodal representational space in continual learning.

Chapter 3 introduces the experimental framework. It reviews standard evaluation metrics, describes the benchmarks, datasets, and model architectures used throughout the dissertation, together with the protocols and implementation details that ensure reproducibility and fair comparison.

Chapter 4 addresses the problem of memory construction. It presents explicit strategies, particularly the Deep Features Buffer, as well as implicit strategies, such as Continual Visual Mapping and HebbCL, the latter inspired by Hebbian learning. The chapter analyzes the design and experimental performance of these strategies, as well as their comparative strengths.

Chapter 5 focuses on memory usage. It investigates replay protocols, showing how merging or separating replay samples with current data leads to different optimization trajectories and experimental outcomes. The chapter also examines sample selection from the buffer, including classical heuristics and gradient-based methods, and introduces batch-oriented strategies such as Random Batch Sampling.

Chapter 6 turns to evaluation. It analyzes the limitations of widely used metrics and introduces the Knowledge Retention (KR) metric, designed to capture whether a model truly preserves internal representations rather than merely relearning from memory. The case study illustrates the value of KR as a complement to traditional measures.

Chapter 7 concludes the dissertation. It summarizes the main contributions, discusses their implications, and outlines limitations. Finally, it highlights directions for future research.

Chapter 2

Background

CONTINUAL learning has emerged as a central direction in machine learning, driven by the need for systems that adapt to non-stationary environments without forgetting past knowledge. Before the proposed methods and contributions, this chapter establishes the conceptual and technical foundations. It begins with a formal definition of the continual learning problem, the notation used throughout the dissertation, and constraints that characterize realistic scenarios. It then surveys principal continual learning settings, reviews the main method families, and concludes with connections to biological learning mechanisms and multimodal representations inform parts of this work.

2.1 Problem Formulation and Notation

2.1.1 Formal Definition of Continual Learning

Continual learning can be formalized as sequential training over a stream of tasks. Let $\mathcal{T} = \{T_1, \ldots, T_N\}$ denote a sequence of N tasks. Each task T_i is associated with a distribution $\mathcal{D}_i$ over input–label pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}_i$. During training, data are revealed in order according to $\mathcal{D}_1, \ldots, \mathcal{D}_N$.

We write $S_i = \{(x, y) \sim \mathcal{D}_i\}$ for the (conceptual) dataset of task i and define the online stream as

$$\mathcal{S} = \big\langle (x_t, y_t, i_t) \big\rangle_{t=1}^{T}, \qquad (x_t, y_t) \sim \mathcal{D}_{i_t}, \tag{2.1}$$

with a single pass over $\mathcal{S}$ (each (x_t, y_t) is observed once unless stored in memory).

At inference time after seeing tasks $1{:}i$, the learner must predict labels from the cumulative label space

$$\mathcal{Y}_{1:i} = \bigcup_{j=1}^{i} \mathcal{Y}_j, \tag{2.2}$$

without access to task identifiers. In the class-incremental setting considered here, class sets are disjoint:

$$\mathcal{Y}_i \cap \mathcal{Y}_j = \varnothing \text{ for } i \neq j. \tag{2.3}$$

Prediction is over the cumulative space,

$$\hat{y} = \arg\max_{y \in \mathcal{Y}_{1:i}} f_\theta(x)_y, \tag{2.4}$$

where $f_\theta : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}_{1:N}|}$ denotes the classifier whose output head expands as new classes appear (inactive logits may be masked before class i).

Training minimizes cumulative loss on the non-stationary stream subject to a bounded replay memory:

$$\min_\theta \sum_{t=1}^{T} \ell(f_\theta(x_t), y_t) \quad \text{s.t.} \quad \mathcal{M}_t \subseteq \bigcup_{j \le i_t} S_j, \ |\mathcal{M}_t| \le M, \tag{2.5}$$

where ℓ is a supervised loss (e.g., cross-entropy), $\mathcal{M}_t$ is the replay buffer at step t, and M is the memory budget. Only elements of $\mathcal{M}_t$ may be revisited.

2.1.2 Constraints

Three constraints make continual learning particularly challenging. First, memory is bounded: only a limited number of past examples can be retained, which forces explicit strategies for sample selection and buffer management. Second, the data stream is presented in an *online* fashion, where each sample is observed once and must be processed immediately. This contrasts with offline training, where multiple passes over the dataset are permitted. Third, there are no task identifiers at inference time, the classifier predicts over all classes encountered so far. These constraints define the supervised online class-incremental setting that is the focus of this dissertation.

2.1.3 Relation to Adjacent Paradigms

Continual learning intersects with several established machine learning settings but differs in core assumptions.

Transfer learning assumes a fixed source and target domain [69]. Knowledge is acquired on a large source dataset and then adapted once to a target task. In CL, tasks arrive as an open-ended stream and the same model must integrate new tasks while preserving performance on all previous ones.

Multitask learning trains a single model on multiple tasks jointly, with all data available at once [12]. CL instead receives tasks sequentially and cannot revisit full past datasets, making interference and catastrophic forgetting central concerns.

Online learning updates a model as data arrive, typically under a single evolving task and with regret guarantees [85]. CL additionally requires retention across multiple distinct tasks and cumulative label spaces under bounded memory.

Active learning selects the most informative samples for labeling, typically under a stationary distribution [84]. In CL, sample selection concerns memory management rather than label acquisition and must handle non-stationary data.

Domain adaptation addresses distribution shift between a source and a target domain, often with access to both during training [5]. CL faces potentially unbounded shifts and must retain knowledge of all prior domains without full data access.

Curriculum learning orders training data to ease optimization but allows repeated passes and full-data access [6]. CL imposes single-pass streaming and long-term retention across tasks.

These comparisons highlight that continual learning is distinguished by the need for a single model to learn a sequence of tasks under bounded memory and without task identifiers at inference, a constraint not shared by adjacent paradigms.

2.2 Continual Learning Settings

Continual learning is not a single problem but a family of related scenarios that differ in the information available to the learner and the requirements imposed at inference time. The literature has converged on a taxonomy that distinguishes *task-incremental*, *domain-incremental*, and *class-incremental* learning, with each setting posing progressively stricter challenges [87, 18]. Figure 2.1 provides a schematic overview of all three settings. In addition, CL can be conducted in either *offline* or *online* regimes, depending on whether multiple passes over the training data are permitted. The following subsections summarize these settings.

2.2.1 Task-Incremental Learning

In the task-incremental setting, the learner is trained on a sequence of tasks and, at inference time, receives the task identifier together with the input. The prediction problem thus reduces to classifying within the label space of the current task. This setting is the least restrictive and often serves as a preliminary testing ground—task labels at inference allow separate classifier heads and greatly reduce interference between tasks.

2.2.2 Domain-Incremental Learning

In domain-incremental learning, all tasks share the same label space, but the input distribution shifts across tasks. Typical examples include changes in lighting, viewpoint, or style in visual datasets. The learner must generalize across domains without access to the domain label at inference time. This setting overlaps with research in domain adaptation and robust generalization but is cast here in a sequential framework. Because the label space remains fixed, catastrophic forgetting is usually less severe than in class-incremental learning, although distributional shifts still pose significant challenges.

2.2.3 Class-Incremental Learning

The class-incremental setting requires the learner to integrate new classes over time into a shared classifier without access to task identifiers at inference. After observing task i, the predictor must discriminate among all classes seen so far. This setting is widely considered the most challenging form of continual learning, as it combines interference between tasks with the necessity of expanding the decision boundary to cover an ever-growing set of classes. It is also the focus of this dissertation, as it most closely reflects real-world scenarios in which new categories appear over time and no external oracle provides task information.

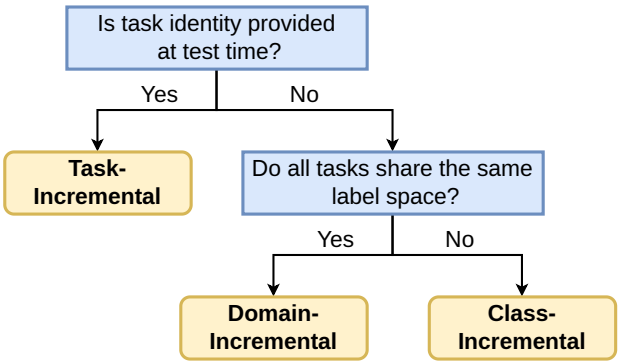


Figure 2.1: A decision tree illustrating the main continual learning settings. If task identity is available at test time, the problem is *task-incremental* learning. Without task identifiers, the key distinction is whether all tasks share the same label space, yielding *domain-incremental* learning, or whether the label space expands over time, which defines the more challenging *class-incremental* learning.

2.2.4 Online vs. Offline Continual Learning

Orthogonal to these settings is the regime in which training data are presented. In the offline regime, the learner may revisit task data multiple times, approximating conventional batch training with task boundaries. In the online regime, each sample is observed only once when it arrives from the stream. This restriction better reflects real-world constraints on storage and access, but also makes the learning problem substantially more difficult. Replay buffers become especially critical in the online setting, as they provide the primary mechanism for retaining access to past data under a bounded memory budget.

2.3 Families of Continual Learning Methods

Research in continual learning has produced a wide range of strategies to mitigate catastrophic forgetting. These approaches can be grouped into five major families: *replay-based*, *architecture-based*, *representation-based*, *optimization-based*, and *regularization-based* methods. Each family targets a different part of the learning process, and many recent algorithms combine elements across categories.

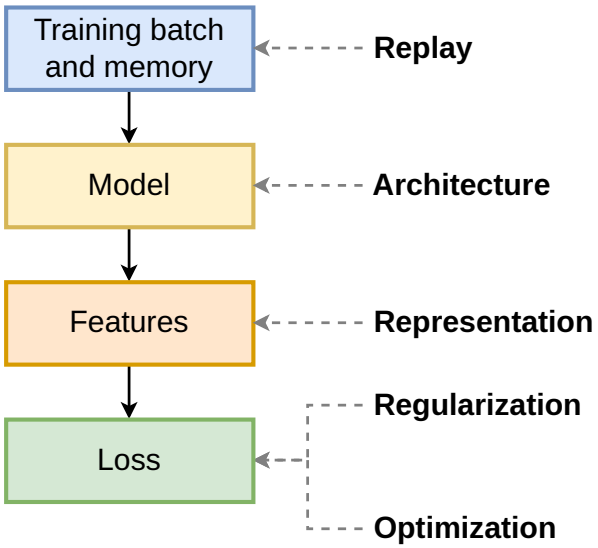


Figure 2.2: Illustration of the five main families of continual learning methods. Replay modifies the data stream; architecture methods alter the model structure; representation methods target feature learning; optimization methods reshape gradient updates; and regularization methods constrain parameter changes or the loss. Each family intervenes at a different stage of the learning pipeline.

2.3.1 Replay-Based Methods

Replay-based methods preserve a subset of past experiences and interleave them with new data during training. A simple and strong baseline is Experience Replay (ER) [15], where a small memory buffer is maintained with reservoir sampling to approximate uniform sampling over the observed stream. The strength of replay is its directness: by re-exposing the model to past data, it reinforces old decision boundaries while learning new ones. Replay often outperforms more complex approaches across benchmarks. The weaknesses are the need to allocate memory, potential redundancy in stored samples, and sensitivity to how replay is scheduled. Compared to other families, replay provides a simple and robust baseline.

2.3.2 Architecture-Based Methods

Architecture-based methods prevent forgetting by controlling which parameters of a network are used for each task. The most extreme solution is to train a separate network for every task, which avoids interference but scales poorly in memory and computation. A more efficient solution is to train a single shared model together with lightweight masks that specify which parameters are active for each task. PackNet [59], for example, allocates capacity by iterative pruning: parameters deemed less important for the current task are pruned and later reassigned to future tasks. Piggyback [60] and related methods refine this idea

by learning masks directly through backpropagation.

Mask-based approaches provide a clear intuition: each task carves out a subnetwork from the larger model, preventing interference while reusing shared weights. The strengths are memory efficiency (storing masks instead of full models) and faster inference compared to maintaining separate networks. The weaknesses are that task identifiers must be provided at inference, and forward or backward transfer between tasks is minimal.

2.3.3 Representation-Based Methods

Representation-based continual learning [18] aims to learn features that generalize across tasks, reducing interference. One direction uses contrastive or self-supervised objectives during the sequence to maintain transferable embeddings. For example, Co^{2}L learns with a contrastive loss and preserves past features via self-supervised distillation [13], while CaSSLe converts self-supervised learning losses into a distillation term that aligns current and past representations [22]. Another direction uses meta-learning to shape representations for fast adaptation [23].

The intuition is straightforward: if features capture task-agnostic structure in the data, then new tasks can be learned without overwriting old knowledge. The strength of this family is transfer: well-learned representations accelerate new tasks and reduce the need for replay. The weaknesses are reliance on strong inductive biases or auxiliary data, and difficulty in guaranteeing stability without additional mechanisms. Compared to replay, representation learning focuses on building robustness into the features rather than explicitly rehearsing past samples.

2.3.4 Optimization-Based Methods

Optimization-based methods modify the update rule to avoid interfering with past knowledge. Gradient Episodic Memory (GEM) [58] projects gradients of new tasks so that they do not increase the loss on stored examples from past tasks. Note that GEM relies on a small buffer, so it sits at the intersection of replay and optimization. Its simplified variant A-GEM [16] improves efficiency by approximating the projection. Other methods modulate updates based on estimated gradient conflicts or parameter importance

The intuition is that forgetting arises from conflicting gradients: new updates push parameters in directions that harm old tasks. By reshaping updates, optimization-based methods reduce destructive interference. Their strength is that they work without changing the architecture or requiring large buffers. Their weakness is computational overhead, since they often require gradient storage or additional constraints, and their effectiveness diminishes as tasks accumulate.

2.3.5 Regularization-Based Methods

Regularization-based methods penalize changes to parameters deemed important for past tasks. Elastic Weight Consolidation (EWC) [36] estimates importance using the Fisher information matrix, while Synaptic Intelligence (SI) [96] tracking each parameter's past-task contribution during training. Learning without Forgetting (LwF) [55] preserves past knowledge through distillation, matching the outputs of the old model on new data.

The central idea is that certain parameters matter more for past tasks, and the learner should resist updating them. The strength of these approaches is efficiency: they require no replay buffer and add little overhead. Their weakness is scalability: as tasks accumulate, the growing set of constraints hampers plasticity, and accuracy deteriorates. Compared to replay, regularization offers a lightweight alternative but often underperforms in long task sequences when replay has sufficient memory.

Figure 2.2 situates the five families within the learning pipeline. Replay intervenes at the data level, architecture methods modify the model structure, representation methods shape features, optimization reshapes updates, and regularization constrains the loss. Each family provides valuable tools: architecture ensures stability, representation promotes transfer, optimization manages updates, and regularization saves memory. Replay, however, is often used as part of or an enhancement to other methods, and it usually outperforms other families given a large enough memory budget.

2.4 To-Buffer Strategies

The simplest strategy for memory buffer population is *reservoir sampling*, which ensures that every incoming example has an equal probability of being retained [90]. Reservoir sampling is efficient and unbiased but often leads to redundancy, as similar samples may dominate the buffer.

2.4.1 Class-Balanced Reservoir Sampling

Class imbalance in the stream causes plain reservoir sampling to overrepresent frequent classes. *Class-Balancing Reservoir Sampling* (CBRS) counters this by enforcing balance across classes in a single pass [17]. CBRS operates in two stages: it fills memory as usual until capacity is reached; thereafter, for an incoming instance (x_i, y_i), if class y_i is *not* yet full, CBRS replaces a randomly chosen item from the currently largest class. If y_i is already full, CBRS replaces a random stored instance of the same class with probability

$$u \leq \frac{m_c}{n_c}, \tag{2.6}$$

where $u \sim \text{Uniform}(0, 1)$, m_c is the number of stored instances of class c, and n_c is the number of class-c instances observed so far. This rule preserves an i.i.d.

subset per class while mitigating stream imbalance and guarantees that severely underrepresented classes are retained when memory allows.

2.4.2 Partitioning Reservoir Sampling

Partitioning Reservoir Sampling (PRS) extends reservoir ideas to long-tailed and multi-label streams by partitioning memory into head and tail regions and maintaining balanced coverage across them [35]. As the stream arrives, PRS tracks class-frequency statistics and routes each incoming instance to the corresponding partition. Replacement is then performed within a partition, ensuring that minority concepts are not continually displaced by majority ones. PRS is particularly useful when inter- and intra-task imbalances coexist, where simple class-agnostic reservoir sampling yields poor minority retention.

2.4.3 Gradient-Based Sample Selection

The Gradient-Based Sample Selection (GSS) method addresses the challenge of populating the replay buffer in continual learning without relying on task boundaries or i.i.d. assumptions [2]. The core intuition is that selecting diverse examples, in terms of gradient directions, leads to a better representation of past knowledge, thus mitigating catastrophic forgetting.

Formally, let (x_t, y_t) denote the current input–label pair at time t. GSS aims to maximize the diversity of stored samples' gradient directions to approximate the feasible region defined by constraints from previously encountered samples:

$$\min_{\mathcal{B}} \sum_{i,j \in \mathcal{B}} \frac{\langle g_i, g_j \rangle}{\|g_i\| \, \|g_j\|}, \qquad g_i = \nabla_\theta \mathcal{L}\big(f_\theta(x_i), y_i\big), \tag{2.7}$$

where g_i is the gradient of the loss with respect to parameters θ for sample (x_i, y_i).

Due to computational complexity, GSS employs a heuristic procedure: each sample in the buffer is assigned a score based on the maximum cosine similarity with a random subset of other buffered samples. Incoming samples are compared against these scores, and samples with higher redundancy (i.e., higher cosine similarity) are probabilistically replaced by more diverse new samples. This greedy strategy maintains diversity in gradient space with practical efficiency.

2.4.4 Rainbow Memory

Rainbow Memory stores past data by applying multiple data augmentations to each candidate sample, estimating prediction uncertainty across those views, and keeping the samples with the highest aggregated uncertainty to build a diverse replay buffer for continual learning [4]. Originally proposed for blurry class-incremental scenarios, the strategy generalizes as a population policy that counteracts redundancy and preserves hard-yet-informative exemplars.

Diversity-aware and consistency-aware population strategies make more effective use of limited memory but add computational overhead and design

choices (e.g., score proxies, entropy estimation, augmentation strength). A central open question, and a focus of this dissertation, is how to construct replay memory that balances diversity, informativeness, and efficiency under strict online constraints.

2.5 From-Buffer Strategies

The application of experience replay in reinforcement learning shows that prioritizing certain samples (e.g., high-error or high-surprise) can accelerate training [80], sparking interest in whether non-uniform sampling could similarly benefit continual learning. Non-uniform sampling assumes that some samples are more informative for mitigating forgetting and adjusts selection probabilities accordingly. However, this comes at the cost of reduced sample diversity, which can introduce potentially harmful biases, especially given already limited memory.

2.5.1 Maximally Interfered Retrieval

Maximally Interfered Retrieval (MIR) challenges random replay by targeting samples whose loss would increase most under the incoming update [1]. The core intuition is that, at each training step, not all stored samples are equally at risk of being forgotten. If the past examples most adversely affected by the incoming minibatch can be identified, then replaying these *most interfered* samples more aggressively could help mitigate catastrophic forgetting. Figure 2.3 illustrates the two-model pass and top-k selection.

