UNIVERSITÀ CAMPUS BIO-MEDICO DI ROMA

DEPARTMENT OF ENGINEERING

UNIVERSITY OF CATANIA

DEPARTMENT OF ELECTRICAL, ELECTRONIC AND COMPUTER ENGINEERING

Italian National Ph.D. in Artificial Intelligence
Health and Life Sciences

XXXVIII Cycle

From Human Cognition to Continual Learning: Neural Selectivity, Memory Consolidation and Dreaming

Supervisors

Prof. Simone Palazzo

Prof. Concetto Spampinato

Candidate

Amelia Sorrenti

Abstract

Continual learning addresses the problem of incrementally acquiring knowledge from a sequence of tasks or a non-stationary data stream, without retraining from scratch. Despite a wide range of proposed strategies, neural networks trained sequentially often suffer from *catastrophic forgetting*, i.e., the tendency to lose performance on previously learned tasks after learning new ones. On the contrary, humans have a remarkable ability to learn continuously, retaining past experiences while quickly adapting to new tasks and problems. This gap between artificial and biological learning can be attributed to the inherent structure and optimization approaches of neural networks, which differ significantly from the way humans learn and build neural connectivity over a lifetime.

Motivated by this discrepancy, this dissertation investigates biologically inspired mechanisms for continual learning, leveraging neuroscientific insights as design principles to improve retention while facilitating future adaptation. Architectural inductive biases inspired by neural selectivity and pattern separation are investigated to promote more compartmentalized representations that reduce interference between tasks. Drawing inspiration from synaptic consolidation, strategies that selectively constrain updates to parts of the network are explored to support rapid adaptation while preserving previously acquired knowledge. This principle is then developed within a wake-sleep cycle grounded in Complementary Learning Systems (CLS) theory, where online learning is complemented by offline replay and multi-timescale memory organization. Within this view, replay and dreaming are conceived as functional components that support the restructuring of knowledge, which in turn lays the groundwork for future learning. This perspective is further developed through an approach in which dreams are generated directly from learned representations, enabling the autonomous construction of more adaptive representations.

Across standard benchmarks, the experimental results demonstrate that integrating neuroscience-inspired concepts into continual learning methods improves performance, strengthening both the preservation of past knowledge and the ability to learn from new tasks.

Contents

1 Introduction **16**

1.1 Experience requires time . . . 17
1.2 Time erodes experience . . . 18
1.3 Injecting brain mechanisms into continual learning . . . 20

2 Foundations of Continual Learning **23**

2.1 Problem formulation . . . 24
2.2 Scenarios . . . 24
2.2.1 Task-incremental learning . . . 25
2.2.2 Domain-incremental learning . . . 25
2.2.3 Class-incremental learning . . . 26
2.3 Benchmarks . . . 27
2.4 Evaluation metrics . . . 29
2.5 State of the Art . . . 31
2.5.1 Regularization-based methods . . . 32
2.5.2 Architectural methods . . . 34
2.5.3 Replay-based methods . . . 35
2.5.4 PEFT-based methods . . . 38

3 Architectural Inductive Bias **40**

3.1 Related work . . . 42
3.2 Method . . . 43
3.2.1 B-cos networks . . . 44
3.2.2 B-cosified architectures . . . 46
3.3 Experimental results . . . 46
3.3.1 Datasets and metrics . . . 46
3.3.2 Baselines . . . 47
3.3.3 Training procedure . . . 47

3.3.4 Results . . . 47
3.3.5 Generalization analysis . . . 50
3.4 Interpretability . . . 52
3.4.1 Explanations . . . 52
3.4.2 Qualitative results . . . 53
3.4.3 Quantitative results . . . 54
3.5 Discussion . . . 57
3.6 Publications . . . 58

4 Selective Freezing for Efficient Continual Learning 59
4.1 Related work . . . 60
4.2 Method . . . 61
4.2.1 Selective freezing . . . 61
4.3 Experimental results . . . 64
4.3.1 Datasets and metrics . . . 64
4.3.2 Baselines . . . 64
4.3.3 Training procedure . . . 64
4.3.4 Static freezing . . . 65
4.3.5 Selective freezing . . . 69
4.4 Discussion . . . 73
4.5 Publications . . . 73

5 Wake-Sleep Consolidated Learning 74
5.1 Related work . . . 75
5.2 Method . . . 77
5.2.1 Wake phase . . . 78
5.2.2 Sleep phase . . . 80
5.3 Experimental results . . . 81
5.3.1 Datasets and metrics . . . 81
5.3.2 Training procedure . . . 82
5.3.3 Results . . . 83
5.3.4 Ablation study . . . 86
5.4 Discussion . . . 91
5.5 Publications . . . 92

6 Dream2Learn: Structured Generative Dreaming 93
6.1 Related work . . . 94

6.2 Method . . . 95
6.2.1 Dreaming optimization and mapping . . . 97
6.2.2 Oracle-guided dreaming optimization . . . 100
6.3 Experimental results . . . 101
6.3.1 Datasets and metrics . . . 101
6.3.2 Training procedure . . . 102
6.3.3 Results . . . 102
6.3.4 Ablation study . . . 104
6.3.5 Comparison with Generative Replay . . . 106
6.4 Discussion . . . 107
6.5 Publications . . . 108

7 Continual Object 6D Pose Estimation 109
7.1 Related work . . . 111
7.2 Method . . . 112
7.2.1 Object pose estimation . . . 113
7.2.2 Memory-based continual learning . . . 113
7.2.3 Optimization . . . 115
7.3 Experimental results . . . 116
7.3.1 Datasets and metrics . . . 116
7.3.2 Baselines . . . 117
7.3.3 Training procedure . . . 118
7.3.4 Results . . . 119
7.3.5 Ablation study . . . 121
7.3.6 Object grasping . . . 122
7.4 Discussion . . . 123
7.5 Publications . . . 124