Formally, let θ denote the current model parameters, and let $\mathcal{L}_{\text{new}}$ be the loss on the newly arrived minibatch. The *virtual* gradient update on that minibatch is simulated as

$$\theta_v \;=\; \theta \;-\; \alpha\, \nabla_\theta \mathcal{L}_{\text{new}}(\theta), \tag{2.8}$$

where α is a step size (learning rate). For each sample (x_j, y_j) stored in the replay buffer, MIR approximates how that sample's loss would change under θ_v:

$$\Delta_j \;=\; \mathcal{L}\big(\theta_v; x_j, y_j\big) \;-\; \mathcal{L}\big(\theta; x_j, y_j\big). \tag{2.9}$$

Intuitively, a large positive Δ_j indicates that learning the new minibatch interferes substantially with (x_j, y_j). MIR then prioritizes replaying the top-k samples whose Δ_j values are greatest. MIR requires an extra forward–backward pass to form the virtual update, increasing compute per step.

Empirical results in online continual learning suggest that, especially under tight replay and compute budgets, MIR can outperform the basic experience replay. However, recent large-scale studies question whether non-uniform sampling really offers significant advantages. For example, the investigation on computationally budgeted CL [71] reports that under tight computational budgets, simple uniform replay outperforms more complex continual learning methods, including advanced sampling strategies. Similarly, the baselines for GRASP method [28] replicate a wide range of popular approaches and simple

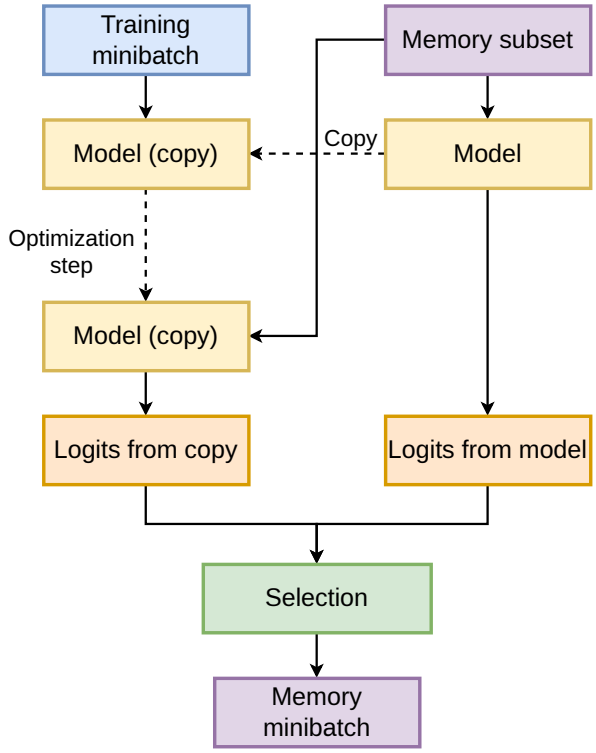


Figure 2.3: Workflow of Maximally Interfered Retrieval. A training minibatch updates a temporary model copy. Both the original model and the updated copy compute logits on a candidate memory subset. Samples with the largest loss increase are selected as the memory minibatch for rehearsal with the current task data.

heuristics under best practices (such as using a class-balanced memory) and find only marginal gains over uniform sampling. Recent analysis [88] shows that MIR is disproportionately focused on a small subset of the buffer, with just 10% of the memory accounting for 75% of all replays, highlighting a risk of overfitting due to extreme sample reuse.

2.5.2 Balanced Retrieval

MIR and similar approaches may reduce forgetting but can limit plasticity. To address this, Balanced Retrieval [94] proposes selecting samples from both extremes of the loss-change distribution—high loss change examples, suggested by MIR, to preserve past knowledge and low loss change examples, inverted to MIR, to maintain plasticity. In essence, each minibatch retrieves a balance of gradient-conflicting and gradient-aligned samples from the memory buffer.

2.5.3 GRASP

GRASP (GRAdually Select less Prototypical) [28] introduces a straightforward yet highly effective replay strategy that strongly relies on SIESTA [27] continual learning method with unique training setting combining online and offline training, not explored in this dissertation. Its core motivation stems from the observation that constantly replaying only the most challenging samples (i.e., those farthest from the class prototypes) can destabilize the learned representations, while replaying only the easiest examples may insufficiently reinforce harder decision boundaries. GRASP addresses this by progressively sampling from easy to harder samples within each rehearsal session.

These strategies highlight a trade-off: random sampling is computationally cheap and stable, while informed selection can improve accuracy but may introduce bias or higher computational cost. Compared to other families of continual learning methods, replay's strength is that even simple random sampling often outperforms more sophisticated alternatives, underscoring its robustness.

2.6 Replay Strategies

2.6.1 Dark Experience Replay

Dark Experience Replay (DER) [9] and its extensions DER++ and X-DER [8] illustrate how replay methods can augment standard experience replay with logit matching. We describe these variants to emphasize their shared reliance on *uniform* buffer sampling, an often overlooked limitation that motivates exploring non-uniform strategies.

DER stores both the raw inputs $\{x\}$ and the model outputs (logits) $\{z\}$ for past tasks. Training minimizes a cross-entropy loss on new data plus a distillation term on replayed samples:

$$\mathcal{L}_{\text{DER}}(\theta) = \underbrace{\mathcal{L}_{\text{CE}}(f_\theta(x_t), y_t)}_{\text{new data}} + \alpha \cdot \frac{1}{B} \sum_{j=1}^{B} \underbrace{\left\| h_\theta(x_j) - z_j \right\|_2^2}_{\text{distillation on stored logits}}, \tag{2.10}$$

where $h_\theta(x_j)$ is the current logit vector for past input x_j, z_j is its stored logit vector, B is the replay minibatch size, and α controls the weight of the distillation term.

DER++ adds a supervised cross-entropy loss on replayed samples, giving

$$\mathcal{L}_{\text{DER++}}(\theta) = \mathcal{L}_{\text{DER}}(\theta) + \beta \cdot \frac{1}{B} \sum_{j=1}^{B} \underbrace{\mathcal{L}_{\text{CE}}\big(f_\theta(x_j), y_j\big)}_{\text{CE on replayed data}}, \tag{2.11}$$

where β weights the additional supervised term.

2.6.2 Incremental Classifier and Representation Learning

Incremental Classifier and Representation Learning (iCaRL) [75] is another method that combines replay with distillation. It is based on three main components:

1. **Nearest-mean-of-exemplars classification.** Instead of a linear classifier, iCaRL keeps a small exemplar set P_y for each class y. For an input x, it predicts

$$\hat{y} = \arg\min_y \|\phi(x) - \mu_y\|, \qquad \mu_y = \frac{1}{|P_y|} \sum_{p \in P_y} \phi(p), \tag{2.12}$$

   where $\phi(\cdot)$ is the feature extractor. This prototype rule decouples the classifier from continual changes in the representation.

2. **Representation learning with rehearsal and distillation.** When new classes arrive, iCaRL updates ϕ using a joint loss of cross-entropy on current data and a distillation term on stored exemplars, which preserves performance on earlier classes.

3. **Exemplar management.** With a memory budget K, iCaRL allocates $m = K/t$ exemplars per class after observing t classes. A herding procedure selects exemplars to approximate the true class mean and supports later pruning while maintaining representativeness.

Its effectiveness relies on storing exemplars, which greatly mitigates forgetting.

2.7 Hebbian Plasticity and Implicit Memory

Biological systems learn continuously—integrating new information while retaining old skills—often under sparse and noisy feedback. This observation motivates mechanisms that preserve knowledge without external storage. Among them, Hebbian plasticity provides a local rule for forming synaptic traces that can function as *implicit memory*, a useful complement to explicit replay buffers in continual learning.

2.7.1 Hebbian Learning

Hebbian learning [32] states that synaptic strength increases when pre- and post-synaptic neurons co-activate, typically formalized as $\Delta w = \eta xy$. While simple and biologically grounded, the rule suffers from instability due to unbounded weight growth. Extensions address these issues: Oja's rule [68] normalizes weights and aligns with principal component learning; the covariance rule [83] subtracts mean activities to handle correlated inputs; the BCM rule [7] introduces

a dynamic threshold for potentiation and depression; and anti-Hebbian learning [24] supports inhibition and decorrelation.

Recent work adapts Hebbian mechanisms to deep learning. Hebbian-descent [62] stabilizes learning by centering activities and avoiding derivatives. HardWTA [66] integrates competition, inhibition, and BCM dynamics into convolutional models, achieving performance comparable to backpropagation. Neuron-centric Hebbian Learning (NcHL) [21] shifts from synapse-specific to neuron-specific updates, reducing parameters from $\mathcal{O}(W)$ to $\mathcal{O}(N)$ while remaining effective in reinforcement learning tasks. The Hebbian Forward-Forward algorithm [44] introduced an efficient supervised training method for deep, fully connected models. The Scalable Forward-Forward algorithm [47], a scalable variant for large convolutional neural networks, suggested a competitive alternative to backpropagation.

Together, these contributions demonstrate a resurgence of biologically inspired learning rules in deep models, offering alternatives that trade off architectural constraints or update granularity for scalability or plausibility.

2.7.2 Hebbian Learning as Implicit Memory

Hebbian plasticity provides a natural mechanism for storing information directly in synaptic weights. Patterns of co-activation leave a trace in the form of strengthened connections, which can act as distributed prototypes of past experiences. This implicit memory differs from explicit replay buffers: no samples are stored externally, but the network itself retains information through structural changes in its connectivity. Variants of Hebbian-inspired algorithms have been shown to approximate associative memory systems, where previously seen patterns can be recalled from partial input cues [37].

For CL, this perspective complements replay. Explicit buffers re-expose past samples to preserve plasticity, whereas Hebbian updates emphasize stability by reinforcing consistent correlations. Chapter 4 leverages this connection by introducing a Hebbian-inspired mechanism in which synaptic updates serve as an implicit buffer for representation preservation.

2.7.3 Connection to Forgetting

The relevance of Hebbian learning to continual learning lies in the stability–plasticity dilemma [26]. A system must remain plastic enough to acquire new knowledge while being stable enough to preserve what has already been learned. Purely gradient-based optimization often sacrifices stability, leading to catastrophic forgetting. Hebbian updates, by contrast, favor stability through the reinforcement of correlations, though they risk reduced plasticity if applied alone.

2.8 Multimodal Representations

Training models that connect modalities such as vision and text yields richer and more reusable representations. CLIP [72] jointly trains image and text encoders

with a contrastive objective so paired image–text embeddings are close in a shared space. This insight underlies a family of vision–language models that refine alignment and scaling, achieving strong zero-shot and transfer performance.

There are two dominant approaches to align modalities. Contrastive methods pull matched pairs together and push mismatches apart, offering efficiency and scalability (e.g., ALIGN [34], LiT [97]). Fusion and generative methods couple cross-attention with language generation or matching, capturing finer semantics and enabling captioning and VQA (e.g., ALBEF [52], BLIP [53], BLIP-2 [54]). Both yield semantically structured embedding spaces that can serve downstream continual learning.

2.8.1 Prompt-Based Continual Learning

Prompt-based continual learning has emerged as a way to leverage large frozen models with minimal extra parameters. **Learning to Prompt (L2P)** learns a prompt pool and dynamically selects a small subset per instance. This allows a backbone model to adapt to new tasks without rehearsal or requiring task identity at inference [92]. **DualPrompt** extends this by decomposing prompts into general and expert sets: the former encodes task-invariant information, the latter task-specific nuances, enabling more effective knowledge sharing across tasks [91]. **PIVOT** (Prompting for Video Continual Learning) shifts prompting to the video domain: it uses CLIP as a frozen image-language model and learns spatial and temporal prompts to handle evolving video tasks, thereby reducing forgetting in video class-incremental benchmarks [89].

These methods share the principle of freezing large pretrained backbones and learning lightweight components to adapt to new tasks. CVM follows a related concept but changes the objective: rather than prompting per task, it trains a visual encoder to map images into a fixed semantic space derived from a language model, eliminating the need for task-specific prompts or memory buffers.

Chapter 3

Experimental Framework

The effectiveness of continual learning methods cannot be judged in isolation from the experimental framework used to evaluate them. Choices of metrics, datasets, architectures, and protocols strongly influence results and can even change the relative ranking of methods. This chapter describes the framework adopted in this dissertation. It begins with evaluation metrics, highlighting both standard measures and their limitations. It then introduces the benchmarks and datasets, the model architectures employed, the evaluation protocols, and finally the implementation details required for reproducibility.

3.1 Evaluation Metrics

3.1.1 Standard Metrics

Let $a_{i,j}$ denote the test accuracy on task T_j after finishing training on task T_i ($1 \leq j \leq i \leq N$).

Average Accuracy [19] shows the ability of the model to maintain accuracy on all tasks encountered up to a given point. The average accuracy after completing task i is

$$A_i = \frac{1}{i} \sum_{j=1}^{i} a_{i,j}, \qquad A_N \text{ is reported after the final task.} \tag{3.1}$$

Here $a_{i,j}$ is the accuracy of the model after training on tasks 1:i when evaluated on task T_j. Average Accuracy is the most straightforward and commonly used metric, providing an overall assessment of performance across all tasks.

Average Forgetting [19] quantifies the loss of performance on previously learned tasks when a model adapts to a new task. For a given earlier task T_j with $1 \leq j < i$, let

$$\max_{k<i} a_{k,j}$$

denote the highest accuracy on T_j achieved before training on task i. The performance drop for T_j at step i is then

$$\max_{k<i} a_{k,j} - a_{i,j}.$$

Average forgetting at step i is the mean of these drops over all previous tasks:

$$F_i \;=\; \frac{1}{i-1}\sum_{j=1}^{i-1}\left[\max_{k<i} a_{k,j} - a_{i,j}\right]. \tag{3.2}$$

This metric reflects the model's capacity to retain knowledge across tasks. While widely used as an indicator of knowledge retention, our experiments show that accuracy alone may not capture true preservation of learned representations, so generalization ability should also be considered.

Backward Transfer (BWT) [19] captures how later learning changes earlier tasks at the end:

$$\mathrm{BWT} \;=\; \frac{1}{N-1}\sum_{t=1}^{N-1}\left(a_{N,t} - a_{t,t}\right). \tag{3.3}$$

Positive values indicate beneficial backward transfer, while the negative values indicate forgetting.

Forward Transfer (FWT) [19] quantifies zero-shot generalization to a task before seeing its labels:

$$\mathrm{FWT} \;=\; \frac{1}{N-1}\sum_{i=2}^{N} a_{i-1,i}. \tag{3.4}$$

Overall Performance [29] normalizes final average accuracy by a joint-training oracle:

$$\mathrm{OP} \;=\; \frac{A_N}{A_N^{\mathrm{joint}}}, \qquad A_N^{\mathrm{joint}} = \frac{1}{N}\sum_{j=1}^{N} a_{N,j}^{\mathrm{joint}}, \tag{3.5}$$

where $a_{N,j}^{\mathrm{joint}}$ is accuracy from a model trained once on the union of all tasks. This metric enables an incrementally trained algorithm to be compared relative to an offline trained algorithm.

Rescaled Average Accuracy / Forgetting [63] follow a similar approach by dividing Average Accuracy or Average Forgetting by the corresponding metric of the random classifier. With $C_i = |\mathcal{Y}_{1:i}|$ classes seen by step i, chance accuracy is $1/C_i$:

$$\mathrm{RAA}_i \;=\; \frac{A_i}{1/C_i} \;=\; C_i\, A_i, \qquad \mathrm{RAA}_N \;=\; C_N\, A_N. \tag{3.6}$$

For forgetting, report

$$\mathrm{RAF}_N \;=\; \frac{F_N}{1 - 1/C_N} \tag{3.7}$$

to place F_N in $[0,1]$ relative to its maximal range against a chance baseline. This decoupling of the task difficulty is allowing to identify whether performance changes are due to an increase in the number of classes or a failed stability-plasticity trade-off.

Intransigence [14] measures the inability to learn new tasks, for example, due to the loss of plasticity, compared to an oracle trained for that task:

$$\mathrm{I}_i \;=\; a_{i,i}^{\mathrm{solo}} - a_{i,i}, \tag{3.8}$$

where $a_{i,i}^{\mathrm{solo}}$ is the accuracy of a model trained from scratch on task T_i (or the joint oracle restricted to T_i).

3.1.2 Novel Perspectives

Standard accuracy summaries are informative, but they can hide how performance is achieved. In replay-heavy CL, a strong final score may arise from repeated exposure to a small set of stored examples, leading to high performance on those items without preserving broadly useful representations. This *buffer overfitting* appears when small changes to buffer contents produce large changes in results.

Accuracy and forgetting also combine two objectives: acquiring new tasks (plasticity) and preserving prior knowledge (stability). They do not indicate whether internal features were preserved or instead re-learned from whatever the buffer retained. Good end-of-sequence accuracy can therefore mask representational drift.

These issues motivate metrics that look beyond end-task accuracy and assess the state of the learned representation. Prior work argues for measuring representation stability and generalization beyond the replay set [30]. Chapter 6 introduces a complementary *Knowledge Retention* metric and evaluates it alongside accuracy and forgetting so that reported results reflect both plasticity and genuine retention, rather than buffer dependence alone.

3.2 Benchmarks and Datasets

Common to all datasets, tasks are formed by partitioning the class set into disjoint subsets. Training proceeds in a single pass over the stream with a bounded replay buffer. Data augmentation follows the default transformations provided by the Avalanche [11] library. Class order is fixed by ascending label index (from 0 to the last class).

3.2.1 MNIST

The MNIST dataset of handwritten digits [50] is adapted for CL by splitting the ten classes into disjoint tasks. A common variant trains sequentially on five two-class increments (e.g., $\{0,1\}$, then $\{2,3\}$, and so on). MNIST is appealing for its simplicity and speed of experimentation, but it is limited: low input dimensionality and high class separability make it a relatively easy benchmark that may not reflect real-world challenges.

3.2.2 SVHN

The Street View House Numbers (SVHN) dataset [65] contains digit images captured from house numbers in natural scenes (32×32 resolution), with substantial variability in background, lighting, and blur. Compared to MNIST, SVHN is more challenging and is useful for testing robustness to visual clutter while retaining the same label space. The task split follows the MNIST.

3.2.3 CIFAR-10 and CIFAR-100

The CIFAR datasets [39] contain 32×32 color images with greater variability than digit datasets. CIFAR-10 has ten classes, while CIFAR-100 has one hundred. In CL, CIFAR-10 is commonly split into 5 tasks of 2 classes each, and CIFAR-100 into 10 tasks of 10 classes each, yielding longer sequences. These benchmarks are standard for replay-based methods because they balance difficulty with manageable size and permit controlled large-scale comparisons.

3.2.4 Omniglot

The Omniglot dataset [49] consists of 1,623 character classes from 50 alphabets, with about 20 examples per class drawn by different writers. In CL, Omniglot probes one-shot and few-shot generalization: it tests whether models learn representations that transfer to new classes with little data. Its diversity across alphabets makes it well-suited to evaluating representation-oriented methods and implicit memory mechanisms.

3.2.5 CORe50

CORe50 is a continual object recognition benchmark comprising 50 household objects grouped into 10 categories, collected across 11 sessions with varying backgrounds and lighting, and released as RGB-D frames cropped at 128×128 resolution [57]. Classification can be performed at the *object* level (50 classes) or at the *category* level (10 classes). CORe50 defines three scenarios: *NI* (new instances), *NC* (new classes), and *NIC* (new instances and new classes). This dissertation adopts the class-incremental *NC* variant at the object level unless otherwise stated, with a fixed test set drawn from held-out sessions. The combination of session variability and growing label space makes CORe50 a strong test of robustness to both appearance change and class expansion.

3.2.6 TinyImageNet

TinyImageNet [11] is a reduced version of ImageNet with 200 classes, 64×64 images, and 500 training and 50 validation images per class. For CL, a common protocol splits the label set into 10 tasks of 20 classes each. The long sequence, limited replay budget, and high class diversity make TinyImageNet a severe test of scalability. Results on this dataset indicate how approaches may extend to larger-scale applications.

3.3 Model Architectures

The choice of neural network architecture directly influences the difficulty of continual learning experiments. Shallow models highlight forgetting in simple settings, while deeper architectures test scalability to more complex datasets. This dissertation employs three representative architectures.

3.3.1 Multilayer Perceptron (MLP)

For low-dimensional datasets such as MNIST, a multilayer perceptron serves as a simple baseline. A typical configuration includes two hidden layers with ReLU activations and fewer than one million parameters. MLPs are fast to train and allow controlled analysis of forgetting in settings where model capacity is not a limiting factor. Their weakness is poor scalability to natural image datasets, where convolutional architectures are required.

3.3.2 ResNet-18

ResNet-18 [31] is a widely used convolutional backbone with residual connections that stabilize training of deep networks. It offers a good balance of expressiveness and efficiency, with approximately 11 million parameters. Its widespread adoption in continual learning research makes it a standard choice for reproducible comparisons.

The SlimResNet-18 [75] used in our experiments follows the ResNet-18 topology but is substantially narrower and adapted to 32×32 inputs. The base channel width is set by a factor 20, producing stage widths of $[20, 40, 80, 160]$ instead of the standard $[64, 128, 256, 512]$, which reduces parameters by more than a factor of three. The initial stem is a single 3×3 convolution with stride 1 and no max-pool, and spatial downsampling occurs only through stride-2 convolutions in layers 2–4. A fixed 4×4 average pool replaces the global pooling used in the full model. Aside from these width and stem modifications, the network retains the four residual stages with two BasicBlocks each, matching the depth of a standard ResNet-18 while being far more lightweight for continual learning on CIFAR-sized images.

3.3.3 MobileNetV2

MobileNetV2 [79] is a lightweight convolutional architecture based on depthwise separable convolutions and inverted residual blocks. It is used for TinyImageNet experiments, where memory and computational efficiency are important. With significantly fewer parameters than ResNet-18, MobileNetV2 demonstrates how continual learning methods behave under resource constraints. It also reflects practical deployment scenarios where continual learners may need to operate on edge devices.

3.4 Experimental Hyperparameters

All continual learning experiments, except for Deep Features Buffer and HebbCL in Chapter 4, were implemented with the Avalanche library [11], which also provided the standard replay buffer based on class-balanced reservoir sampling.

Hyperparameter optimization was performed with Bayesian optimization, implemented in the GuildAI experiment tracking framework.

The key tuned parameters included:

- **Learning rate:** explored across a log scale from 1×10^{-6} to 1×10^{-1}.
- **Optimizer:** SGD with momentum (0.9) for all experiments in Chapter 4 (DFB, HebbCL and Continual Visual Mapping). In the rest of the experiments, it is Adam.
- **Batch size:** 32, unless stated otherwise, following the [9]. As suggested in [93], the memory batch is always identical to the current task batch.
- **Weight decay:** is always 0, which is common in online continual learning.
- **Dropout:** is used only for MobileNetV2, where is set to the default (0.2).
- **Epochs:** are set to 1, except for the CVM.
- **Protocol:** we are combining the current task minibatch with the memory minibatch in all experiments by default.

For each method, the configuration with the highest validation accuracy was chosen. Experiments were repeated with seeds $s \in \{0, \ldots, 10\}$, while all other sources of randomness were fixed across seeds. All experiments were performed on an NVIDIA GeForce GTX 1080 Ti (11 GB), an Intel Xeon E5-1620 CPU, and 64 GB of RAM.

Chapter 4

Memory Construction

This chapter presents three complementary approaches to memory construction in continual learning, developed through joint research with collaborators and published as *Diverse Memory for Experience Replay in Continual Learning* [48], *Hebbian Continual Representation Learning* [64], and *Continually Learn to Map Visual Concepts to Language Models in Resource-Constrained Environments* [74].

The initial motivation is the central challenge of experience replay in continual learning: with a fixed and limited buffer, a system must decide which experiences to retain without access to future data or complete task boundaries. On the class-balanced vision benchmarks common in the literature, where few samples are truly corrupted or redundant, simple reservoir sampling already approximates the underlying data distribution well, leaving little headroom for naive improvements. This chapter investigates how progressively richer representations can be used to overcome that limitation.

The first method, **Deep Features Buffer (DFB)**, keeps an explicit replay buffer but selects samples using their deep representations, discarding those with close neighbors in latent space to maximize diversity within each class. Building on this idea, **HebbCL** replaces the explicit buffer with an implicit one: it freezes sparse, Hebbian-learned representations to preserve past knowledge without storing raw examples. **Continual Visual Mapping (CVM)** extends this trajectory by removing stored representations entirely and guiding a compact visual model to align with fixed semantic anchors derived from a large language model, providing stable long-term structure without any replay memory.

Together these methods trace a path from carefully managed explicit memory toward purely representational strategies, offering a unified perspective on how continual learners can balance limited storage with the need to remember and transfer knowledge across tasks.

4.1 Deep Features Buffer

Experience replay maintains a bounded memory $\mathcal{M}_t \subseteq \bigcup_{j \leq i_t} S_j$ with $|\mathcal{M}_t| \leq M$ to mitigate forgetting on the non-stationary stream $\mathcal{S} = \langle (x_t, y_t, i_t) \rangle_{t=1}^{T}$. A

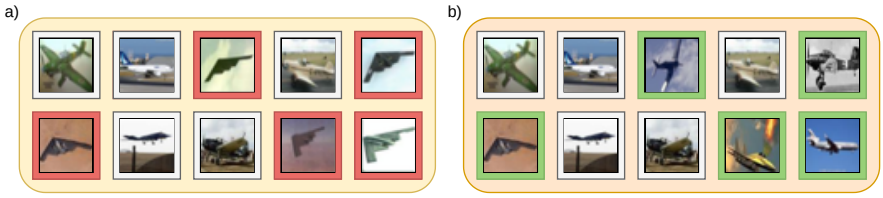


Figure 4.1: Conceptual illustration of the replay buffer. (a) Standard reservoir sampling admits near-duplicate samples within a class. (b) The Deep Features Buffer (DFB) maintains higher intra-class diversity by discarding close neighbors in latent space.

standard baseline is *reservoir sampling*, which inserts each incoming example with probability M/t, ensuring that $\mathcal{M}_t$ is an unbiased sample from all data seen so far. While effective on balanced benchmarks, reservoir sampling ignores representation structure and often stores many near-duplicates. Class-balanced extensions [2, 17] reduce class imbalance but still disregard similarity in the model's latent space.

4.1.1 Method

The Deep Features Buffer (DFB) augments the reservoir with a representation-aware eviction rule. At step t, let $B_t \subset \mathcal{S}$ denote the current mini-batch. First, draw a provisional subset $B_t^{\text{res}} \subseteq B_t$ using reservoir sampling. Let $r_l(x)$ be the activation of model layer l. For every $x \in B_t^{\text{res}}$ compute its representation $r_l(x)$ and insert the pair $(x, r_l(x))$ into $\mathcal{M}_t$.

If after insertion $|\mathcal{M}_t| > M$, periodically every T steps we refresh all stored representations

$$R_t = \{r_l(x) \,:\, (x, \cdot) \in \mathcal{M}_t\}. \tag{4.1}$$

We then compute all pairwise cosine similarities in R_t and repeatedly remove the sample from the most similar pair until $|\mathcal{M}_t| = M$. If labels are available and the two closest samples belong to different classes, both are retained and the next most similar pair is considered. This procedure implements a closest-pair search with expected $O(|\mathcal{M}_t|)$ time per update using a hierarchical clustering index.

Two hyperparameters govern DFB: the chosen representation layer l and the refresh interval T. Larger T lowers computation but may lag behind representation drift; $T = 1$ provides the most up-to-date features.

Experiments follow the protocol of Chapter 3: MNIST, SVHN, and CIFAR-10 are each divided into five class-incremental tasks. An MLP is used for MNIST and a ResNet-18 for SVHN and CIFAR-10. At each training step, the learner receives a mini-batch of size 10 from the current task and replays 10 samples drawn uniformly from $\mathcal{M}_t$ (none during the first task).

DFB preserves the unbiased sampling property of the reservoir while explicitly maximizing diversity in latent space, yielding higher accuracy and lower forgetting than plain reservoir sampling across all three datasets.

Algorithm 1 Deep Features Buffer update at step t

Input: model f_θ, mini-batch B_t, replay buffer $\mathcal{M}_t$, max size M, layer index l, refresh interval T
$B_t^{\text{res}} \leftarrow \text{ReservoirSample}(B_t)$
for $x \in B_t^{\text{res}}$ **do**
  $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup \{(x, r_l(x))\}$
end for
if $|\mathcal{M}_t| > M$ **then**
  if $t \bmod T = 0$ **then**
    refresh all stored representations $r_l(x)$
  end if
  while $|\mathcal{M}_t| > M$ **do**
    find pair (a, b) with minimal cosine distance
    remove a from $\mathcal{M}_t$ (unless a and b have different labels)
  end while
end if
return $\mathcal{M}_t$

4.1.2 Results

Table 4.1 reports average accuracy A_5 and average forgetting F_5 after all five tasks for MNIST, SVHN, and CIFAR-10. Across all datasets and memory budgets, Deep Features Buffer (DFB) matches or exceeds the baseline reservoir sampling.

MNIST. DFB improves accuracy by about 2–4% over reservoir sampling for both random and MIR replay when the buffer is small (50 or 100 samples) and reduces forgetting by a similar margin. At the largest buffer (200) DFB maintains accuracy while lowering forgetting slightly, indicating that latent-space diversity is most valuable when memory is tight.

SVHN. Improvements are smaller but consistent. With random replay DFB raises A_5 from 15.6% to 16.4% at M=50 and from 33.7% to 34.6% at M=200, and reduces forgetting in nearly all cases. Under MIR sampling DFB yields similar gains of roughly 1% in accuracy and modest reductions in F_5.

CIFAR-10. DFB provides steady benefits across all memory sizes. For random replay the largest margin appears at M=500 with accuracy rising from 29.6% to 31.1% and forgetting dropping from 54.6% to 53.3%. With MIR, accuracy improves by 1–1.5% and forgetting decreases by up to 2.5%.

Overall, DFB almost always improves both accuracy and retention, with the strongest effect on smaller, balanced datasets where uncontrolled redundancy in the buffer is more harmful.

4.1.3 Conclusions

Deep Features Buffer enhances experience replay by enforcing diversity in representation space rather than relying on class counts or raw data statistics. It preserves the nature of reservoir sampling while selectively removing near-duplicates, yielding higher accuracy and lower forgetting on three benchmarks and multiple memory budgets. The gains are most pronounced when the buffer is

Table 4.1: Average accuracy A_5 and average forgetting F_5 on all five tasks after learning all of them, for MNIST, SVHN and CIFAR-10. Each value is the average of 10 runs. *Sampling* is a strategy for selecting samples from the replay buffer for Experience Replay and *Memory* is the total size of the replay buffer.

MNIST					
Sampling	Memory	Reservoir		DFB (our)	
		$A_5(\%) \uparrow$	$F_5(\%) \downarrow$	$A_5(\%) \uparrow$	$F_5(\%) \downarrow$
Random	50	61.80 ± 2.63	38.08 ± 2.63	$\mathbf{63.80 \pm 1.65}$	$\mathbf{34.38 \pm 1.63}$
	100	75.91 ± 1.41	22.71 ± 1.67	$\mathbf{77.33 \pm 1.62}$	$\mathbf{21.56 \pm 1.78}$
	200	$\mathbf{86.55 \pm 1.47}$	13.08 ± 2.08	86.44 ± 1.33	$\mathbf{12.23 \pm 0.96}$
MIR	50	70.60 ± 2.24	28.95 ± 2.80	$\mathbf{74.64 \pm 2.07}$	$\mathbf{23.38 \pm 2.39}$
	100	83.80 ± 1.40	14.90 ± 1.09	$\mathbf{85.59 \pm 1.24}$	$\mathbf{13.03 \pm 1.47}$
	200	90.65 ± 0.86	8.84 ± 1.07	$\mathbf{91.26 \pm 0.88}$	$\mathbf{7.89 \pm 1.02}$
SVHN					
Sampling	Memory	Reservoir		DFB (our)	
		$A_5(\%) \uparrow$	$F_5(\%) \downarrow$	$A_5(\%) \uparrow$	$F_5(\%) \downarrow$
Random	50	15.63 ± 0.98	68.19 ± 0.66	$\mathbf{16.43 \pm 1.23}$	$\mathbf{67.65 \pm 1.12}$
	100	$\mathbf{23.03 \pm 1.88}$	$\mathbf{62.03 \pm 1.45}$	22.46 ± 1.81	63.18 ± 1.55
	200	33.73 ± 2.00	53.18 ± 1.64	$\mathbf{34.63 \pm 2.79}$	$\mathbf{52.66 \pm 2.58}$
MIR	50	21.81 ± 1.71	62.77 ± 1.44	$\mathbf{22.89 \pm 2.73}$	$\mathbf{62.18 \pm 3.08}$
	100	30.11 ± 3.24	56.28 ± 2.82	$\mathbf{31.18 \pm 2.52}$	$\mathbf{56.06 \pm 2.64}$
	200	42.79 ± 2.98	44.28 ± 2.94	$\mathbf{43.78 \pm 2.45}$	$\mathbf{44.28 \pm 2.08}$
CIFAR-10					
Sampling	Memory	Reservoir		DFB (our)	
		$A_5(\%) \uparrow$	$F_5(\%) \downarrow$	$A_5(\%) \uparrow$	$F_5(\%) \downarrow$
Random	200	23.84 ± 0.90	58.64 ± 0.64	$\mathbf{24.31 \pm 0.69}$	$\mathbf{58.24 \pm 0.82}$
	500	29.59 ± 0.73	54.58 ± 1.13	$\mathbf{31.09 \pm 1.04}$	$\mathbf{53.30 \pm 0.92}$
	1000	35.62 ± 0.99	48.42 ± 1.22	$\mathbf{35.78 \pm 1.52}$	$\mathbf{47.95 \pm 1.71}$
MIR	200	25.70 ± 0.98	57.67 ± 0.88	$\mathbf{26.67 \pm 0.95}$	$\mathbf{56.40 \pm 1.45}$
	500	31.59 ± 0.10	52.23 ± 1.10	$\mathbf{32.58 \pm 0.83}$	$\mathbf{50.53 \pm 0.93}$
	1000	38.27 ± 0.55	45.90 ± 0.96	$\mathbf{39.51 \pm 0.76}$	$\mathbf{43.37 \pm 1.74}$

small, confirming that diversity is critical when capacity is limited. These results demonstrate that even on clean, class-balanced streams where simple reservoir sampling is competitive, leveraging learned representations provides measurable advantages without increasing memory size or adding complex training objectives.

4.2 Hebbian Replay

The Deep Features Buffer maintained an explicit replay memory and selected stored samples using their latent representations. Hebbian Replay advances this idea by eliminating the buffer entirely and preserving knowledge directly in the

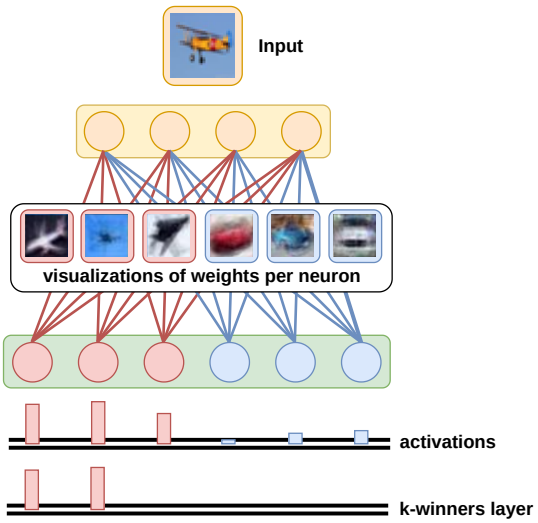


Figure 4.2: Weights of a one-layer network trained by HebbCL. Each $3 \times 32 \times 32$ patch shows the receptive field of a neuron. Red units activate on airplanes, blue on cars, demonstrating class-specific structure emerging without supervision.

network weights.

The method, called **HebbCL**, builds representations online without labels or task boundaries. It follows three biologically motivated principles:

- *Local Hebbian updates*: weights connected to highly active neurons are strengthened in proportion to input similarity, avoiding global back-propagation.
- *Sparse coding*: only the k most active neurons contribute to the representation, improving stability and interpretability.
- *Dynamic expansion and freezing*: when a neuron converges to a pattern its weights are frozen, and a new neuron is added to maintain capacity, creating an implicit memory of past inputs.

Because past information is stored as fixed receptive fields, the network can learn continually without replaying raw samples or their embeddings. The resulting weights form an interpretable basis that supports downstream tasks such as clustering and classification.

Although the main focus of this dissertation is supervised continual learning, Hebbian learning is inherently unsupervised, so we also evaluate HebbCL in that setting. Research on unsupervised continual learning is comparatively limited. Notable approaches include CURL [73] and CN-DPM [51], which construct mixtures of experts or latent distributions specialized to subsets of the data. HebbCL offers a more biologically plausible alternative and achieves competitive results in practice, demonstrating that local, unsupervised mechanisms can serve as an effective implicit memory.

Algorithm 2 Unsupervised HebbCL algorithm

Input: network (weights W), minibatch of training examples X
Hyperparameters: learning rate ϵ, threshold t
Output: updated weights W

for all $x_i \in X$ **do**
  $m \leftarrow \arg\max(Wx_i)$ {identify the highest activation}
  $\Delta w \leftarrow x_i - W_m$ {difference between input and weight vector}
  $W_m \leftarrow W_m + \epsilon \Delta w$ {update weights}
end for
$W_m \leftarrow W_m / \phi$ {normalize weights}

for all row vectors $W_j \in W$ **do**
  for all $x_i \in X$ **do**
    $distances[i] \leftarrow d^2(W_j, x_i) \;/\; ||x_i||_1$ {calculate normalized Euclidean distance}
  end for
  $m \leftarrow \arg\min(distances)$ {identify the shortest distance}
  if $distance[m] < t$ **then**
    freeze W_j {freeze a row vector of weights}
    add a new neuron {add a new row vector of weights}
  end if
end for

return W

4.2.1 Method

In a typical gradient–descent network all weights are updated to some degree for every example, including those critical for earlier tasks. Over time this leads to catastrophic forgetting. HebbCL avoids this by using purely local Hebbian updates and by freezing converged neurons so that essential weights remain unchanged.