8 Personalized Mental Well-being Monitoring 125
8.1 Related work . . . 126
8.2 Method . . . 129
8.2.1 Buffer-based approach . . . 130
8.2.2 Prompt-based approach . . . 130
8.3 Experimental results . . . 131
8.3.1 Datasets and metrics . . . 131
8.3.2 Architectures . . . 133
8.3.3 Training procedure . . . 134

8.3.4 Results 135
8.4 Discussion 138
8.5 Publications 138

Conclusions **139**

List of Figures

2.1 **Task-incremental learning (Task-IL) scenario.** Each task i introduces a distinct set of classes $\mathcal{Y}_i$. At inference time, since task identity is available, the model is required to predict only among the classes associated with the task to which the input sample belongs. 25

2.2 **Domain-incremental learning (Domain-IL) scenario.** Each task i exposes the model to data from a different domain while sharing the same label space $\mathcal{Y}$. As task identity is not available at inference, the model must predict among all classes encountered so far, despite domain shifts that may substantially alter the visual appearance of the inputs. 26

2.3 **Class-incremental learning (Class-IL) scenario.** Each task i introduces a distinct subset of classes $\mathcal{Y}_i$. At inference time, since task identity is not available, the model must discriminate among all classes encountered so far. 27

3.1 **Visualization of the classification tasks for different values of B.** For $B = 1$ (first column), the neuron behaves as a standard linear unit, activating over the positive semi-hyperplane defined by its weight vector. Assuming the yellow cluster is the same in both tasks, the optimal separating direction differs substantially across tasks (compare rows within the same column). As B increases, the neuron's activation hypercone narrows around its weight vector, making activations more selective and the resulting decision regions more consistent across tasks (last column), reducing task-dependent representational interference. 45

3.2 **Explanation degradation across tasks.** Comparison of Grad-CAM maps for ER, DER++, and ER-ACE (top) and B-cos explanations for different values of B (bottom) over the task sequence. Red borders indicate instances in which the model produces an incorrect prediction. 54

3.3 **Explanation degradation across tasks using images from the first task.** Comparison of Grad-CAM maps for ER-ACE and B-cos explanations for $B = 2$ over the task sequence. Red borders indicate instances in which the model produces an incorrect prediction. 56
3.4 **Explanation degradation across tasks using images from later tasks in the sequence.** Comparison of Grad-CAM maps for ER-ACE and B-cos explanations for $B = 2$ over the task sequence. Red borders indicate instances in which the model produces an incorrect prediction, while orange borders indicate instances for which the model has not been trained on the corresponding task yet. 57

4.1 **ResNet-18 architecture and selective freezing strategy**. After an initial convolutional layer (apart from the last fully-connected layer), each colored portion represents a network's *main block*, consisting of two residual *basic blocks*, each applying two convolutions. For our selective freezing strategy, we treat each basic block as the smallest unit of freezing, named *layer*. 62
4.2 **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the DER++ method and a buffer size of 200.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen. 67
4.3 **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the DER++ method and a buffer size of 500.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen. 68
4.4 **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the ER-ACE method and a buffer size of 200.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen. 68
4.5 **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the ER-ACE method and a buffer size of 500.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen. 69

4.6 **Most frequent automatically learned freezing scheme for DER++ on Split CIFAR-10.** Results are computed over 10 runs. Values within each bar segment indicate the number of parameters. 71
4.7 **Total number of parameter updates for DER++ with and without selective freezing.** The number of updates is computed over the complete training for different numbers of epochs. Values above the vanilla DER++ bars report the relative change (%) with respect to the selective freezing strategy, where positive values denote more updates. 71
4.8 **Most frequent automatically learned freezing scheme for ER-ACE on Split CIFAR-10.** Results are computed over 10 runs. Values within each bar segment indicate the number of parameters. 72
4.9 **Total number of parameter updates for ER-ACE with and without selective freezing.** The number of updates is computed over the complete training for different numbers of epochs. Values above the vanilla ER-ACE bars report the relative change (%) with respect to the selective freezing strategy, where positive values denote more updates. 72

5.1 **Wake-Sleep Consolidated Learning.** In the wake stage, the model (which emulates the neocortex) fast adapts to the new sensory experience, storing episodic memories (as in the hippocampus) in the short-term memory to be replayed during sleep. The sleep phase foresees two alternating processes: 1) the NREM stage, where the DNN model consolidates its synapses based on the replayed (recent and past) samples and the long-term memory is updated; 2) the REM stage, where the DNN is trained with dreamed samples to prepare the model for future sensory inputs. 77
5.2 **Impact of dreaming quality**, in terms of noise (left) and image resolution (right). Results refer to ER-ACE and DER++ with WSCL (solid lines) and without it (dotted line). 87
5.3 **Impact of dreaming dataset dimension**. Results refer to ER-ACE and DER++ with WSCL (solid lines) and without it (dotted line). 89
5.4 **WSCL model efficiency**: Left: the most frequent automatically learned freezing scheme (values within bars are number of parameters) during the wake phase for ER-ACE on Tiny-ImageNet$_{1/2}$. Right: number of parameter updates for the whole training of ER-ACE with and without WSCL on Tiny-ImageNet$_{1/2}$ (from 10 epochs to 100 training epochs). 90

5.5 **WSCL model efficiency**: Left: the most frequent automatically learned freezing scheme (values within bars are number of parameters) during the wake phase for ER-ACE on Split ImageNet-100. Right: number of parameter updates for the whole training of ER-ACE with and without WSCL on Split ImageNet-100 (from 10 epochs to 100 training epochs). The numbers above the green bars represent the improvement in percent points with respect to the baseline alone. 90

6.1 **Dreamed classes generated by D2L**. Examples of dreamed classes synthesized from their corresponding real classes (left). These samples emerge as semantically distinct yet structurally coherent representations in the generator's latent space, forming intermediate concepts that enhance the continual classifier's generalization to future tasks. 94

6.2 **Overview of Dream2Learn.** (1) During CL training, a deep neural network (DNN) learns from real sensory images (the current task distribution plus the buffer) and dreamed samples produced by a latent diffusion model (LDM). (2) The dreaming optimization process refines the LDM prompts, with an Oracle Network providing a stopping criterion that prevents collapse. (3) Prompts generate auxiliary classes: dreamed samples are not buffered, but rather enrich the representation space with coherent latent clusters that foster knowledge reuse and adaptation (see Algorithm 1) 96

6.3 **Visualization of the dreaming optimization process in the latent space of a LDM.** Given a real sample $\mathbf{x}$, the optimization refines the soft prompt $\mathbf{p}_{\text{soft},c}$ to steer the diffusion model towards generating a dreamed counterpart that aligns with the target class c (e.g., a green mamba in this example). The dreaming process explores latent regions where images are visually similar yet distinct from target classes, forming novel intermediate classes (violet zones). 99

6.4 **Examples of dreaming optimization trajectories showing collapse**. From left to right, the images depict different stages of the optimization process. Each row illustrates the evolution of three example images throughout the same prompt optimization. Initially, the generated samples maintain meaningful variations. However, as optimization progresses, they become increasingly similar, reducing diversity and leading to less effective representations. 100

7.1 **Performance degradation during sequential training on five objects of the Linemod dataset.** Each color bar corresponds to a specific object. For example, pose estimation accuracy for *ape* (in red) drops from 92.1% at task 1 to 2.9% at task 5. 110
7.2 **Continual object 6D pose estimation framework.** Our model takes an RGB-D frame and its target object mask as input for 6D pose estimation. The trained model is sequentially adapted to new objects by using data from both the new object and the memory buffer, where keyframes of previously seen objects are stored. 112
7.3 **Visualization of 6D pose.** The results on (top) Linemod and (bottom) YCB-Video. The green 3D bounding boxes are obtained based on the ground truth 6D pose, while the red ones are generated from the model prediction after the incremental adaptation to the final object. 119
7.4 **Evaluation metrics for ILPose on Linemod.** First row: per-task ADD-0.1d(↑), $R_{err}(\downarrow)$ and $T_{err}(\downarrow)$ computed after training on each task. Second row: comparison between average metrics of our method and the Fine-tune model. 121
7.5 **Comparison of the impact of different keyframes selection strategies.** Per-task ADD-0.1d computed at the end of the training on the Linemod dataset. On the right, we report the ADD-0.1d$^{FA}(\uparrow)$. 122

8.1 **Overview of the proposed method.** On the left, the input sequence $\mathbf{x}_j$, collected over the K days preceding a given stress report y_i, consists of behavioral and contextual features such as app usage, phone call duration, physical activity, conversation, and academic deadlines. In the middle, the replay-based approach combines a sequence model, such as an RNN, with a memory buffer of past task examples. On the right, the prompt-based method leverages a transformer encoder and a memory buffer, using learnable task-specific prompts P_i prepended to the input sequence to enable task adaptation. 128
8.2 **Length of the stress self-reporting period across students.** Students to the right of the red dashed line are retained in the final subset of 16 students, while those to the left are excluded. 132
8.3 **Number of stress self-report submissions in the filtered subset.** For each of the 16 selected students, at least 30 submissions are available over the 10-week period. Values above the bars indicate exact counts. 133

List of Tables

3.1 **Results on Split CIFAR-10 in class-incremental learning with varying buffer sizes.** We report *final average accuracy* (FAA ↑) and *final average forgetting* (FAF ↓). We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column. 48

3.2 **Results on Split Mini-ImageNet in class-incremental learning with varying buffer sizes.** We report *final average accuracy* (FAA ↑) and *final average forgetting* (FAF ↓). We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column. 49

3.3 **Results on Split ImageNet-100 in class-incremental learning with varying buffer sizes.** We report *final average accuracy* (FAA ↑) and *final average forgetting* (FAF ↓). We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column. 50

3.4 **Results on Split Mini-ImageNet in class-incremental learning using ResNet-50 and DenseNet-121 as backbones.** We report *final average accuracy* (FAA ↑) for different buffer sizes. We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column. 51

3.5 **Interpretability evaluation results on Split Mini-ImageNet in class-incremental learning with buffer size of 500.** We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column. 55

4.1 **Effect of layer freezing when using a pre-trained backbone with DER++.** Results are computed at the end of the last task on Split CIFAR-10, for different pre-training modalities, in the Class-IL setting. 65

4.2 **Effect of layer freezing when using a pre-trained backbone with ER-ACE.** Results are computed at the end of the last task on Split CIFAR-10, for different pre-training modalities, in the Class-IL setting. 67
4.3 **Effect of selective freezing obtained using DER++ and ER-ACE methods.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting, for both 200 and 500 buffer sizes. 69
4.4 **Effect of selective freezing with DER++ and ER-ACE methods using a backbone pre-trained on the CIFAR-100 dataset.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting, for both 200 and 500 buffer sizes. 70