HebbCL builds a representation with a single, wide, fully connected layer. This architecture has proved effective for Hebbian learning in other contexts [40, 56]. Extending Hebbian rules to deeper networks is possible but often fails to improve accuracy and can even degrade performance [3], so progress on this shallow setting directly benefits broader research.

Hebbian updates. For each minibatch the algorithm applies a simplified Krotov–Hopfield rule. Given input x, it identifies the most active neuron $m = \arg\max(Wx)$ and updates only its weights,

$$W_m \leftarrow W_m + \epsilon(x - W_m), \tag{4.2}$$

followed by normalization by the largest absolute weight ϕ. Because only the maximally activated neuron is modified, past representations remain largely

Algorithm 3 Supervised HebbCL algorithm

Input: network (weights W), training examples X from a single class C_j
Hyperparameters: learning rate ϵ, EPOCHS, a number of new neurons n
Output: updated weights W

denote unfrozen neurons as neurons belonging to class C_j

for e in range(EPOCHS) **do**
  for all minibatches B_k **do**
    for all $x_i \in B_k$ **do**
      $m \leftarrow \arg\max(Wx_i)$ {identify the highest activation}
      $\Delta w \leftarrow x_i - W_m$ {difference between input and weight vector}
      $W_m \leftarrow W_m + \epsilon \Delta w$ {update weights}
    end for
    $W_m \leftarrow W_m / \phi$ {normalize weights}
  end for
end for

freeze W {freeze all the weights}
expand the network with n new neurons {add n new row vectors to W}
return W

intact.

Expansion and freezing. After each minibatch every neuron j is checked for convergence by computing the normalized Euclidean distance

$$d_j = \min_{x \in X} \frac{\|W_j - x\|_2^2}{\|x\|_1}. \tag{4.3}$$

If $d_j < t$, the neuron is frozen and a new neuron is added, preserving model capacity while storing earlier patterns as fixed receptive fields.

Sparse representation. At inference time the feature vector for input x is $y = f(Wx)$, where f keeps only the k largest activations (k-winners-take-all), yielding stable and interpretable predictions.

Algorithms 2 and 3 give the full unsupervised and supervised procedures. The supervised variant trains on one class at a time, freezes all weights at the end of each class, and then adds n new neurons for the next class.

This combination of local Hebbian plasticity, neuron freezing, and dynamic expansion allows HebbCL to learn continually without storing raw samples or their embeddings, while retaining accurate and interpretable representations.

Unsupervised HebbCL

The unsupervised variant receives a single pass over an unlabeled stream with no task boundaries. Training proceeds in minibatches. For each example x, the

network computes activations $a = Wx$ and selects the most active neuron

$$m = \arg\max_{m'} (Wx)_{m'}. \tag{4.4}$$

Only this neuron is updated by the local Hebbian rule

$$\Delta w = x - W_m^i, \qquad W_m^{i+1} = W_m^i + \epsilon\, \Delta w / \phi, \tag{4.5}$$

where ϵ is the learning rate and ϕ is the largest absolute weight for normalization. As training proceeds, Δw decreases and W_m converges toward a stable input pattern.

To preserve earlier representations we freeze neurons that have converged. For each neuron j we compute the normalized squared distance

$$d_j = \frac{\|W_j - x\|_2^2}{\|x\|_1}. \tag{4.6}$$

If $d_j < t$, with threshold t tuned on validation data, the weights W_j are frozen and a new neuron is added to maintain capacity. Algorithm 2 summarizes the procedure, which is easily parallelized on a GPU.

Supervised HebbCL

With labels available, the same architecture and local update rule are used but freezing and expansion follow class boundaries. Classes $C_1, C_2, \ldots$ arrive sequentially. After training on all samples of class C_j for a fixed number of epochs, all current weights are frozen and the layer is expanded with n new neurons for the next class. Algorithm 3 gives the pseudocode.

At inference, activations of neurons assigned to each class are summed, and the predicted label is the class with the highest total activation. This supervised variant preserves the continual, non-replay character of HebbCL while exploiting label information to simplify neuron assignment.

4.2.2 Experiments

Continual Unsupervised Representation Learning

For unsupervised learning, we follow the CURL [73] evaluation protocol, which was introduced for continual unsupervised representation learning. We test HebbCL on MNIST and Omniglot. For MNIST, we use the default split between the training and test sets. For Omniglot, 50 alphabets serve as 50 classes, each with 15 training and 5 test examples.

Representations produced by HebbCL are compared with those from CURL, reported for both its variants with and without *mixture generative replay* (MGR). CURL is a continual unsupervised method that alternates training on real data and on samples generated by a snapshot of its own model, adapting the Deep

Figure 4.3: Visualization of weights: left MNIST (28×28 per neuron), right Omniglot (105×105 per neuron).

Generative Replay idea [86]. In MGR the model samples latent component labels y_{gen} from a learned mixture prior and generates synthetic observations

$$x_{\text{gen}} \sim p_{\theta_{\text{prev}}}(x \mid z_{\text{gen}}), \tag{4.7}$$

where θ_{prev} denotes the previous model snapshot. These generated samples are interleaved with the current data stream, reinforcing earlier concepts without storing past examples. CURL without MGR omits this synthetic replay and trains only on the incoming stream.

For evaluation we use two metrics: *cluster accuracy* and k-Nearest Neighbors (k-NN) error. Cluster accuracy is computed in three steps. First, the learned representations of the dataset are clustered with the k-means algorithm. Second, each cluster is assigned to the true class that occurs most frequently within it (true labels are used only for this mapping). Finally, using these assigned labels we measure the classification accuracy on the test set. The k-NN error is

Table 4.2: The average cluster accuracy and 10-Nearest Neighbors (10-NN) error across all five tasks of the MNIST and Omiglot, evaluated after learning the whole sequence. Each value is the average of five runs (with standard deviations). For CURL the average number of clusters is not an integer, because it was found by the algorithm.

MNIST			
Method	Clusters (№)	Acc (%)↑	10-NN error (%)↓
HebbCL (our)	25	$74.23_{\pm 0.67}$	$\mathbf{6.48_{\pm 0.20}}$
	50	$\mathbf{78.35_{\pm 0.73}}$	$\mathbf{6.48_{\pm 0.20}}$
CURL w/ MGR	$25.20_{\pm 2.23}$	$77.74_{\pm 1.37}$	$6.29_{\pm 0.50}$
CURL w/o MGR	$55.80_{\pm 1.94}$	$45.35_{\pm 1.50}$	$17.46_{\pm 1.25}$
Omniglot			
Method	Clusters (№)	Acc (%)↑	10-NN error (%)↓
HebbCL (our)	50	$14.07_{\pm 0.34}$	$\mathbf{71.62_{\pm 0.46}}$
	100	$\mathbf{16.46_{\pm 0.25}}$	$\mathbf{71.62_{\pm 0.46}}$
CURL w/ MGR	$101.20_{\pm 8.45}$	$13.21_{\pm 0.53}$	$76.34_{\pm 1.10}$
CURL w/o MGR	$189.60_{\pm 9.75}$	$13.36_{\pm 1.06}$	$81.91_{\pm 1.36}$

Table 4.3: The average accuracy across all five tasks of the MNIST and CIFAR-10, evaluated after learning the whole sequence of tasks. Each value is the average of five runs (with standard deviations).

Method	MNIST	CIFAR-10
SGD	19.01 ± 0.04	15.56 ± 5.08
L2	18.88 ± 0.18	16.31 ± 3.52
EWC	18.90 ± 0.06	11.83 ± 4.10
Online EWC	18.89 ± 0.07	15.49 ± 5.20
Synaptic Intelligence	17.94 ± 0.57	17.38 ± 4.13
MAS	17.38 ± 4.19	11.79 ± 4.01
LwF	49.37 ± 0.68	13.89 ± 5.33
EfficientPackNet	$\mathbf{99.42 \pm 0.15}$	—
HebbCL (our)	93.25 ± 0.38	$\mathbf{40.91 \pm 0.30}$

the percentage of test examples misclassified by a k-nearest-neighbor classifier operating directly in the learned representation space.

Results are given in Table 4.2. Omniglot, with 50 classes and high intra-class variability, is more challenging than MNIST. HebbCL outperforms CURL on Omniglot for both metrics and matches CURL on MNIST while requiring far less memory. For example, on MNIST HebbCL uses a single hidden layer with 500 neurons, several times smaller than CURL.

Figure 4.4: Visualization of weights trained on all tasks of CIFAR-10 in the supervised setting. Each square represents 32x32x3 weights connected to a given neuron. Here we show only neurons related to the first class (airplanes) with possible influence of other classes.

Figure 4.3 visualizes MNIST weights after training. Most filters capture meaningful structures and all classes are represented; noisy filters have little effect because the k-winners layer zeroes their activations. On Omniglot, HebbCL captures all alphabets, though some individual characters remain hard to distinguish.

Continual Supervised Learning

We next evaluate the supervised HebbCL on MNIST and CIFAR-10, each divided into five two-class tasks (Table 4.3). Training proceeds sequentially in the class-incremental setting, where task identifiers are unavailable at inference.

Because HebbCL uses no replay buffer or generative model, we compare it to regularization-based and parameter-isolation methods. On MNIST HebbCL surpasses all regularization approaches and approaches the best parameter-isolation method, EfficientPackNet [82]. CIFAR-10 is more demanding; we therefore use a wider layer with 2000 units (versus 640 for MNIST). Weights remain interpretable and class-specific (Fig. 4.4). HebbCL achieves much higher accuracy than regularization-based methods. EfficientPackNet has no published results for harder benchmarks (such as CIFAR-10) in this class-incremental regime.

Ablation study

HebbCL contains four key components: local Hebbian learning, weight freezing, dynamic neuron expansion, and the k-winners sparsity layer that zeros all but the k largest activations. To assess the contribution of each, we performed an ablation study in the unsupervised setting, measuring cluster accuracy and 10-NN error with 10 clusters for MNIST and 50 for Omniglot. Table 4.4 shows that removing any element degrades both metrics, and Fig. 4.5 illustrates the resulting changes

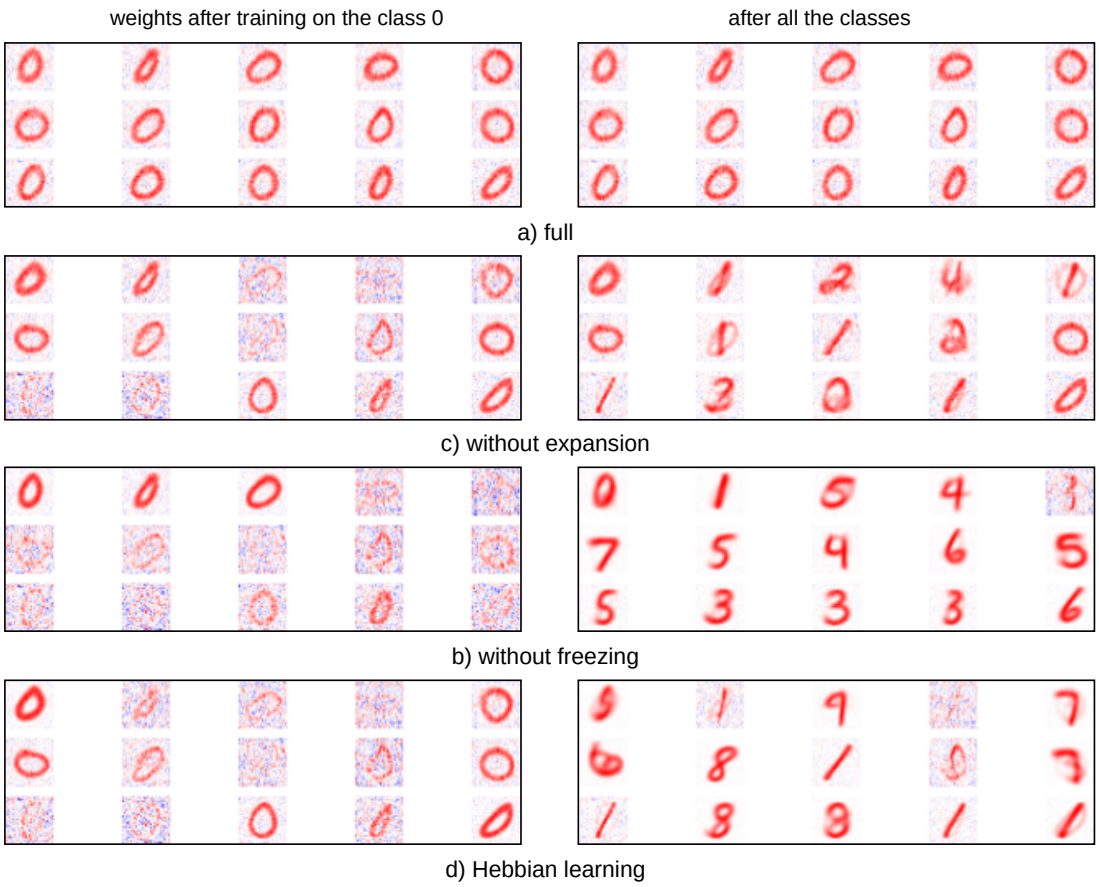


Figure 4.5: Investigation of catastrophic forgetting for different variants of HebbCL. Figures show visualizations of weights after training on the first class and the very same weights after a complete training on all classes. Clearly, in variants without freezing b) and d) the weights have been forgotten (overwritten) by more recent classes.

Table 4.4: Ablation study for HebbCL performed in an unsupervised setting. The sign ✓ indicates which steps were used in the experiment - (H)ebbian learning, (F)reezing weights, (E)xpansion and (K)-winners

Steps				**MNIST**		**Omniglot**	
H	F	E	K	Acc (%)↑	10-NN error (%)↓	Acc (%)↑	10-NN error (%)↓
✓	✓	✓	✓	63.09 ± 1.53	6.48 ± 0.21	14.07 ± 0.34	71.62 ± 0.46
✓	✓		✓	28.96 ± 0.43	8.79 ± 0.14	9.30 ± 0.19	81.95 ± 0.31
✓			✓	48.56 ± 0.36	7.99 ± 0.20	14.50 ± 0.27	79.81 ± 0.49
✓	✓	✓		35.62 ± 0.14	6.14 ± 0.05	12.45 ± 0.21	71.92 ± 0.53
✓				32.2 ± 0.15	6.8 ± 0.03	14.42 ± 0.16	60.18 ± 0.22

in the learned representations.

Without expansion, all neurons are available at the start and the network quickly exhausts its capacity, and once most neurons are frozen it cannot repre-

sent new classes. Without freezing, new tasks overwrite earlier patterns, causing severe forgetting. Disabling both freezing and expansion produces especially weak and indistinct representations. Omitting the sparse k-winners layer also lowers cluster accuracy significantly, particularly on MNIST.

On Omniglot the accuracy drop is less uniform, likely because the dataset's higher intra-class variability makes interactions between these mechanisms more complex and less predictable.

4.2.3 Conclusion

HebbCL tackles continual learning when task labels and boundaries are unknown, combining local Hebbian updates, neuron freezing, dynamic expansion, and sparse k-winners coding to build a stable and interpretable representation. It achieves state-of-the-art cluster accuracy and low 10-NN error on MNIST and Omniglot while using far less memory than CURL. The same principles extend naturally to supervised class-incremental learning, where HebbCL matches the best baselines on MNIST and clearly outperforms regularization approaches on CIFAR-10. Its learned weights correspond to human-readable prototypes, an advantage for safety-critical applications.

HebbCL demonstrates that a system can retain past information without maintaining an explicit replay buffer, as is done in the Deep Features Buffer. Yet, it can still perform competitively on unsupervised and supervised continual learning benchmarks by memorizing representations directly in the weights.

A key open challenge is scaling HebbCL to deeper networks and more complex datasets while preserving its stability and interpretability. Addressing this will further bridge representation-driven replay strategies and biologically inspired local learning as complementary tools for continual learning.

4.3 Continual Visual Mapping

Transformer architectures have shown that scaling data and model size consistently improves predictive performance across text, vision, and multimodal tasks. Large pre-trained Transformers now serve as reusable representations for a wide range of applications, but adapting or fine-tuning them typically demands substantial data and compute. Recent research therefore seeks efficient ways to transfer their knowledge to new tasks and domains without costly retraining.

Continual learning introduces an additional challenge: a model must learn from a non-stationary stream while retaining past knowledge. Explicit replay buffers, as used in the Deep Features Buffer (DFB), and implicit weight-based memory, as in HebbCL, address this for moderate-scale models. Continual Visual Mapping (CVM) extends this trajectory to the setting where the goal is not to store examples or latent representations at all, but to map incoming visual inputs directly into a fixed, high-capacity semantic space defined by a large frozen language model.

CVM uses a visual encoder, which is relatively small compared to LLMs, to

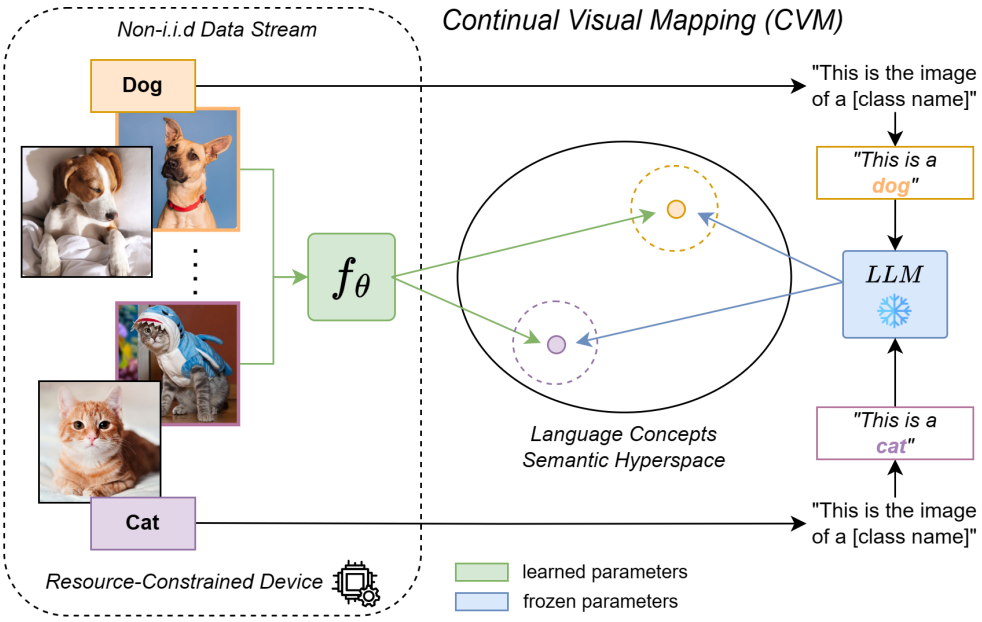


Figure 4.6: Conceptual image of our Continual Visual Mapping (CVM) method. A small and efficient parametric model f_θ is continually learned to map visual concepts into semantically richer embeddings extracted from frozen and pre-trained Large Language Models (LLMs).

align its outputs with static semantic anchors. This effectively distills the representational power of a language model into a continually updated vision module. Because the target space remains constant, the visual encoder can accumulate knowledge without replay or parameter growth, yielding a stable representation suitable for downstream tasks. This section presents the CVM algorithm, its architectural choices, and empirical evaluation on resource-constrained continual learning benchmarks.