5.1 **Final average accuracy of rehearsal-based methods with and without WSCL in the Class-IL setting.** Results are reported for different buffer sizes. Bio-inspired methods are included for comparison. Bold values indicate the best performance in each column. 83
5.2 **Final forward transfer of rehearsal-based methods with and without WSCL in the Class-IL setting.** Results are reported for different buffer sizes. Bold values indicate the best performance in each column. 84
5.3 **Final forgetting of rehearsal-based methods with and without WSCL in the Class-IL setting.** Results are reported for different buffer sizes. Bold values indicate the best performance in each column. 85
5.4 **Final average accuracy on CORe50 with multiple dreaming datasets in the Class-IL setting.** Results are reported for rehearsal-based methods, with and without WSCL, across different buffer sizes. Bold values indicate the best performance in each column. 85
5.5 **Final average accuracy of state-of-the-art continual learning methods in the Class-IL setting.** Results are reported for different buffer sizes where applicable. Bold values indicate the best performance in each column. 86
5.6 **Ablation on the WSCL processing stages.** Results refer to ER-ACE on Tiny-ImageNet$_{1/2}$ and on CORe50 datasets with buffer size of 200. 87
5.7 **Comparison between WSCL and CaSpeR in terms of final average accuracy in the Class-IL setting.** Results are reported only for DER++ and ER-ACE, as the remaining methods on which CaSpeR is applicable manipulate future logits, hindering WSCL application. Dreaming datasets are CIFAR-100 and *ImageNet*^{aux}, respectively, for CIFAR-10 and CIFAR-100 benchmarks. Buffer size is 500. 88

5.8 **Impact of the selective freezing strategy on ER-ACE with WSCL.** We report final average accuracy and parameter updates (ΔU), i.e., the percentage change in the number of parameter updates induced by selective freezing relative to the no-freezing baseline. Results are computed on Tiny-ImageNet$_{1/2}$ with buffer size 500 for different numbers of epochs. 91

6.1 **Final average accuracy of rehearsal-based methods with and without D2L in the Class-IL setting.** Results are reported for buffer sizes of 2000 and 5000. Bold values indicate the best performance in each column. 103

6.2 **Final forward transfer of rehearsal-based methods with and without D2L in the Class-IL setting.** Results are reported for buffer sizes of 2000 and 5000. Bold values indicate the best performance in each column. 103

6.3 **Final average accuracy of state-of-the-art continual learning methods in the Class-IL setting.** Results are reported for buffer sizes of 2000 and 5000, where applicable. Bold values indicate the best performance in each column. 104

6.4 **Ablation on the oracle.** Results on Mini-ImageNet comparing our method with Fixed optimization. 105

6.5 **Impact of dream class updates.** Comparison on Mini-ImageNet of different dreaming strategies. 105

6.6 **Ablation on dream generation strategies.** Evaluation on Mini-ImageNet comparing interpolation-based baselines with our proposed D2L. 106

6.7 **Comparison with generative replay methods under the same buffer constraint (2000 samples).** While D2L is not a generative replay method, its use of generation for anticipatory transfer leads to superior performance compared to GR approaches. 107

7.1 **6D pose estimation results on Linemod.** We report final average metrics for different numbers of keyframes per object stored in the buffer. We compare ILPose against adapted incremental learning baselines. Bold values indicate the best performance in each column. 120

7.2 **6D pose estimation results on YCB-Video.** We report final average metrics for different numbers of keyframes per object stored in the buffer. We compare ILPose against adapted incremental learning baselines. Bold values indicate the best performance in each column. 120

7.3 **Effect of regularization.** We compare our original model, which employs EWC, with the L_2-based model, which uses L_2 regularization, while keeping all other settings unchanged. 122
7.4 **Grasping success rate.** The robot attempts to grasp the object from 6 different views. A grasp is counted to be successful when the robot gripper holds for at least 5 seconds. 123

8.1 **Performance of joint training across different model architectures.** Comparison of RNN, BiLSTM, and transformer with varying prompt set size $|P|$. Performance is reported in terms of MSE (↓) and MAE (↓), including standard deviations. 135
8.2 **Performance of continual training without rehearsal**: a comparison of RNN, BiLSTM, and transformer architectures. Performance is reported in terms of MSE (↓) and MAE (↓), including standard deviations. 136
8.3 **Comparison of buffer-only and prompt-augmented approaches for stress prediction across 16 tasks, with one student per task.** Results are reported by varying the memory buffer size ($|\mathcal{M}|$). In the prompt-based setting, "Tx" denotes the transformer encoder and $|P_i|$ the number of learned prompts per task. Performance is reported in terms of MSE (↓) and MAE (↓), with standard deviation. 136
8.4 **Comparison of buffer-only and prompt-augmented approaches for stress prediction across 8 tasks, with two students per task.** Results are reported by varying the memory buffer size ($|\mathcal{M}|$). In the prompt-based setting, "Tx" denotes the transformer encoder and $|P_i|$ the number of learned prompts per task. Performance is reported in terms of MSE (↓) and MAE (↓), with standard deviation. 137
8.5 **Comparison of buffer-only and prompt-augmented approaches for stress prediction across 4 tasks, with four students per task.** Results are reported by varying the memory buffer size ($|\mathcal{M}|$). In the prompt-based setting, "Tx" denotes the transformer encoder and $|P_i|$ the number of learned prompts per task. Performance is reported in terms of MSE (↓) and MAE (↓), with standard deviation. 137

List of Algorithms

1 Dream2Learn (D2L) 98
2 Incremental object 6D pose estimation for each task 114

Chapter 1

Introduction

Although machine learning methods are able to reproduce abilities natural to humans, they rely on foundations that are different from those underlying the human mind. Such divergence results in a lack of adaptability when contrasted with the human brain's remarkable capacity to continuously evolve through experience. The roots of this limitation can be found in the inception and development of machine learning.

1.1 Experience requires time

The question of how human beings become aware of the world has long been considered a fascinating problem. From ancient philosophy to contemporary neuroscience, scholars have investigated how sensory experience is perceived, how knowledge is acquired and retained, and how reasoning generates ideas. Within philosophy, these issues are addressed by gnoseology, the study of the nature and origin of human knowledge. Two of its principal schools of thought are *innatism* and *empiricism*.

First formulated by Plato in 385 BCE, innatism holds that the human mind possesses innate ideas or structures of knowledge prior to experience [12, 13]. In the *Meditationes de prima philosophia* [14], René Descartes offered a seventeenth-century reformulation of this view, distinguishing between ideas inscribed *a priori* in the human mind and those derived from imagination and the experience of the external world. On the empiricist side, in 350 BCE Aristotle laid the foundations of a contrasting view, rejecting the existence of innate ideas and maintaining that all cognition originates in sensory perception [15, 16]. In the *Essay Concerning Human Understanding* [17], John Locke further developed this tradition, claiming that the human mind at birth is a blank slate upon which ideas are imprinted through experience, specifically by means of sensation and reflection.

Machine learning models can be associated with Locke's *tabula rasa* when trained from scratch, as their parameters do not encode any task-specific knowledge and all relevant information must be acquired from the data used for training. In contemporary practice, this empiricist picture is complemented by a form of "machine innatism". Through large-scale pretraining, models are endowed with generic representations that play the role of innate structures for downstream tasks. In the standard supervised setting, a model is trained once on a static dataset and subsequently applied to new inputs, where it is expected to perform without any adaptation. Such an approach largely disregards the incremental nature of information in real-world scenarios, where data often becomes available only gradually and must be integrated over time.

In the late eighteenth century, in an effort to transcend the limitations inherent in both innatism and empiricism, Immanuel Kant articulated a position that foreshadowed many contemporary insights. In the *Critique of Pure Reason* [18], he argued that knowledge does not arise exclusively from experience, since the mind is structured by *a priori* forms and categories such as space, time and causality. These structures actively organize sensory input and thereby make empirical knowledge possible.

The *a priori* foundations described by Kant are closely aligned with contemporary evidence that humans are born with predispositions, biases, and structural priors that make

learning possible and orient the acquisition of knowledge from experience. Over time, experience enables the human mind to build upon these innate predispositions, progressively acquiring increasingly complex and sophisticated forms of knowledge.

This contrast highlights the asymmetry between natural and artificial learning processes: humans do not learn from scratch. When confronted with a novel task, we recognize familiar elements from previous contexts and use them as a starting point.

For instance, once we have formed the concept of a bicycle, we do not need thousands of examples of bicycles and motorcycles to distinguish between them. A few observations are sufficient for us to encode their shared structure, which includes two wheels, a frame, a saddle and handlebars, as well as their systematic differences, such as the presence of an engine, their typical sound and speed, and the way in which they are started and controlled. When we later encounter a moped or scooter, we again focus on how it reuses and recombines familiar features: it still has two wheels and handlebars, but combines a compact frame with a small engine and a different riding posture. Prior knowledge thus guides attention towards informative similarities and differences, allowing us to integrate the new concept from very few examples without overwriting the previous ones.

By contrast, while artificial systems may exhibit inductive biases which favor learning of certain data modalities, they generally rely on extensive exposure to large amounts of data in order to achieve comparable performance. A standard deep learning model typically needs many examples of bicycles, motorcycles and mopeds presented together in a single training dataset in order to disentangle these categories reliably. If they are learned sequentially and no specific mechanism is used to preserve past knowledge, the model instead tends to overwrite earlier representations, leading to a phenomenon known as "catastrophic forgetting".

1.2 Time erodes experience

Continual learning (CL) is the branch of artificial intelligence that studies how to train a model on a stream of tasks while preserving previously acquired knowledge. In this setting, neural networks may incur *catastrophic forgetting*: updates on new tasks can progressively overwrite earlier representations, leading to an abrupt degradation of performance on past tasks.

Originally referred to as "catastrophic interference", this phenomenon was first observed in early connectionist neural networks [19, 20, 21, 22]. In the late 1980s, McCloskey and Cohen [23] attributed it to the use of a single set of overlapping, distributed weights to store multiple memories. Subsequent work by Ratcliff [24] explored alternative strategies

for updating the weights, including modifying all connections, adapting only a subset of them, or adding new hidden units. Despite these efforts, none of the variants mitigated the forgetting problem. In retrospect, these early experiments anticipated several of the design dimensions explored by modern continual learning methods, such as constraining parameter updates, allocating additional capacity, or structurally separating new and old knowledge.

Since improving performance on the most recent patterns came at the expense of earlier ones, these findings reveal an intrinsic trade-off between retaining old information and incorporating new experience. This trade-off is an instance of the more general *stability–plasticity dilemma* [25]. The dilemma captures two opposing requirements for any learning system: the *plasticity* to rapidly encode new information and the *stability* to preserve useful knowledge that has already been acquired. Excessive plasticity leads to high levels of forgetting, as new learning overwrites older memories, while excessive stability makes the system rigid and unable to adapt to novel situations. Given that addressing this dilemma is far from trivial, the way the human brain balances stability and plasticity provides a source of inspiration for artificial neural networks.