4.3.1 Method

A standard classifier f_Θ consists of a feature extractor f_θ and a classifier c_ω, where θ and ω are trainable parameters. Given an input x_i, the prediction is

$$\hat{y}_i = c_\omega\big(f_\theta(x_i)\big). \tag{4.8}$$

In continual learning, repeated updates of θ and ω cause interference between tasks, so decision boundaries learned early are overwritten as new classes arrive. Because the model relies only on limited visual inputs, its representations become biased toward seen classes and transfer poorly to future tasks.

Anchor-based classifier. Continual Visual Mapping replaces the trainable classifier c_ω with a fixed set of *anchor vectors* that represent each class in a semantically rich latent space. These anchors are extracted once from a frozen large pre-trained language model. For each class c, we query the LLM with the prompt "This is an

image of *label*", where *label* is the class name, and store the resulting embedding C_c. This one-time query occurs only when new classes are detected. *The LLM is not used during training, so its weights always remain frozen.*

During inference, the class of x_i is predicted as the anchor with minimum cosine distance:

$$\hat{y}_i = \arg\min_{c \in C_t} d(f_\theta(x_i), C_c)\,, \tag{4.9}$$

where C_t contains anchors of all classes observed up to task t.

Triplet loss with margin α. To align the visual encoder with the fixed semantic space, we use a triplet loss [81] with cosine distance:

$$L_m = \frac{1}{N}\sum_{i=1}^{N} \max\left[0,\, d(f_\theta(x_i), C_i^P) - d(f_\theta(x_i), C_i^N) + \alpha\right], \tag{4.10}$$

where C_i^P is the positive anchor of the true class and C_i^N is a randomly chosen negative anchor. Random negatives performed better empirically than choosing the closest or farthest class.

Semantic distance loss. Because different classes may share semantic similarity, we regularize distances to previously seen classes to preserve their relative structure. Let $f_{\theta^{t-1}}$ be the encoder after task $t-1$ and C^{t-1} the corresponding anchor set. For current input x_i we minimize

$$L_d = \frac{1}{N}\sum_{i=1}^{N} d\Big(d(f_{\theta^t}(x_i), C^{t-1}), d(f_{\theta^{t-1}}(x_i), C^{t-1})\Big)\,, \tag{4.11}$$

which aligns the current distances to anchors of past classes with those produced by the previous model.

Total objective. The final training loss combines mapping and distillation terms:

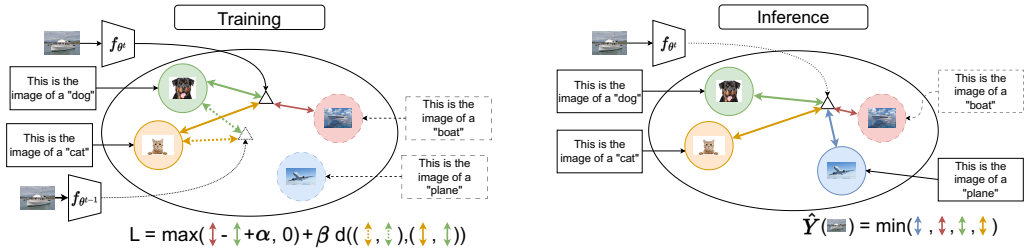
$$L_t = L_m + \beta L_d, \tag{4.12}$$

where β controls the strength of similarity preservation. Figures 4.7b and 4.7a illustrate inference and training, respectively.

By grounding a continually trained visual encoder in a fixed, semantically meaningful space, CVM transfers knowledge from a large language model without replay buffers or parameter growth, reducing forgetting and improving forward transfer.

4.3.2 Experiments

Simulating a resource-constrained environment, we use a reduced version of the ResNet architecture proposed in [75] for the visual model. The difference with the full-sized ResNet-18 is described in 3.3.2. The trainable part of CVM differs from the baseline methods only in increased numbers of channels of the last



(a) (b)

Figure 4.7: Diagram of how CVM works at Training and Inference. During training (a), when adding new classes (plane and boat), two losses are involved: the triplet loss with a margin α and the semantic distance loss restrained by coefficient β. At inference (b), the prediction $\hat{Y}$ is obtained by selecting the minimum distance between the image embedding and the seen classes in the latent space.

convolutional layer, to be the same as the length of the text representation. This modification allows us to remove the last linear layer, which means that with CVM, we train fewer parameters. To verify, we also ran experiments with the expanded convolutional layer for other methods but got similar results. For the text model, we use frozen SentenceBERT [76] to extract the textual representation for the classes.

We compare our proposal to multiple types of baselines described in Chapter 2. First, we compare our method against some classic continual learning methods like AGEM [16], EWC [36], and LwF [55]. We also compare against memory-based methods like ER [15], DER++ [9] and iCaRL [75]. iCaRL is especially interesting to our case since it a nearest-mean-of-exemplars classification strategy instead of linear classifier at inference time. However, the methods differ mainly in the training, as iCaRL uses a cross-entropy loss with labels instead of the distance to anchor vectors. The way iCaRL generates the vector per class is also different since here are generated from the samples in the memory of that same class by averaging the vectors.

In these experiments, we use an SGD optimizer with a learning rate of 0.1 and a batch size of 32. CIFAR-100 and TinyImagenet train for 50 epochs per experience, and for CORe50, we train for 30 epochs.

4.3.3 Results

Table 4.5 reports average accuracy for the class-incremental learning (CIFAR-100 and TinyImageNet) and domain-incremental learning (CORe50) benchmarks. As in prior work, regularization-only methods (Naive, AGEM, LwF, EWC) perform poorly on class-incremental learning, remaining near 7–9% accuracy because the sharp distribution shifts across tasks overwhelm their constraints.

Replay-based methods improve results substantially: ER reaches 21.0% on CIFAR-100 and 13.1% on TinyImageNet, and DER++ achieves 19.1% and 13.9%, respectively. Adding EWC to ER helps on TinyImageNet (17.7%) but lowers CIFAR-100 accuracy slightly.

	Class-IL		Domain-IL
	CIFAR-100	TinyImageNet	CORe50
Naive	9.0 ± 1.7	7.4 ± 0.1	23.1 ± 1.0
AGEM	9.3 ± 0.9	7.5 ± 0.1	26.0 ± 1.3
LWF	8.8 ± 1.5	7.4 ± 0.0	31.7 ± 2.2
EWC	8.6 ± 0.2	7.3 ± 0.1	24.0 ± 3.4
ER	21.0 ± 0.1	13.1 ± 0.2	33.0 ± 0.4
+ L_m	20.9 ± 0.7	10.7 ± 0.3	32.8 ± 1.3
ER + EWC	19.2 ± 0.4	17.7 ± 1.0	31.9 ± 0.1
+ L_m	25.2 ± 0.6	11.5 ± 1.2	32.2 ± 2.3
DER++	19.1 ± 0.8	13.9 ± 0.4	34.8 ± 2.3
+ L_m	21.0 ± 1.2	11.7 ± 0.4	33.1 ± 0.6
iCaRL	29.1 ± 1.1	24.3 ± 0.2	17.8 ± 1.8
CVM	**32.9** ± 0.4	**27.0** ± 0.8	**37.6** ± 1.1

Table 4.5: Accuracy for class-incremental learning benchmarks: CIFAR-100 and TinyImageNet, and CORe50. CVM outperforms all methods regarding average accuracy due to the combination of the change of the classifier and the semantic loss.

Replacing the cross-entropy loss with the mapping loss L_m (Eq. 4.10) shows mixed effects, when on CIFAR-100, ER+EWC gains about 6%, and DER++ gains about 2%. However, L_m does not benefit ER (21.0% $\rightarrow$ 20.9%) and even reduces accuracy on TinyImageNet for several methods. These results indicate that, while it can help when combined with EWC on CIFAR-100, *L_m is not universally advantageous alone*.

CVM incorporates L_m together with the semantic-distance loss L_d, which preserves relationships among previously seen classes. This combination delivers the best overall performance: 32.9% on CIFAR-100 and 27.0% on TinyImageNet, surpassing iCaRL by roughly 4.3%, respectively, and exceeding ER and DER++ by more than 10% on both datasets.

A similar pattern appears in the domain-incremental learning scenario. Regularization methods again lag behind, and replay methods perform best among baselines, with DER++ reaching 34.8%. Despite being replay-based, iCaRL struggles (17.8%) because domain shifts distort its class means. Here L_m brings no gains to ER or DER++, yet CVM attains 37.6% accuracy, the highest overall, confirming the benefit of aligning visual features to a fixed semantic space with semantic-distance regularization.

In the case of memory-based methods, increasing the memory size improves the representativeness of the buffer, which should help increase performance by decreasing forgetting. As shown in Table 4.6, DER++ is the method that scales the best as we increase the buffer size, significantly increasing its performance. However, it still achieves worse results than CVM, which consistently outperforms previous methods.

	Memory Size				
	500	1000	2000	3000	5000
ER	21.0%	28.2%	36.8%	40.5%	47.3%
+ L_m	20.9%	27.9%	37.7%	41.8%	47.3%
ER + EWC	19.2%	26.1%	35.0%	39.4%	43.6%
+ L_m	25.2%	26.1%	35.5%	39.1%	46.8%
DER++	19.1%	26.6%	39.3%	45.3%	50.9%
+ L_m	21.0%	28.2%	39.6%	43.1%	47.5%
iCaRL	29.1%	36.4%	37.7%	38.8%	39.6%
CVM	**32.9%**	**38.7%**	**43.5%**	**47.5%**	**51.4%**

Table 4.6: Accuracy of the CIFAR-100 benchmark when increasing memory size in different memory-based methods.

4.3.4 Conclusion

Continual Visual Mapping (CVM) introduces a continual learning strategy that anchors visual representations to a fixed semantic space derived from a frozen large language model. By mapping images to this knowledgeable latent space, a lightweight visual encoder can acquire new concepts while preserving relations among previously learned classes, without replay buffers or task-specific classifiers.

CVM advances the research path established by the Deep Features Buffer and HebbCL. DFB demonstrated the value of selecting diverse examples for explicit replay, while HebbCL eliminated the buffer by storing knowledge directly in network weights through Hebbian updates. CVM moves further by discarding both replay and implicit weight-based memory: it stores only static language-model embeddings and trains a vision module to align with them. This design enables experiments on far more complex datasets—such as CIFAR-100, TinyImageNet, and CORe50—while remaining efficient in data and compute.

Empirical results show that CVM achieves state-of-the-art accuracy in both class-incremental and domain-incremental settings, outperforming strong replay-based baselines under resource constraints. These findings indicate that continual learners can exploit the rich semantic structure of large language models to stabilize training and improve forward transfer, opening a direction for future methods that combine lightweight vision encoders with powerful frozen text models.

Chapter 5

Memory Sampling

The effectiveness of replay in continual learning depends not only on how the memory buffer is populated but also on how samples are drawn from it during training. This chapter follows that line of inquiry. The results presented were previously published in *Merging versus Separating Replay Samples in Continual Learning* [45] and *Batch Sampling for Experience Replay* [41].

We begin by examining how different **replay–update protocols** influence the impact of sample selection. Using a range of established heuristics, we compare two practical options: merging replay samples with the current minibatch for a single backward pass or processing them in a separate pass. A detailed analysis and extensive experiments reveal that this implementation detail alone can shift accuracy and even reverse the relative ranking of selection strategies.

Building on these insights, we move from evaluating existing heuristics to designing a new selection approach. Guided by active-learning principles of informativeness and diversity, we introduce **Random Batch Sampling** (RBS). RBS searches a small set of candidate batches and selects the one that most effectively complements the incoming data, yielding consistent gains over standard experience replay and strong baselines.

Taken together, these studies show that the way replay samples are chosen at each update is as critical as the content of the buffer itself, and careful attention to selection strategy is essential for reliable performance.

5.1 Merging Versus Separating Replay Samples

5.1.1 Replay Selection Strategies

Various heuristics have been proposed to improve on naive random selection from the memory buffer. Following the largest benchmarks [95, 28], we will compare the next variants:

Random: Randomly selects samples from the replay buffer without any specific criterion, serving as a baseline strategy.

Entropy: Selects samples based on prediction entropy $H = -\sum p_i \log p_i$, prioritizing samples with high uncertainty (high entropy) where the model is least confident.

Confidence: Selects samples with the highest maximum predicted probability $\max(p_i)$, focusing on examples where the model is most confident about its predictions.

Margin: Selects samples based on the difference between the top two predicted class probabilities, prioritizing samples with small margins where the model struggles to distinguish between classes.

K-means: Clusters sample embeddings using k-means clustering and selects the sample closest to each cluster center, ensuring diverse representation across the feature space.

Coreset: Implements greedy core-set selection by iteratively choosing samples that are farthest from already selected samples in the embedding space, maximizing coverage and diversity.

Bayesian: Uses Monte Carlo dropout to estimate predictive uncertainty by computing the difference between total entropy and expected entropy across multiple (10 in our experiments) forward passes.

MIR (Maximally Interfered Retrieval): Selects samples that would cause the largest increase in loss after a virtual gradient update, identifying samples that most interfere with the current learning objective. [1]

In the result tables, (Max) or (Min) refers to selecting samples with the highest or lowest scores, respectively.

5.1.2 Replay Protocols

Despite the progress in advanced selection heuristics, there is an understudied detail that can substantially affect performance: whether replay samples are merged with the current minibatch in a single backward pass or processed separately in a separate backward pass before a joint optimizer update. While some influential studies explicitly use separate backward passes (e.g., implementations of MIR and related methods [1, 67, 94]), popular continual learning libraries such as Avalanche [11] and Mammoth [8] default to a single merged backward pass. It is worth noting that one gradient computation step is more computationally efficient than two steps, which could be a strong argument for prioritizing the second method.

Although several publications provide valuable guidance on fair evaluation protocols for experience replay [10, 38], they do not investigate the potential impact of merging versus separating replay samples during training.

5.1.3 Theoretical Analysis

The replay-based continual learning is based on maintaining a small replay memory buffer $\mathcal{M}$ of previously observed samples. Let $D_t = \{(x_i, y_i)\}_{i=1}^{b}$ be the new data arriving at step t, and let $B \subset \mathcal{M}$ be the batch of replay samples sampled from $\mathcal{M}$. We then seek to update the model parameters θ to minimize the training loss on both new and replay data.

Consider a model f_θ parameterized by θ, and define the total loss on a set of samples S as $L(\theta; S) = \sum_{(x,y)\in S} \ell(f_\theta(x), y)$, where $\ell(\cdot, \cdot)$ is a per-sample loss

function.

Merging (Single Backward Pass). New and replay samples are concatenated into a single merged batch $D_t^{\text{merged}} = D_t \cup B$. A single gradient step updates θ:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta [L(\theta; D_t) + L(\theta; B)]|_{\theta=\theta_t} \tag{5.1}$$

Here, the gradient from new data and replay data is combined at the same parameter point θ_t. This is computationally cheaper but makes it impossible to distinguish which portion of the update was primarily driven by D_t versus B.

Separating (Two Backward Passes). New and replay data are processed in distinct backward passes. First, θ is updated using only the new data, then the updated parameter is used for a second update with replay data:

$$\theta_{t+1}^{(1)} = \theta_t - \eta \nabla_\theta L(\theta; D_t)|_{\theta=\theta_t} \tag{5.2}$$

$$\theta_{t+1} = \theta_{t+1}^{(1)} - \eta \nabla_\theta L(\theta; B)\big|_{\theta=\theta_{t+1}^{(1)}} \tag{5.3}$$

Crucially, the second gradient is computed at the updated parameter $\theta_{t+1}^{(1)}$, not at θ_t. Thus the two gradients are applied sequentially, leading to a different final θ_{t+1} than in the merging case.

Impact on Optimization Trajectory. Even if B is sampled randomly, merging vs. separating can produce different parameter updates. In merging, the step is:

$$\Delta_{\text{merge}} = -\eta [\nabla L(\theta_t; D_t) + \nabla L(\theta_t; B)] \tag{5.4}$$

computed once at θ_t. In separating, the second step depends on the updated parameter:

$$\Delta_{\text{sep}} = -\eta \nabla L(\theta_t; D_t) - \eta \nabla L(\theta_t - \eta \nabla L(\theta_t; D_t); B) \tag{5.5}$$

which typically differs from simply summing the two gradients at θ_t, because $\nabla L(\theta'; B)$ can change nontrivially once θ' has already been updated using D_t.

The fundamental difference between the protocols can be analyzed using Taylor expansion. The second gradient term in separating can be approximated as:

$$\nabla L(\theta_t - \eta \nabla L(\theta_t; D_t); B) \approx \nabla L(\theta_t; B) - \eta H_B \nabla L(\theta_t; D_t) \tag{5.6}$$

where H_B is the Hessian of $L(\theta; B)$. Therefore, the difference between the two update protocols is:

$$\Delta_{\text{sep}} - \Delta_{\text{merge}} \approx \eta^2 H_B \nabla L(\theta_t; D_t) \tag{5.7}$$

This difference term is a *random variable* that depends on the curvature structure of the replay samples (H_B) and the gradient direction induced by new incoming data ($\nabla L(\theta_t; D_t)$). Since both H_B and $\nabla L(\theta_t; D_t)$ vary unpredictably across different batches, tasks, and datasets, the relative performance of the two protocols is inherently inconsistent. This directly explains our empirical observation that neither protocol consistently outperforms the other across different experimental settings.

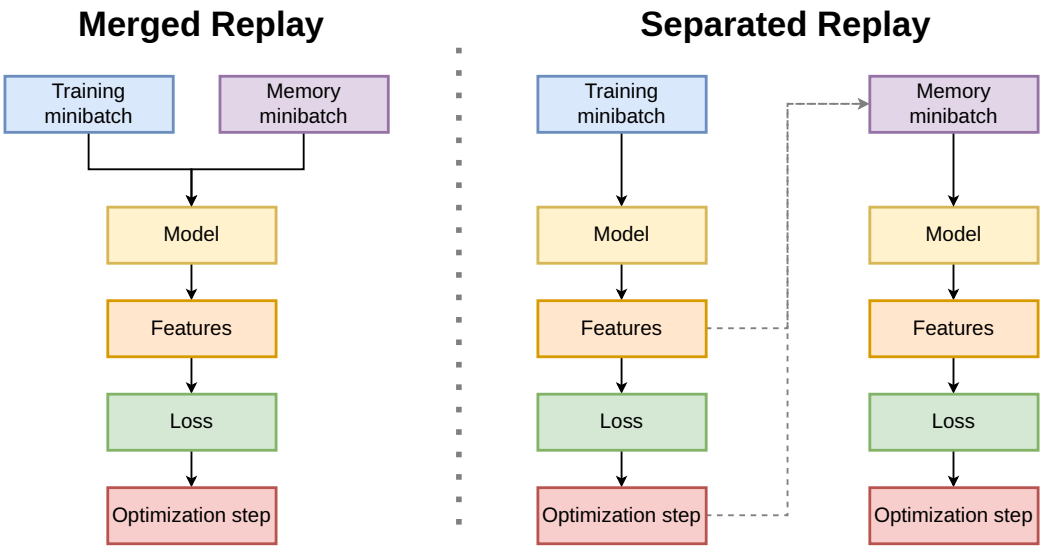


Figure 5.1: Comparison of Merged and Separated replay protocols.

5.1.4 Experiments

We evaluated replay-based continual learning approaches under different sample selection heuristics and two distinct model update protocols (Figure 5.1): (1) *Merging*, which combines replay samples with the current minibatch in a single backward pass, and (2) *Separating*, which processes replay samples in a separate backward pass before a joint optimizer update.

We tested three architectures, in addition to the ResNet-18 and MobileNetV2 (pre-trained with ImageNet) we are using a lightweight convolutional network with three 3×3 convolutional layers: 32, 64, and 128 channels. All models were optimized with either Adam (results in Table 5.1) or SGD (Table 5.3), using a single training epoch per new batch in the online setting. The learning rate was tuned to each combination of other parameters (ranging from 0.00005 to 0.2). The online and memory batch sizes were set to 32 for CIFAR-10/100 and 128 for TinyImageNet.

We compared random selection against a range of sample prioritization heuristics, including confidence, entropy, margin, Bayesian disagreement, clustering approaches (KMeans and Core-Set), and a loss-based interference criterion. We use the same random subsampling of size 50 for all strategies, as introduced in MIR. This step increases the diversity of the selected samples, preventing the strategy from selecting the same high-scoring samples for each mini-batch. The replay samples were selected by computing a score (e.g., the entropy of the output distribution) and taking either the highest or lowest scoring examples.

MIR strategy limitation. The version of MIR for the merged protocol differs from the original implementation. Since it relies on how the loss of a sample changes before and after new data is seen, two separate gradient computations are naturally required. The difference is that we use a separated protocol for

sample selection, zero the gradients, and then merge the selected samples with the current mini-batch for the merged protocol.

5.1.5 Results

Table 5.1 reports the final average accuracy on CIFAR-10 for various sample selection strategies under the online class-incremental setting, using both CNN and ResNet-18 architectures. We highlight two important observations.

First, the absolute performance of each heuristic varies considerably between merging and separating, sometimes reversing the ranking of the strategies. For instance, margin-based sampling shows competitive results under merging in certain cases (e.g., Margin (Min) with CNN), but does not consistently outperform random sampling when tested with separating.

Second, methods that rely on an additional gradient computation (e.g., MIR or other interference-based sampling variants) cannot be applied in the merging scenario without first computing the gradients for the current minibatch in order to select samples for the memory batch. This emphasizes the protocol-dependent nature of these strategies.

Table 5.1: Final average test accuracy (%) on CIFAR-10 under an online class-incremental continual learning setting with a fixed memory buffer of 200 samples and 5 task increments. Results are shown for different sample selection strategies using two network architectures (CNN and ResNet-18) with Adam optimizer, each evaluated under two update protocols. Results are averaged over 10 runs with different seeds (mean ± std).

	CNN		**ResNet-18**	
Strategy	Merging	Separating	Merging	Separating
Random	46.41 ± 1.62	42.71 ± 2.68	40.79 ± 1.27	38.23 ± 2.24
Bayesian (Min)	27.92 ± 1.27	29.19 ± 2.99	36.07 ± 1.45	42.65 ± 5.41
Bayesian (Max)	42.52 ± 2.36	46.04 ± 1.82	31.09 ± 3.88	33.75 ± 3.81
Confidence (Min)	38.22 ± 1.85	39.74 ± 1.84	34.46 ± 3.10	37.02 ± 2.81
Confidence (Max)	21.67 ± 1.37	26.73 ± 4.04	21.75 ± 0.86	22.19 ± 1.99
Core-Set	29.58 ± 2.22	34.64 ± 1.33	35.70 ± 2.64	38.58 ± 3.33
Entropy (Min)	43.00 ± 1.50	42.35 ± 2.92	35.14 ± 3.44	36.11 ± 2.55
Entropy (Max)	29.66 ± 2.02	30.57 ± 2.19	22.52 ± 1.18	21.99 ± 1.51
K-Means	41.83 ± 1.39	41.06 ± 2.35	37.51 ± 2.87	41.49 ± 1.72
Margin (Max)	32.26 ± 1.87	31.05 ± 2.62	21.26 ± 1.37	23.61 ± 2.58
Margin (Min)	45.27 ± 0.81	44.05 ± 1.80	38.90 ± 2.76	40.61 ± 1.47
MIR	46.00 ± 1.10	42.99 ± 1.05	40.77 ± 2.40	42.02 ± 2.13

As shown in Table 5.2 , there is a clear trade-off between accuracy and computational cost, which is notably affected by memory size. Across the CIFAR-10 and CIFAR-100 datasets, increasing the memory size from 200 to 1000 consistently improved the accuracy of the MIR strategy. The MIR strategy

Table 5.2: Final test accuracy (%) and average time of run in minutes for different selection strategies across datasets and memory sizes. The time is measured in minutes.

CIFAR-10 - ResNet-18					
		Merging		**Separating**	
Strategy	**Memory**	Acc. (%)	Time	Acc. (%)	Time
Random	200	40.18 ± 0.85	1.0	36.04 ± 0.66	1.2
Margin (Min)		35.74 ± 0.69	2.4	38.43 ± 0.71	2.7
MIR		41.45 ± 1.62	6.5	41.06 ± 4.22	3.2
Random	500	45.58 ± 1.11	1.0	37.02 ± 4.05	1.2
Margin (Min)		47.25 ± 0.26	4.8	44.30 ± 0.03	5.1
MIR		49.70 ± 1.44	6.5	33.40 ± 3.53	5.6
Random	1000	47.78 ± 0.05	1.0	33.88 ± 3.75	1.3
Margin (Min)		44.28 ± 1.54	4.9	30.56 ± 5.42	5.2
MIR		54.02 ± 1.81	6.5	51.65 ± 0.56	3.1
CIFAR-100 - ResNet-18					
Random	200	13.48 ± 0.90	1.3	14.35 ± 0.43	1.7
Margin (Min)		13.44 ± 0.19	2.8	14.52 ± 0.26	3.2
MIR		14.30 ± 0.41	6.6	15.22 ± 0.08	3.7
Random	500	17.14 ± 0.78	1.3	17.60 ± 0.32	1.7
Margin (Min)		17.58 ± 0.12	4.3	18.67 ± 0.03	4.0
MIR		18.19 ± 1.30	6.6	18.40 ± 0.77	4.4
Random	1000	19.97 ± 0.28	1.3	20.79 ± 0.88	1.6
Margin (Min)		19.96 ± 1.18	5.0	20.48 ± 0.22	4.7
MIR		20.97 ± 0.21	6.6	21.11 ± 0.38	3.7

achieved the highest accuracy (54.02% and 21.11%, respectively), but it incurred significantly higher computational costs, compared to random sampling. Despite its moderate accuracy improvements over Random at smaller memory sizes, the Margin (Min) strategy becomes less effective at larger memory sizes, as is particularly evident on CIFAR-10. Additionally, as expected, the separating protocol is slightly slower than the merging protocol.

Table 5.3 focuses on the random selection baseline, comparing merging and separating across CIFAR-10, CIFAR-100, and TinyImageNet. We see no universal trend favoring one update protocol over the other. For example, on CIFAR-10, merging yields a higher average accuracy than separating, whereas on CIFAR-100 the opposite is observed. In both experiments, the performance differences between the two approaches were found to be statistically significant, as confirmed by paired t-tests and Wilcoxon signed-rank tests ($p < 0.01$).

Across extensive hyperparameter sweeps involving multiple learning rates, batch sizes, and two popular optimizers (Adam and SGD), no consistent per-

Table 5.3: Comparison of the model update protocols with SGD optimizer for the random sampling baseline. Results are averaged over 10 runs with different seeds (mean ± std).

Dataset	Model	Merging	Separating
CIFAR-100	CNN	41.43 ± 1.47	43.13 ± 1.32
CIFAR-10	CNN	41.68 ± 1.18	38.19 ± 1.57
TinyImageNet	MobileNetV2	26.80 ± 0.43	25.13 ± 0.51

formance pattern emerged favoring one update protocol over the other. This further reinforces our conclusion that the optimal choice between merging and separating is context-dependent and should be reported explicitly to ensure fair comparisons. A complementary analysis of replay sampling (see Appendix A) shows that post-hoc selection of random non-uniform distributions provides a clear empirical upper bound, revealing measurable performance headroom beyond uniform sampling [46].

5.1.6 Conclusions

We examined a seemingly minor implementation detail, the choice to *merge* or *separate* replay samples from the current minibatch during training, and showed that this replay protocol can strongly influence the effectiveness of experience replay in continual learning. Extensive experiments across datasets, model architectures, and sampling heuristics revealed that switching between these two update protocols can produce substantial changes in overall accuracy. In some cases, a replay strategy that clearly outperforms a random baseline under one protocol fails to do so under the other.

These findings highlight the need for greater transparency in continual learning research. When introducing new replay methods, authors should specify whether replay samples are merged into the main training step or processed in a dedicated backward pass. Omitting this detail can obscure comparisons and invite misinterpretation, especially when baselines use different update routines. Widely used continual learning libraries also differ in their default settings, reinforcing the importance of standardization for fair and reproducible benchmarking.

5.2 Batch Sampling

The challenge of sample selection is that, according to the active learning literature [20], the batch has three measures that influence its potential performance: *informativeness*, *diversity*, and *representativeness*. The difference between batches selected with only one dominant measure is shown in Figure 5.2.

Informativeness measures how useful a sample could be to update the model. This can be based on data attributes, like class probability distributions, or on the

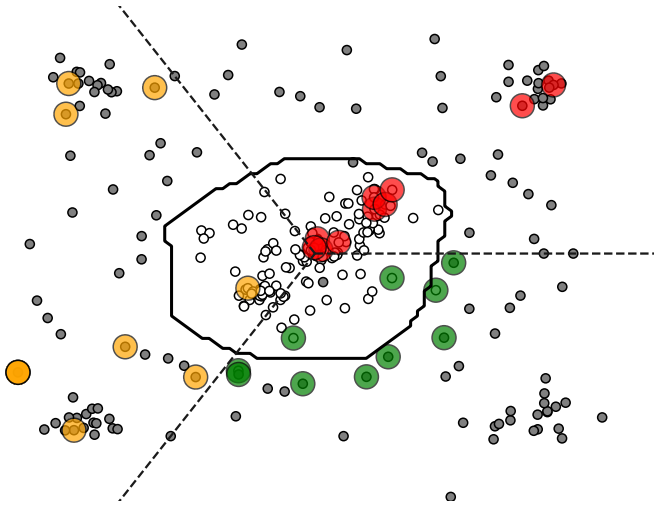


Figure 5.2: Visualization highlighting three key properties of batch selection on an example of a binary classification problem: informativeness, where the samples (green) provide valuable insights by their proximity to the decision boundary; representativeness, where the samples (red) capture the predominant trends of the dataset by clustering in high-density areas; and diversity, where the samples (yellow) are uniformly selected across the dataset, ensuring a comprehensive representation. Black and white circles denote samples of distinct classes, and the solid line indicates the decision boundary. The dashed lines separate regions where different selection strategies have been applied.

model's current state, such as proximity to the decision boundary. For our work, we use informativeness as a metric for its potential to counteract forgetting.

Representativeness captures how well a sample represents the underlying data distribution. Methods for quantifying this can range from density estimation techniques to average distance calculations with nearest neighbors. In some cases, cluster medoids are chosen as representative samples.

Diversity refers to heterogeneity within a batch of samples, preventing redundancy and information overlap. Techniques to ensure diversity could involve maximizing pairwise distances or employing clustering methods to choose samples from various clusters.

It is difficult to predict the importance of each property in a specific training step of the model and approximate all into one metric. Therefore, the problem was reduced to combining or simultaneously improving only the informativeness and diversity. We imply that *representativeness* was secured by the method of selecting samples to the memory buffer. For the same reason, we did not address the problem of outliers that may occur in the memory buffer.

5.2.1 Method

Algorithm 1 describes a continual learning procedure that extends standard experience replay by introducing two linked contributions.

Batch Cosine Distance (BCD) is a ranking metric designed to identify the

Algorithm 1: Replay Batch Sampling (RBS) with Batch Cosine Distance (BCD)

Input: Tasks T, learning rate α, replay batch size $\mathcal{B}$, number of trials $\mathcal{N}$

1 **Initialize:** memory $\mathcal{M}$; current model θ
2 **for** $t \in 1..N$ **do**
3 **for** $B_n \sim T_t$ **do**
4 $\theta^v \leftarrow SGD(B_n, \alpha)$ `// temporary model updated on the online batch`
5 **for** $i \in 1..\mathcal{N}$ **do**
6 $B_{M_i} \sim RandomSampleMemory(\mathcal{M}, \mathcal{B})$
7 $E_\theta \leftarrow ExtractEmbeddings(B_{M_i}, \theta)$
8 $E_{\theta^v} \leftarrow ExtractEmbeddings(B_{M_i}, \theta^v)$
9 $C_i \leftarrow CosineDistance(E_\theta, E_{\theta^v})$
10 **end**
11 $B_{opt} \leftarrow SelectMaxDistance(C)$
12 $\theta \leftarrow SGD(B_n \cup B_{opt}, \alpha)$
13 $\mathcal{M} \leftarrow UpdateMemory(B_n)$
14 **end**
15 **end**

memory batch that induces the greatest representational change relative to the current model. When a new online batch B_n arrives, the model parameters are temporarily updated to θ^v with a single SGD step. For each candidate replay batch B_{M_i} randomly drawn from the memory buffer $\mathcal{M}$, we extract normalized embeddings from both the current model θ and the temporary model θ^v. The cosine distance between these embeddings, C_i, quantifies how strongly B_{M_i} would alter the internal representations if learned together with B_n. The batch with the maximum distance, B_{opt}, is therefore expected to provide the most informative rehearsal signal.

Replay Batch Sampling (RBS) is a proof-of-concept selection strategy that applies BCD. The space of possible replay batches is astronomically large (for example, more than 2.63×10^{23} combinations when sampling 10 examples from a memory buffer of size 1000). Since the exhaustive search is infeasible, at each model update RBS samples only nine candidate batches uniformly at random and evaluates them with BCD. To facilitate comparison with existing methods, we also include a single batch chosen by the MIR heuristic, giving a total of ten BCD evaluations per update. Although it is possible to replace most, if not all, of the minibatch candidates with those proposed by the MIR or other heuristics, we are demonstrating the potential of the ranking metric using the simplest approach. This idea was extended to inter-batch diversity [43], suggesting that the diversity of samples within a batch and across a sequence of batches should positively impact optimization.

The remainder of Algorithm 1 shows how these two ideas are combined. For each new online batch, the temporary model θ^v is created (line 4). Lines 6–10 perform the RBS loop: a small set of candidate memory batches is drawn, their embeddings are compared using BCD, and the batch with the highest cosine

Table 5.4: The average accuracy across all five tasks of the MNIST and CIFAR-10, evaluated after learning the whole sequence. Each value is the average of 20 runs with standard deviation.

CIFAR-10				
Memory	ER	Class-balanced	MIR	**RBS**
200	22.54 ± 2.00	23.38 ± 1.64	24.39 ± 1.85	$\mathbf{27.74 \pm 3.45}$
500	26.51 ± 2.76	26.13 ± 2.74	32.06 ± 3.51	$\mathbf{36.79 \pm 2.73}$
1000	28.53 ± 4.17	28.87 ± 2.86	40.09 ± 3.52	$\mathbf{42.15 \pm 2.08}$
MNIST				
200	79.50 ± 5.00	80.37 ± 2.72	80.86 ± 4.09	$\mathbf{81.81 \pm 2.70}$
500	84.91 ± 3.61	85.06 ± 1.88	85.09 ± 5.61	$\mathbf{87.01 \pm 3.25}$
800	86.22 ± 2.29	85.64 ± 2.88	$\mathbf{89.47 \pm 1.94}$	89.33 ± 2.49

distance is selected as B_{opt}. Finally, as shown in line 11, the current model is updated on the union of B_n and B_{opt}, after which the memory buffer is refreshed.

Implementation details follow the same structure across architectures. In experiments with CIFAR-10 and a ResNet-18 backbone, embeddings are taken from the output of the third residual block after average pooling. For the two-layer MLP baseline, embeddings are extracted from the second hidden layer. This choice of layer acts as a tunable hyperparameter but is fixed per experiment to ensure consistent measurement of BCD.

5.2.2 Results

In Table 5.4, we present the results of our batch sampling method (RBS) compared to Experience Replay (ER), Class-balanced and Maximally Interfered Retrieval (MIR). Each result shown in the table is the average of 20 independent runs with the standard deviation mentioned. Class-balanced Experience Replay is a simple modification of naive ER, where the random sampling is stratified by classes. For the CIFAR-10 dataset, our method consistently outperformed both ER and MIR in all memory buffer sizes. In particular, with a smaller memory size, our method achieved a greater improvement, since the low variability of the available samples increases the impact of the selection strategy.

For the MNIST dataset, our method maintained its competitive performance. With a memory size of 200 and 500, our batch sampling approach achieved the highest average accuracies of 81.81% and 87.01%, respectively, outperforming ER and MIR. However, for a memory size of 800, MIR slightly surpassed our method with an average accuracy of 89.47, compared to 89.33 achieved by our method. The Wilcoxon signed rank test found that only for this memory size of all shown in Table 5.4 the MIR and BS results belong to the same distribution. These results empirically demonstrate the effectiveness of our batch sampling method in the

Replay-based Continual Learning setting.

5.2.3 Limitations and Future Work

Our research, while promising, is not without limitations. Here, we outline the potential areas where this approach may be constrained and could benefit from further investigation and refinement.

A significant area of focus is the trade-off between effectiveness, represented by *informativeness*, and *diversity* within a replay batch. Balancing the two aspects might be challenging, especially since the appropriate balance could vary between different tasks and even model updates.

Another concern pertains to the scope of diversity. Although our method excels at fostering diversity within individual replay batches, it does not inherently ensure diversity across a sequence of such batches. The absence of such cross-batch diversity could compromise the model's long-term performance when facing an extended sequence of tasks, warranting additional investigation into how to integrate inter-batch diversity.

The choice of metric also introduces another layer of complexity. We have used cosine similarity between hidden representations, largely due to its computational efficiency and its established role in assessing the similarity between embeddings. However, this metric, like any other, may not capture all the intricacies of the change in learned knowledge represented by the hidden layers.

Moreover, given the unpredictable dynamics of task transitions in continual learning, there may exist situations where catastrophic forgetting is so pronounced that virtually any replay batch would offer utility in mitigating it. In such scenarios, the computational overhead of an exhaustive search for an optimal batch using methods such as RBS becomes unnecessary and could be avoided. Further investigation of the appropriate time for application of the method could noticeably increase the training speed.

Finally, there is the question of benchmarking methods that utilize secondary memory buffers, which includes our proposed RBS. For a fair comparison, the additional memory utilized should be taken into account in the overall memory budget.