Neural population codes provide the substrate for learning, shaping how new information is integrated and how much it interferes with existing knowledge [26, 27]. Many cortical representations are *sparse* and *selective*, meaning that only a small subset of neurons responds strongly to a given stimulus [28, 29]. Beyond efficiency, such coding schemes can limit interference by reducing overlap between the neural ensembles recruited by different stimuli or experiences, helping new information coexist with what has already been learned. Moreover, stabilizing newly acquired information is fundamental to reduce its susceptibility to subsequent interference. Synaptic plasticity rapidly adjusts connection strengths to incorporate recent experience, but these initial traces are fragile. Through *synaptic consolidation*, transient modifications are gradually stabilized, making newly acquired knowledge more resistant to disruption and enabling longer-term reorganization [30, 31, 32]. Nevertheless, when experiences are highly similar, consolidation alone is not sufficient to prevent nearby memories from collapsing into one another. In such cases, the hippocampal formation is often implicated, as it is thought to implement *pattern separation* by transforming overlapping inputs into more distinct internal codes, enabling reliable discrimination between similar episodes at recall [33, 34].

Further supporting the role of the hippocampus in memory consolidation, the *complementary learning systems* (CLS) theory [35, 36] frames learning as an interaction between two systems operating at different time scales. A fast-learning hippocampal system can rapidly encode new episodic experiences, while a slow-learning neocortical system gradually extracts statistical structure across many episodes while avoiding catastrophic interference.

Rapid hippocampal learning prevents new experiences from disrupting consolidated neocortical knowledge, whereas offline consolidation and replay progressively integrate hippocampal traces into more stable cortical representations [37]. Beyond wakefulness, the wake–sleep cycle provides a window in which recently acquired memories can be reactivated and reorganized [38, 39, 40]. During NREM sleep, slow-wave activity supports active systems consolidation by coordinating hippocampal replay with neocortical and thalamo-cortical oscillations, promoting the redistribution and long-term storage of memories [41, 42]. In parallel, sleep may also regulate plasticity via synaptic homeostasis mechanisms, counteracting the net synaptic potentiation accumulated during wake and restoring conditions that favor future learning [43].

In contrast, standard artificial neural networks typically incorporate new information by updating shared parameters, without an intrinsic mechanism to selectively modulate plasticity for those that are important to previously acquired knowledge. As a consequence, continual learning may induce progressive overwriting and reduced accuracy on earlier experiences. This effect is exacerbated when successive data distributions are highly overlapping, as similar inputs are mapped to nearby representations and therefore compete for shared features.

Even aside from the fact that continual learning departs from the standard offline paradigm, neural network training lacks an intrinsic consolidation stage. In the absence of a replay mechanism built into the training procedure, past experiences are not spontaneously reactivated. Therefore, mechanisms that mimic CLS interactions are not inherent to conventional training and thus must be introduced explicitly.

In view of these considerations, it is reasonable to wonder whether neuroscientific mechanisms might open up new research directions to address catastrophic forgetting. Indeed, the continual learning literature has long sought to overcome catastrophic forgetting by designing architectures and learning rules that maintain useful representations across tasks. Despite the broad range of strategies proposed in the literature, biologically inspired approaches remain relatively rare and are still far from matching the human ability to learn over time.

1.3 Injecting brain mechanisms into continual learning

This dissertation aims to address continual learning by leveraging neuroscientific principles to devise bio-inspired models and training procedures that improve long-term retention and mitigate catastrophic forgetting.

Continual learning in literature. Chapter 2 provides the methodological background for this thesis by formalizing the continual learning problem and introducing the most common experimental scenarios. It further discusses widely used benchmarks and evaluation metrics, and concludes with a survey of the major families of continual learning methods, outlining representative approaches and their key assumptions.

Neuro-inspired and cognitive continual learning. Chapter 3 addresses catastrophic forgetting by tackling representational interference at its source. Inspired by hippocampal pattern separation and sparse neural coding, the chapter introduces an architectural inductive bias that promotes highly selective activations. This selectivity is achieved by replacing standard linear transformations in convolutional neural networks with B-cos transforms, leading to more compartmentalized and inherently interpretable representations. The empirical analysis shows that these learned representations support continual learning, while model-inherent explanations reveal a trade-off between accuracy and explanation stability over time, suggesting that interpretability can provide an additional perspective on forgetting.

Complementary to the method discussed above, knowledge retention can also be pursued through mechanisms inspired by biological memory consolidation. Drawing inspiration from synaptic consolidation processes in the human brain, Chapter 4 addresses the stability-plasticity dilemma by introducing parameter freezing strategies that restrict updates to the most relevant network components. Both static and selective freezing strategies demonstrate that relevant features learned on previous tasks can be preserved and reused over time, reducing computational cost without compromising continual learning performance.

Building on this consolidation-inspired perspective, Chapter 5 draws on Complementary Learning Systems theory and on consolidation processes associated with wake–sleep cycles. The proposed approach uses a deep neural network to model neocortical learning and a two-level memory buffer to approximate hippocampal function, separating short-term storage from long-term consolidation. Within this structure, training is organized into an online *wake* phase, where the model rapidly adapts to new experience and stores episodic traces, and an offline *sleep* phase that alternates replay-based consolidation with a dreaming process. Experimental results show that alternating these states reduces forgetting and enables positive forward transfer, indicating that offline consolidation can prepare the network for future knowledge.

Since the initial dreaming mechanism relied on surrogate real data, Chapter 6 moves toward the autonomous generation of task-relevant "dreams" aimed at constructing representations more conducive to future adaptation. By synthesizing dream samples directly

from the classifier's internal representations, the proposed generative dreaming process expands the representation space and turns negative forward transfer into positive.

Applications of continual learning. Beyond standard image classification benchmarks, the remainder of this thesis illustrates how continual learning extends to real-world applications. In these domains, the key challenges of continual learning become concrete, as models must adapt over time under changing conditions and with limited access to past data.

Chapter 7 investigates incremental 6D object pose estimation for robot grasping under continual learning constraints. In many robotic deployments, access to past data may be limited or unavailable due to resource constraints and privacy requirements, making repeated full retraining impractical [44, 45]. Although evaluated on standard benchmarks, the proposed setting is motivated by increasingly relevant applications such as robot-assisted surgery, where accurate pose estimation is essential for safe instrument manipulation in the presence of evolving workflows and tool inventories [46, 47]. In this setting, results highlight that continual adaptation can preserve pose estimation performance over time, and that even modest gains in perception can improve the reliability of downstream grasping.

Chapter 8 investigates mental well-being monitoring from mobile sensing data within a continual learning framework. Given that psychological stress is recognized as a major health concern [48, 49, 50], the chapter focuses on reliable personalized methods for monitoring and predicting stress in daily life. It models each user as an incremental task and evaluates replay-based strategies, including prompt-based adaptations, in both task- and domain-incremental settings. The findings indicate that replay-based continual learning can preserve personalization over time, enabling models to adapt to temporal shifts in user behavior and evolving sensing streams without requiring full retraining.

Chapter 2

Foundations of Continual Learning

By more faithfully mirroring the dynamics of real-world learning, the continual learning setting has found applications across a broad spectrum of domains, ranging from medical diagnosis to autonomous driving and content recommendation. The present thesis aligns with that line of research, focusing specifically on the application of continual learning to visual classification.

2.1 Problem formulation

Continual learning investigates the training of models in scenarios where data are presented as non-stationary streams, and the classical assumption of independently and identically distributed (*i.i.d.*) samples is not satisfied. Rather than operating on a fixed dataset, the learner is exposed to information in a sequential fashion, while access to previously observed samples is either absent or constrained.

Formally, a *task* is defined as the transition from one stationary data distribution to another. Let $\{\tau_1, \ldots, \tau_T\}$ denote a sequence of T tasks with associated data streams $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$. Each sample $(\mathbf{x}, y) \in \mathcal{D}_i$ consists of a data point $\mathbf{x} \in \mathcal{X}$ and the corresponding class label $y \in \mathcal{Y}$. Since the distribution may change across tasks in terms of both input data and class label distributions, samples within $\mathcal{D}_i$ are *i.i.d.*, whereas the global distribution $\mathcal{D}$ differs from this assumption.

Given a classifier $f : \mathcal{X} \to \mathcal{Y}$ with parameters $\boldsymbol{\theta}$, the objective of continual learning is to train f on the data stream $\mathcal{D}$ under the constraint that, at each task τ_i, the model receives inputs sampled exclusively from the corresponding task-specific distribution $(\mathbf{x}, y) \sim D_i$.

The training objective is to minimize a classification loss over the entire sequence of tasks, while preserving performance on previously learned tasks, for the model evaluated at the end of training:

$$\arg\min_{\boldsymbol{\theta}_T} \sum_{i=1}^{T} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_i} \Big[\mathcal{L}\Big(f\left(\mathbf{x}; \boldsymbol{\theta}_T\right), y \Big) \Big] \tag{2.1}$$

where $\mathcal{L}$ denotes a generic classification loss (e.g., cross-entropy).

2.2 Scenarios

In line with the prevailing literature, the incremental learning problem can be posed in terms of three main scenarios: task-, domain-, and class-incremental learning. These scenarios provide a common taxonomy for organizing existing methods by clarifying their underlying assumptions about data availability and task information at inference time [51]. In particular, they differ in whether the task identity is known or unknown, and in how the output space is structured across tasks. In what follows, the abovementioned scenarios are presented and discussed in detail.