5.2.4 Conclusions

This chapter has explored how the use of a replay buffer is just as critical as its composition for effective continual learning. Our investigation began with a detailed analysis of replay–update protocols. We compared merging replay samples with the current minibatch in a single backward pass against processing them in a separate pass before the optimizer update. Through the extensive experiments across multiple datasets, architectures, and sampling heuristics, we showed that this seemingly minor implementation detail can substantially alter optimization dynamics, final accuracy, and even reverse the relative ranking of sample-selection strategies. This result underscores that experience-replay

methods cannot be fairly compared unless their update protocols are explicitly specified and carefully controlled.

Having established the importance of update protocols, we shifted our focus to the selection of samples from the memory buffer. Drawing inspiration from active-learning principles of informativeness and diversity, we proposed *Random Batch Sampling* (RBS), a method that evaluates a small number of candidate replay batches by computing the cosine distance between their hidden representations and those of a temporarily updated model. Despite the enormous combinatorial space of possible batches, we demonstrated that considering just a handful of candidates per update was sufficient to outperform both standard Experience Replay and the strong Maximally Interfered Retrieval baseline. Our experiments further showed that RBS provides the largest gains under tight memory constraints, where the quality of each replay batch is most consequential.

In summary, this chapter demonstrates that the success of experience replay is not determined solely by which samples fill the memory buffer. Equally decisive are the mechanisms by which those samples are chosen and integrated during training. Recognizing and explicitly managing these factors is essential for building continual-learning systems that learn stably and retain knowledge over long sequences of tasks.

Chapter 6

Memory Replay Evaluation

EXISTING metrics in continual learning primarily focus on the model's vulnerability to catastrophic forgetting, the process of performance decay on previously learned tasks due to the acquisition of new information, or its proficiency in knowledge transfer between tasks. However, they fall short of specifically evaluating the model's ability to preserve internal knowledge as opposed to accessing or, in the case of the memory-based CL methods, relearning it, often at the expense of losing generalization across tasks.

To address this need, we introduce Knowledge Retention (published in *Evaluating Knowledge Retention in Continual Learning* [42]), a metric designed to offer a model-, method-, and task-agnostic assessment of knowledge preservation in continual learning scenarios. We operate under the assumption that the proficiency of a model in extracting features from unseen data is indicative of both its generalization capabilities and the extent to which it retains knowledge from previous tasks. This effectiveness can be quantified by comparing the task-wise performance of the original model with a full copy that has its feature extraction layers frozen, both trained on the current task.

Our results show that KR successfully indicates whether a model is actually learning from a sequence of tasks, such as in the case of Experience Replay [77], or simply memorizing data from a memory buffer, using the Greedy Sampler and Dumb Learner (GDumb) [70] as an extreme example of such a strategy. Furthermore, it is capable of isolating the impact of stored samples, which typically influences the overall performance of the model, but it can now be evaluated independently from other variables.

6.1 Metric

To compute Knowledge Retention, we employ two versions of the same model architecture: one that is fully trainable (θ) and another where all layers except the classifier head are frozen (θ_{frozen}). We make a new θ_{frozen} on the beginning of each task by copying the current state of θ.

$$\mathrm{KR} = \frac{1}{N} \sum_{i=1}^{N} \left(A_i - A_i^{frozen} \right) \tag{6.1}$$

The KR metric measures the average difference in performance between θ and θ_{frozen} over all tasks. A lower KR value indicates better knowledge retention, as it suggests that the feature extractor of the model has learned features that generalize well across tasks.

6.2 Experiments

To demonstrate the application of the Knowledge Retention metric, we compare the memory-based methods with the opposite approaches to utilization of the memory buffer, the data structure maintaining the small subset of the experiences from the previous tasks. Experience Replay aims to mitigate catastrophic forgetting by retaining a memory buffer of past data. During training on the new tasks, the algorithm samples from this buffer to merge past and present experiences, allowing the model to replay and stabilize its knowledge. On the other hand, GDumb simply uses the memory buffer to retrain the model from scratch prior to evaluation. Despite the fact that the transfer of knowledge is impossible in this method, it appears to be a strong baseline, often outperforming the other CL methods. Even a detailed comparison of the training trajectory in the parameter space [88] between GDumb and ER cannot reveal any problems with this not quite continual approach. To calculate KR for GDumb, we first train θ_{frozen} based on θ and then create a new θ trained on the memory buffer from scratch. We expect that KR will show that GDumb tends to retrain on the memory noticeably more than ER, since that is the key difference between the methods.

We also modify the original ER, which randomly selects samples for the memory, to preserve the experiences sorted by confidence and accuracy. In **ER-High** the memory consists of the samples classified correctly with high certainty. **ER-Low** is focusing on high-confidence but incorrectly classified samples. Finally, **ER-New** includes only samples never seen by the model before. New strategies are used to investigate how different types of memory buffer influence the retention of existing knowledge and the acquisition of new information.

6.3 Results

According to our results in Table 6.1, the ER modifications behave as predicted in terms of a small isolated influence of the novelty of the stored in the memory samples on the internal knowledge of the model. The metrics are consistent across both datasets and all memory sizes, except for the ER-High Average Accuracy. The possible reason is the difference in the data complexity of the MNIST and CIFAR-10 datasets. For a harder task, it appears to be more beneficial to preserve simpler-to-learn samples, as they are more difficult to forget. That is

Table 6.1: The Average Accuracy, Knowledge Retention and Average Forgetting across all five tasks of the MNIST and CIFAR-10.

Method	Memory	$A_5(\%) \uparrow$	$KR \downarrow$	$F_5(\%) \downarrow$
MNIST				
ER	200	78.78 ± 2.49	2.25 ± 1.27	25.59 ± 3.09
	500	87.37 ± 1.59	3.14 ± 1.21	14.54 ± 1.8
	1000	91.43 ± 0.61	3.8 ± 0.99	9.35 ± 0.84
	2000	93.54 ± 1.31	**3.36 ± 0.83**	6.46 ± 1.71
ER - high	200	71.28 ± 3.89	2.82 ± 0.72	34.91 ± 5.01
	500	*75.82 ± 3.81*	3.45 ± 1.0	*29.27 ± 4.76*
	1000	*78.83 ± 0.79*	3.52 ± 0.43	*25.45 ± 0.92*
	2000	*81.51 ± 2.22*	4.43 ± 0.83	*21.64 ± 3.07*
ER - low	200	*69.18 ± 3.33*	**0.85 ± 1.03**	*37.47 ± 4.13*
	500	81.53 ± 2.47	3.62 ± 0.56	21.94 ± 3.16
	1000	87.83 ± 2.41	**2.72 ± 0.84**	13.86 ± 3.06
	2000	92.15 ± 0.93	3.63 ± 0.34	8.08 ± 1.03
ER - new	200	79.11 ± 1.97	2.38 ± 0.22	25.1 ± 2.51
	500	86.04 ± 1.97	**2.73 ± 0.5**	16.02 ± 2.89
	1000	89.89 ± 1.81	3.78 ± 0.73	10.89 ± 2.04
	2000	92.73 ± 0.99	3.97 ± 0.76	7.06 ± 1.43
GDumb	200	**81.94 ± 1.45**	*6.65 ± 0.92*	**9.06 ± 1.65**
	500	**88.14 ± 0.65**	*6.35 ± 0.59*	**5.87 ± 0.49**
	1000	**91.19 ± 0.33**	*5.53 ± 0.62*	**4.7 ± 0.59**
	2000	**93.54 ± 0.33**	*5.47 ± 0.52*	**3.22 ± 0.45**
CIFAR-10				
ER	200	*26.23 ± 1.93*	3.89 ± 2.58	53.16 ± 3.1
	500	33.17 ± 2.3	5.16 ± 2.72	42.32 ± 4.43
	1000	40.44 ± 4.37	5.74 ± 3.17	30.73 ± 4.96
	2000	44.67 ± 2.6	**4.97 ± 3.41**	23.07 ± 4.67
ER - high	200	**27.58 ± 1.96**	3.84 ± 2.26	51.34 ± 4.45
	500	**35.1 ± 3.57**	5.43 ± 2.04	36.9 ± 5.39
	1000	**40.79 ± 4.01**	5.54 ± 2.72	26.61 ± 6.37
	2000	**45.01 ± 3.17**	5.68 ± 2.84	**19.35 ± 8.01**
ER - low	200	21.78 ± 1.81	**3.18 ± 2.56**	*60.34 ± 4.88*
	500	31.14 ± 5.01	5.0 ± 3.07	*45.14 ± 6.81*
	1000	*37.29 ± 3.17*	5.4 ± 2.59	*35.01 ± 4.23*
	2000	43.15 ± 3.49	5.61 ± 2.29	*24.88 ± 4.68*
ER - new	200	27.27 ± 3.27	5.07 ± 2.68	55.23 ± 5.89
	500	33.81 ± 0.47	**4.88 ± 1.88**	39.47 ± 4.92
	1000	38.9 ± 3.12	**4.22 ± 2.74**	30.16 ± 2.18
	2000	*42.08 ± 2.16*	8.0 ± 0.86	21.52 ± 4.25
GDumb	200	26.28 ± 0.99	*7.56 ± 2.42*	**25.26 ± 1.54**
	500	*31.02 ± 0.94*	*10.44 ± 2.11*	**25.54 ± 1.45**
	1000	37.41 ± 0.75	*11.49 ± 2.23*	**23.09 ± 1.85**
	2000	43.01 ± 1.29	*13.26 ± 2.12*	22.59 ± 0.95

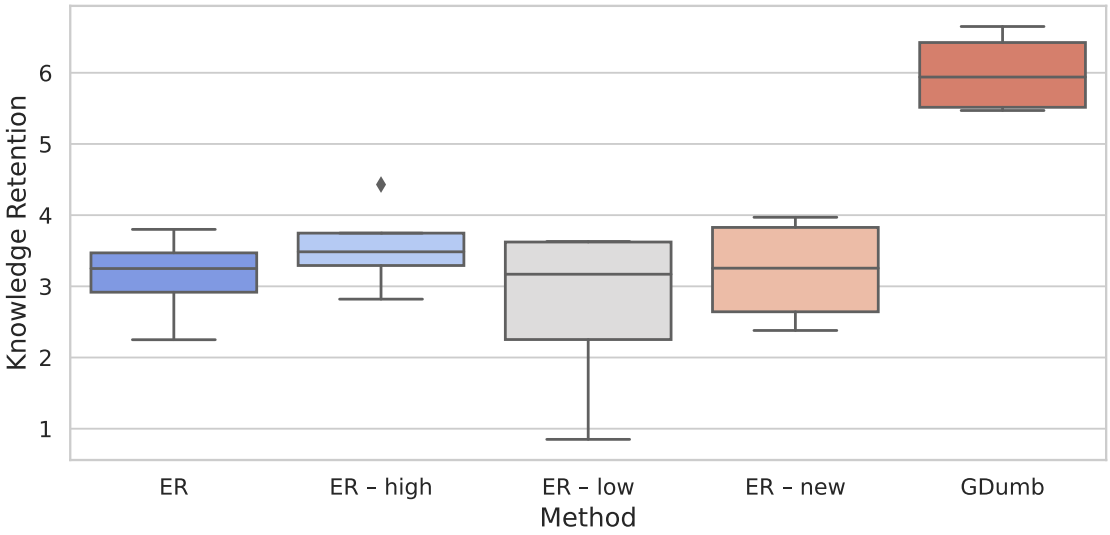


Figure 6.1: Boxplot illustrating the statistical dispersion of the Knowledge Retention for MNIST dataset across ER and GDumb methods with memory sizes 200, 500, 1000 and 2000.

confirmed by the Forgetting metric, which shows the lowest values for ER-High in all CIFAR-10 results except for the GDumb.

But the main result is clearly anomalous KR values for all GDumb experiments (Figures 6.1 and 6.2), which expose the problem of knowledge retention from previous tasks.

6.4 Limitations

Knowledge Retention metric is effective for its intended purpose of evaluating the balance of the model between retaining prior knowledge and adapting to new tasks. However, it requires training a classifier head for an additional model. The cost of training is substantially reduced by exposing the same training batches to both models, eliminating the need for separate data processing.

The next issue is the interpretation of the metric, which is possible only by comparison with the baseline. Relying on the memory is not harmful and rather expected behavior of a memory-based methods, but only if it is not preventing further learning and generalization. Experience Replay is a perfectly balanced example that shows good results in this role, but GDumb may also serve as a point of reference.

6.5 Conclusion

Knowledge Retention provides a direct measure of how well a continual learning model preserves internal representations while acquiring new tasks. Experiments show that ER maintains low KR values, indicating stable feature extraction,

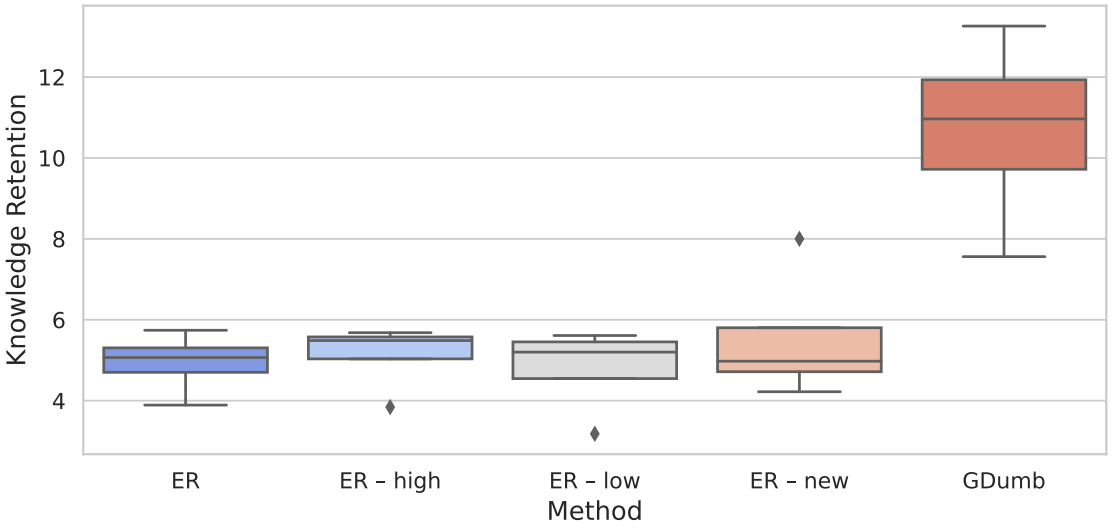


Figure 6.2: Boxplot illustrating the statistical dispersion of the Knowledge Retention for CIFAR-10 dataset across ER and GDumb methods with memory sizes 200, 500, 1000 and 2000.

whereas GDumb consistently yields high KR, confirming its reliance on memory retraining rather than true retention. Variants of ER reveal how buffer composition influences forgetting without compromising generalization. Although KR requires an auxiliary frozen model and baseline comparison, it isolates the effect of stored samples and exposes hidden weaknesses in memory-based methods. This makes KR a practical metric for evaluating knowledge preservation across diverse continual learning approaches.

Chapter 7

Conclusions and Future Work

In this dissertation I explored continual learning through the lens of replay—how memories are constructed, how they guide optimization, and how their impact can be measured. My goal was to move beyond buffer size as the dominant concern and to show that replay is a sequence of interlocking design choices about what to remember and how to use it.

I contributed three algorithms along a spectrum from explicit to implicit memory. **Deep Features Buffer (DFB)** improves explicit replay by keeping examples that are diverse in the model's feature space, improving accuracy and reducing forgetting without larger buffers. **HebbCL** removes the buffer altogether, storing knowledge directly in sparse, frozen weights updated by local Hebbian rules. **Continual Visual Mapping (CVM)** goes a step further, training a compact visual encoder to align with fixed semantic embeddings from a frozen language model, achieving state-of-the-art results on CIFAR-100, Tiny-ImageNet, and CORe50 without replay or parameter growth. Together these methods trace a path from carefully managed buffers to purely representational strategies.

I also showed that low-level training details strongly affect performance. A simple implementation choice—merging versus separating replay samples from the current batch—changed accuracy enough to reverse the ranking of established heuristics. At the sampling level, I introduced **Random Batch Sampling**, which scores entire candidate batches by Batch Cosine Distance and consistently outperforms strong single-example heuristics like MIR, especially when memory is tight.

From these studies I draw three main lessons. First, replay effectiveness depends more on representation geometry and optimization dynamics than on raw buffer size. Second, diversity at both the sample and batch levels is critical for stable long-term learning. Third, metrics matter: I proposed the **Knowledge Retention** measure to separate genuine representational stability from simple relearning, giving a clearer view of long-term memory.

This work has limits. I focused on supervised class-incremental learning with CNN-based models. HebbCL has so far been tested only on shallow networks, and CVM depends on static language-model embeddings whose behavior in other modalities is untested. I did not address generative replay, parameter-isolation methods, or large transformer architectures.

Looking ahead, several promising directions emerge. Scaling HebbCL to deeper networks and higher-resolution data while keeping its interpretability would test its robustness. Extending CVM to multimodal continual learning and exploring adaptive or evolving semantic anchors could broaden its applicability. Developing adaptive replay protocols that choose between merged and separated updates based on gradient statistics may further reduce forgetting. Combining representation-aware memory construction with batch-level selection could yield unified replay strategies. Finally, testing these ideas in self-supervised, reinforcement, and highly non-stationary settings would provide stronger evidence of generality.

Replay remains the most practical tool for mitigating catastrophic forgetting, but my results show that its power lies in how memory, representation, and optimization interact. By treating these as connected design choices and by exploring both explicit and implicit memory, I provide algorithms and methodological guidance for building continual learners that are more robust, interpretable, and scalable.

Bibliography

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019.

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[3] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, and Gabriele Lagani. Hebbian learning meets deep convolutional neural networks. In Elisa Ricci, Samuel Rota Bulò, Cees Snoek, Oswald Lanz, Stefano Messelodi, and Nicu Sebe, editors, *Image Analysis and Processing - ICIAP 2019 - 20th International Conference, Trento, Italy, September 9-13, 2019, Proceedings, Part I*, volume 11751 of *Lecture Notes in Computer Science*, pages 324–334. Springer, 2019.

[4] Jihwan Bang, Heesu Kim, Young Joon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8214–8223, 2021.

[5] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010. doi: 10.1007/s10994-009-5152-4.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009. doi: 10.1145/1553374.1553380.

[7] Elie L Bienenstock, Leon N Cooper, and Paul W Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48, 1982.

[8] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (5):5497–5512, May 2023. ISSN 1939-3539. doi: 10.1109/tpami.2022.3206549. URL http://dx.doi.org/10.1109/TPAMI.2022.3206549.

[9] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[10] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning, 2020. URL https://arxiv.org/abs/2010.05595.

[11] 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. URL http://jmlr.org/papers/v24/23-0130.html.

[12] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

[13] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co^{2}l: Contrastive continual learning, 2021. URL https://arxiv.org/abs/2106.14413.

[14] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[15] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Tiny episodic memories in continual learning. In *arXiv preprint arXiv:1902.10486*, 2019.

[16] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[17] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *ICML*, 2020.

[18] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

[19] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for Continual Learning, October 2018. URL http://arxiv.org/abs/1810.13166. arXiv:1810.13166 [cs].

[20] Adrian Englhardt, Holger Trittenbach, Dennis Vetter, and Klemens Böhm. Finding the sweet spot: Batch selection for one-class active learning. In *SDM*, 2020.

[21] Andrea Ferigo, Elia Cunegatti, and Giovanni Iacca. Neuron-centric hebbian learning, 2024. URL https://arxiv.org/abs/2403.12076.

[22] Enrico Fini, Victor G. Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners, 2022. URL https://arxiv.org/abs/2112.04215.

[23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017.

[24] Péter Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64(2):165–170, 1990.

[25] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[26] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987.

[27] Md Yousuf Harun, Jhair Gallardo, Tyler L. Hayes, Ronald Kemker, and Christopher Kanan. Siesta: Efficient online continual learning with sleep, 2023. URL https://arxiv.org/abs/2303.10725.

[28] Md Yousuf Harun, Jhair Gallardo, Junyu Chen, and Christopher Kanan. Grasp: A rehearsal policy for efficient online continual learning, 2024. URL https://arxiv.org/abs/2308.13646.

[29] Tyler L. Hayes, Ronald Kemker, Nathan D. Cahill, and Christopher Kanan. New Metrics and Experimental Paradigms for Continual Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2112–2123, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00273. URL https://ieeexplore.ieee.org/document/8575441/.

[30] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 466–483, 2020.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[32] Donald O Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

[33] Jeremy Howard. imagenette. URL https://github.com/fastai/imagenette/.

[34] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the 38th International Conference on Machine Learning*, PMLR, 2021.

[35] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, volume 12358 of *Lecture Notes in Computer Science*, pages 411–428. Springer, 2020. doi: 10.1007/978-3-030-58601-0_25.

[36] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, et al. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the National Academy of Sciences*, volume 114, pages 3521–3526, 2017.

[37] Teuvo Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, 100(4):353–359, 1972.

[38] Alexander Krawczyk and Alexander Gepperth. An analysis of best-practice strategies for replay and rehearsal in continual learning. pages 4196–4204, 2024. doi: 10.1109/CVPRW63382.2024.00423.

[39] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[40] Dmitry Krotov and John J Hopfield. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, 116(16):7723–7731, 2019.

[41] Andrii Krutsylo. Batch sampling for experience replay. In *Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD)*, pages 202–206, 2024.

[42] Andrii Krutsylo. Evaluating knowledge retention in continual learning. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, pages 1053–1055, 2024.

[43] Andrii Krutsylo. The inter-batch diversity of samples in experience replay for continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23395–23396, 2024.

[44] Andrii Krutsylo. The hebbian forward-forward algorithm. In *Proceedings of the NeurIPS 2025 Workshop on Optimization for Machine Learning (OptML)*, 2025. URL https://opt-ml.org/. Non-archival workshop paper.

[45] Andrii Krutsylo. Merging versus separating replay samples in continual learning. In *International Conference on Artificial Neural Networks*, pages 600–608. Springer, 2025.

[46] Andrii Krutsylo. Non-uniform memory sampling in experience replay, 2025. URL https://arxiv.org/abs/2502.11305.

[47] Andrii Krutsylo. Scalable forward-forward algorithm, 2025. URL https://arxiv.org/abs/2501.03176.

[48] Andrii Krutsylo and Paweł Morawiecki. Diverse memory for experience replay in continual learning. *ESANN 2022 proceedings*, 2022. URL https://www.esann.org/sites/default/files/proceedings/2022/ES2022-83.pdf.

[49] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. In *Science*, volume 350, pages 1332–1338, 2015.

[50] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998.

[51] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*, 2020.

[52] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven C.H. Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In *Advances in Neural Information Processing Systems*, 2021.

[53] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C.H. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *Advances in Neural Information Processing Systems*, 2022.

[54] Junnan Li, Yixuan Fang, Haoxuan Yang, Pengchuan Li, Xiyang Wang, Lei Zhang, Pengliang Liu, Zhengyuan Zhang, Michael Zeng, and Steven C.H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, PMLR, 2023.

[55] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 614–629, 2017.

[56] Yuchen Liang, Chaitanya Ryali, Benjamin Hoover, Leopold Grinberg, Saket Navlakha, Mohammed J Zaki, and Dmitry Krotov. Can a fruit fly learn word embeddings? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xfmSoxdxFCG.

[57] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017.

[58] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[59] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[60] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

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

[62] Jan Melchior and Laurenz Wiskott. Hebbian-descent, 2019. URL https://arxiv.org/abs/1905.10585.

[63] Nicolas Michel, Giovanni Chierchia, Romain Negrel, Jean-François Bercher, and Toshihiko Yamasaki. New metrics for analyzing continual learners, September 2023. URL http://arxiv.org/abs/2309.00462. arXiv:2309.00462 [cs].

[64] Paweł Morawiecki, Andrii Krutsylo, Maciej Wołczyk, and Marek Śmieja. Hebbian continual representation learning, 2022. URL https://arxiv.org/abs/2207.04874.

[65] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. Technical report, Stanford University, 2011.

[66] Julian Jimenez Nimmo and Esther Mondragon. Advancing the biological plausibility and efficacy of hebbian convolutional neural networks, 2025. URL https://arxiv.org/abs/2501.17266.

[67] Barza Nisar, Hruday Vishal Kanna Anand, and Steven L. Waslander. Gradient-based maximally interfered retrieval for domain incremental 3d object detection, 2023. URL https://arxiv.org/abs/2304.14460.

[68] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[69] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[70] Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In Andrea

Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, volume 12347, pages 524–540. Springer International Publishing, Cham, 2020. ISBN 978-3-030-58535-8 978-3-030-58536-5. doi: 10.1007/978-3-030-58536-5_31. URL https://link.springer.com/10.1007/978-3-030-58536-5_31. Series Title: Lecture Notes in Computer Science.

[71] Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip H.S. Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally Budgeted Continual Learning: What Does Matter? In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3698–3707, Vancouver, BC, Canada, June 2023. IEEE. ISBN 9798350301298. doi: 10.1109/CVPR52729.2023.00360. URL https://ieeexplore.ieee.org/document/10204648/.

[72] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *PMLR*, pages 8748–8763, 2021.

[73] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[74] Clea Rebillard, Julio Hurtado, Andrii Krutsylo, Lucia Passaro, and Vincenzo Lomonaco. Continually learn to map visual concepts to language models in resource-constrained environments. *Neurocomputing*, page 131013, 2025.

[75] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[76] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[77] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience Replay for Continual Learning, November 2019. URL http://arxiv.org/abs/1811.11682. arXiv:1811.11682 [cs, stat].

[78] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, et al. Progressive neural networks. In *arXiv preprint arXiv:1606.04671*, 2016.

[79] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[80] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2015.

[81] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[82] J. Schwarz, Siddhant M. Jayakumar, Razvan Pascanu, Peter E. Latham, and Yee Whye Teh. Powerpropagation: A sparsity inducing weight reparameterisation. *CoRR*, abs/2110.00296, 2021. URL https://arxiv.org/abs/2110.00296.

[83] Terrence J. Sejnowski. Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, 4(4):303–321, 1977.

[84] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison, 2009.

[85] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012. doi: 10.1561/2200000018.

[86] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2994–3003, 2017.

[87] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. April 2019. URL http://arxiv.org/abs/1904.07734. arXiv:1904.07734 [cs, stat].

[88] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9264–9273, 2021.

[89] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24214–24223, 2023.

[90] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 1985.

[91] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022.

[92] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[93] Maciej Wołczyk and Andrii Krutsylo. Remember more by recalling less: Investigating the role of batch size in continual learning with experience replay (student abstract). In *AAAI*, 2021.

[94] Hongye Xu, Jan Wasilewski, and Bartosz Krawczyk. Balanced gradient sample retrieval for enhanced knowledge retention in proxy-based continual learning, 2024. URL https://arxiv.org/abs/2412.14430.

[95] Qihan Yang, Fan Feng, and Rosa Chan. A benchmark and empirical analysis for replay strategies in continual learning, 2022. URL https://arxiv.org/abs/2208.02660.

[96] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995, 2017.

[97] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022.

Appendix A

Estimating the Upper Bound of Replay Sampling Performance in Continual Learning

We revisit the assumption that uniform replay is the default optimal strategy by introducing a controlled framework that isolates only the effect of sampling. Within this setup, we evaluate multiple randomly drawn non-uniform distributions using a fixed replay buffer. Some of these distributions outperform uniform replay, though only when identified post hoc and without forming a usable policy. This controlled comparison therefore provides an empirical *upper bound* on the gains achievable through non-uniform sampling, showing that uniform replay does not sit at the observed performance frontier. The results indicate clear headroom for improvement and motivate the design of adaptive or learned replay strategies, rather than a claim of superiority for any particular random distribution.

A.1 Experiments

Training Setup. All models are trained using the Adam optimizer, with a learning rate of 0.001 for the CNN and 0.0003 for MobileNet. The minibatch sizes for both the online stream and replay are set to 32. The memory buffer size is varied across experiments: $200, 500, 1000, 2000$. Each experiment is repeated 20 times for CIFAR-10 (seeds 0-19) and 5 times for Imagenette (seeds 0-4), using different random seeds for each run. Imagenette is a subset of 10 easily classified classes from ImageNet [33].

To ensure a consistent buffer across sampling conditions, we use class-balanced reservoir sampling and fix the buffer update process by using a dedicated seed per trial. This guarantees that the memory contents and their evolution are identical across all 51 trials (50 non-uniform, 1 uniform). For each non-uniform trial, we generate a sampling probability vector w_i by drawing weights independently from a uniform distribution over $[0, 1)$ and then normalizing to form a probability distribution: $p_i = w_i / \sum_j w_j$. In the uniform baseline,

each memory element is sampled with equal probability: $p_i = 1/|\mathcal{M}|$. The number of non-uniform sampling distributions (50) was chosen as a practical compromise between statistical power and computational cost.

Hyperparameter Tuning. All hyperparameters were tuned only for the uniform replay baseline. Learning rates were searched logarithmically between 0.1 and 0.00001. For DER++, the distillation coefficient α and the supervised loss coefficient β were tuned over $0.1, 0.3, 0.5, 0.7, 1.0$. For GSS-Greedy, we searched the memory strength hyperparameter in the range of 2 to 32.

We use the Avalanche [11] implementation of the training data processing, memory buffer management, experience replay, and specific replay strategies such as MIR, GSS, and DER++ methods.

A.2 Results

We evaluate performance on CIFAR-10 and Imagenette under three sampling strategies (uniform, MIR and non-uniform), and report final test accuracy over multiple seeds. For CIFAR-10, each result reflects the mean and standard deviation over 20 seeds, for Imagenette over 5 seeds.

Tables A.1 to A.3 show the results. For each method, we report the best-performing non-uniform sampling distribution selected *post hoc* from 50 random candidates. This approach allows us to estimate the potential upper bound of performance gain when exploring the space of sampling strategies. Statistical controls are discussed in Statistical Significance section.

Table A.1: Final average accuracy (%) on CIFAR-10 using the ER and GSS methods, comparing uniform sampling vs. the best non-uniform random distribution out of 50 tested. Each entry shows the mean and standard deviation over 20 random seeds.

	ER			GSS	
Size	**Uniform**	**MIR**	**Non-uniform**	**Uniform**	**Non-uniform**
200	40.41 ± 2.20	39.92 ± 2.39	42.12 ± 1.51	37.03 ± 2.08	39.95 ± 1.09
500	43.35 ± 2.00	43.78 ± 1.80	46.39 ± 1.85	40.45 ± 1.72	44.14 ± 1.14
1000	44.85 ± 2.47	44.42 ± 2.90	47.76 ± 1.23	41.90 ± 1.64	43.60 ± 0.76
2000	44.26 ± 3.63	45.02 ± 3.18	48.84 ± 1.75	42.15 ± 2.55	42.80 ± 0.52

Across all memory sizes, selecting the best-performing non-uniform distribution *after* training provides a clear empirical upper bound on what replay sampling can achieve with a fixed buffer. For example, on CIFAR-10 with buffer size 200, uniform replay and MIR yield $40.41\% \pm 2.20\%$ and $39.92\% \pm 2.39\%$, while the strongest post-hoc non-uniform distribution reaches $42.12\% \pm 1.51\%$. Similar gaps appear at every buffer size and across both datasets, indicating that uniform replay does not lie on the observed performance frontier.

Table A.2: Final average accuracy (%) on CIFAR-10 using the DER++ method, comparing uniform sampling vs. the best non-uniform random distribution out of 50 tested. Each entry shows the mean and standard deviation over 20 random seeds.

	DER++	
Size	**Uniform**	**Non-uniform**
200	42.41 ± 2.23	43.95 ± 2.17
500	43.89 ± 3.18	48.32 ± 1.92
1000	46.02 ± 2.54	49.73 ± 2.11
2000	46.40 ± 2.67	49.89 ± 1.57

Table A.3: Final average accuracy (%) on Imagenette, comparing uniform sampling vs. the best non-uniform random distribution out of 50 tested. Each entry shows the mean and standard deviation over 5 random seeds.

	ER		
Size	**Uniform**	**MIR**	**Non-uniform**
200	64.07 ± 4.59	61.13 ± 2.84	69.08 ± 3.47
500	70.79 ± 2.95	68.27 ± 2.49	77.43 ± 2.12
1000	76.29 ± 2.25	73.08 ± 2.35	80.32 ± 1.46
2000	80.01 ± 2.27	77.40 ± 3.59	81.32 ± 1.07

The same pattern holds for GSS, where absolute accuracy is lower than ER but the best non-uniform distribution still provides consistent gains. At buffer size 200, accuracy rises from $37.03\% \pm 2.08\%$ to $39.95\% \pm 1.09\%$, showing that even gradient-based selection leaves measurable headroom.

DER++ shows the largest margin. With buffer size 500, the top non-uniform trial improves accuracy from $43.89\% \pm 3.18\%$ to $48.32\% \pm 1.92\%$, illustrating the potential benefit of more selective sampling when distillation is used.

Imagenette results (Table A.3) mirror these findings. At buffer size 500, the best non-uniform distribution reaches $77.43\% \pm 2.12\%$, compared to $70.79\% \pm 2.95\%$ for uniform and $68.27\% \pm 2.49\%$ for MIR.

Although MIR is designed to reduce interference, it consistently underperforms both the uniform baseline and the empirical upper bound from random non-uniform trials. One possible reason is that we are merging the current task and the memory minibatches, then performing one optimization step on them. This is not the default setting for MIR, which was developed for a two-step setting.

These results should be interpreted as evidence of achievable performance *upper bound*, not as identification of a deployable non-uniform policy.

A.3 Statistical Significance

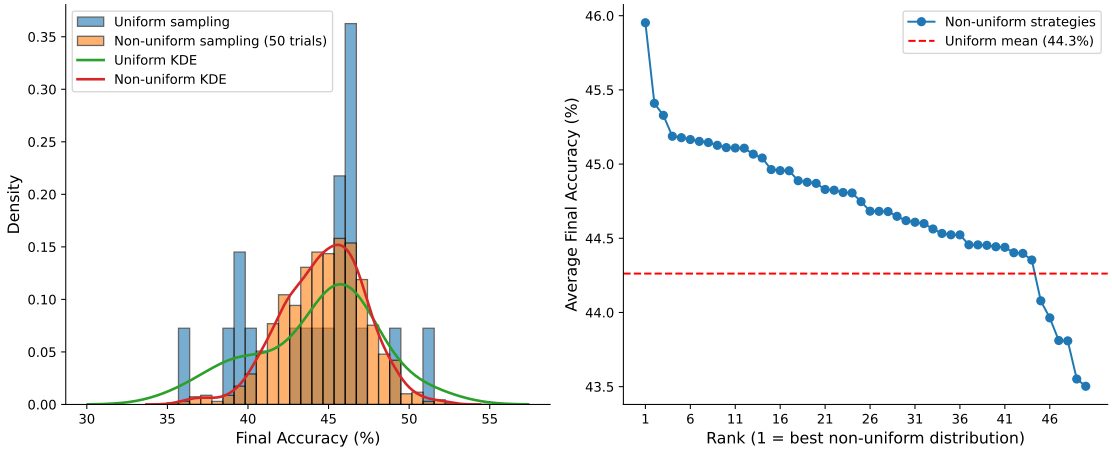


Figure A.1: (Left) Distribution of final test accuracies (%) across 20 independent seeds using uniform sampling (blue) and a large pool of 1000 non-uniform sampling trials (orange, 50 strategies × 20 seeds). Both histogram and KDE estimates are shown for clarity. The distribution for non-uniform sampling is shifted toward higher accuracies, indicating consistently better performance. (Right) Rank–performance plot of the 50 distinct non-uniform sampling strategies, averaged across all seeds. Strategies are sorted in descending order of mean final accuracy. The red dashed line denotes the mean accuracy of uniform sampling.

Multiple Comparisons Correction: To test whether the observed performance differences are statistically meaningful, we compared each of the 50 non-uniform strategies to uniform sampling using paired t-tests across 20 seeds (CIFAR-10 with memory size 1000). Since selecting the best-performing strategy post hoc introduces a risk of false positives, we applied Bonferroni and Holm–Bonferroni corrections. For the top-performing strategy, the uncorrected p-value was 0.0869. After correction, Bonferroni yielded a p-value of 1.0, and Holm–Bonferroni yielded 0.9252, indicating that no individual strategy achieves statistical significance under standard thresholds when accounting for multiple comparisons.

Permutation Test: To assess overall effect direction without assuming normality, we conducted a paired, one-sided permutation test. For each seed, we computed the accuracy difference between the best-performing non-uniform strategy (selected post hoc) and the uniform baseline. The average difference was 4.58%. To build a null distribution, we randomly flipped the sign of each paired difference across 10,000 permutations. None of the permutations yielded a mean difference greater than or equal to the observed one, resulting in a p-value < 0.0001. This result provides robust, non-parametric evidence that non-uniform sampling can systematically outperform uniform sampling, even if no individual fixed strategy is statistically dominant.

Figure A.1 displays the performance distribution comparison where overlaid histograms and corresponding kernel density estimates illustrate a shift towards higher accuracies for the aggregated non-uniform strategies compared to uniform sampling. The rank–performance plot shows the sorted average performance of the 50 non-uniform strategies along with a horizontal line representing the uniform mean, which is consistently outperformed by the majority of non-uniform strategies.

Although no individual non-uniform distribution remains significant after correcting for multiple comparisons, the permutation test validates the estimated upper bound. The best-performing random distribution shows a mean accuracy gain of 4.58%, and this improvement is highly unlikely to arise by chance ($p<0.001$). Together, these findings confirm that the observed gap represents a genuine performance ceiling rather than a statistical artifact. Thus, while no specific random strategy can be declared inherently superior, the analysis provides robust evidence that the space of possible sampling rules contains distributions capable of exceeding the uniform baseline.

A.4 Conclusion

This study establishes an empirical upper bound on the benefit of alternative replay sampling when the memory buffer is held fixed. By evaluating randomly drawn non-uniform distributions, we showed that several exceed the performance of uniform replay, even though none is a deployable policy. Permutation testing confirms that these gains are unlikely to be due to chance, demonstrating that the observed gap reflects a true performance ceiling rather than noise. Because this ceiling is reached without any adaptive mechanism, it highlights untapped potential in the choice of replay sampling. These findings shift the discussion from proving the superiority of a particular heuristic to quantifying the room for improvement that principled or learned strategies might exploit. Future work can build on this framework to design adaptive or meta-learned policies that systematically approach or surpass the upper bound identified here.