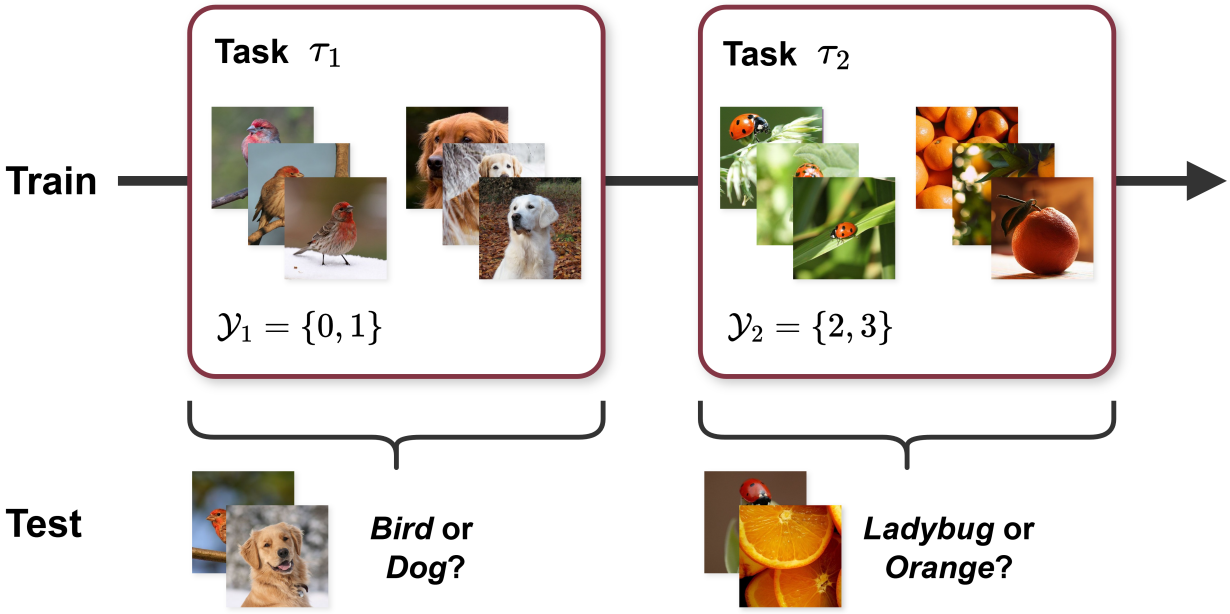

Figure 2.1: **Task-incremental learning (Task-IL) scenario.** Each task i introduces a distinct set of classes $\mathcal{Y}_i$. At inference time, since task identity is available, the model is required to predict only among the classes associated with the task to which the input sample belongs.

2.2.1 Task-incremental learning

In a *task-incremental learning* (Task-IL) scenario, a model incrementally learns a set of distinct tasks with $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset, \forall i \neq j$, under the assumption that the identity of the current task is always available. This assumption reduces the problem to distinguishing among the classes of the current task, thereby making Task-IL the simplest continual learning scenario.

The availability of task identity at test time allows for models with task-specific components, such as separate output heads, or even entirely distinct networks per task, in which case catastrophic forgetting can be avoided. Therefore, the main challenge in task-incremental learning is to design mechanisms for sharing representations across tasks while balancing performance and computational cost.

2.2.2 Domain-incremental learning

In the *domain-incremental learning* (Domain-IL) scenario, a model incrementally encounters data from different domains while the set of class labels remains fixed across all tasks. The overall number of classes is specified from the outset, and each task is associated with the same output space.

Unlike Task-IL, at inference time the task identity is not available, thus catastrophic

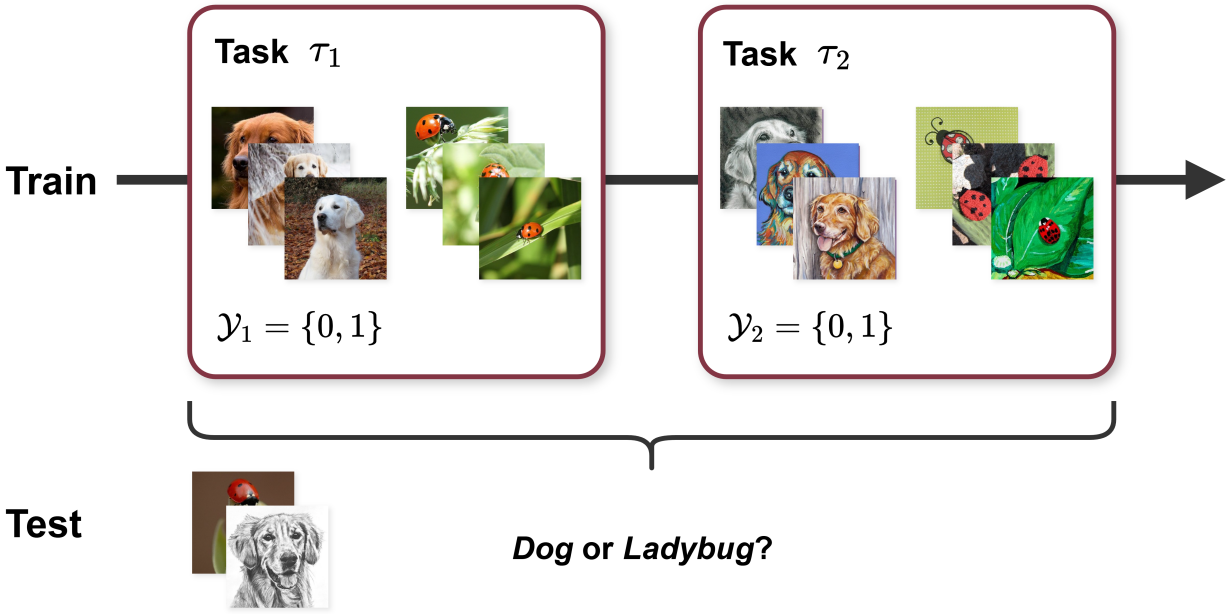

Figure 2.2: **Domain-incremental learning (Domain-IL) scenario.** Each task i exposes the model to data from a different domain while sharing the same label space $\mathcal{Y}$. As task identity is not available at inference, the model must predict among all classes encountered so far, despite domain shifts that may substantially alter the visual appearance of the inputs.

forgetting cannot be avoided by design through task-specific components. As a consequence, the main challenge stems from shifts in the input and class-conditional distributions, which induce different internal representations as new domains are observed. Such domain shifts may arise, for instance, from changes in lighting, background, sensor characteristics, or from artificial transformations such as permutations applied independently to each task. This scenario is closely related to domain adaptation settings [52, 53, 54], with the additional constraint that a single model must be updated over time while retaining good performance on previously seen domains.

2.2.3 Class-incremental learning

In the *class-incremental learning* (Class-IL) scenario, a model is exposed to a sequence of tasks, each introducing a new subset of classes that does not overlap with those of previous tasks. As training progresses, the classifier is required to discriminate among an ever-growing set of categories, while at inference time no information about the task identity is provided. In contrast to Task-IL, where the prediction space is limited to the classes introduced within each task, Class-IL requires the model to predict over the union of all previously encountered classes, resulting in a more demanding classification problem.

The main challenge in this scenario is to preserve the ability to distinguish between classes

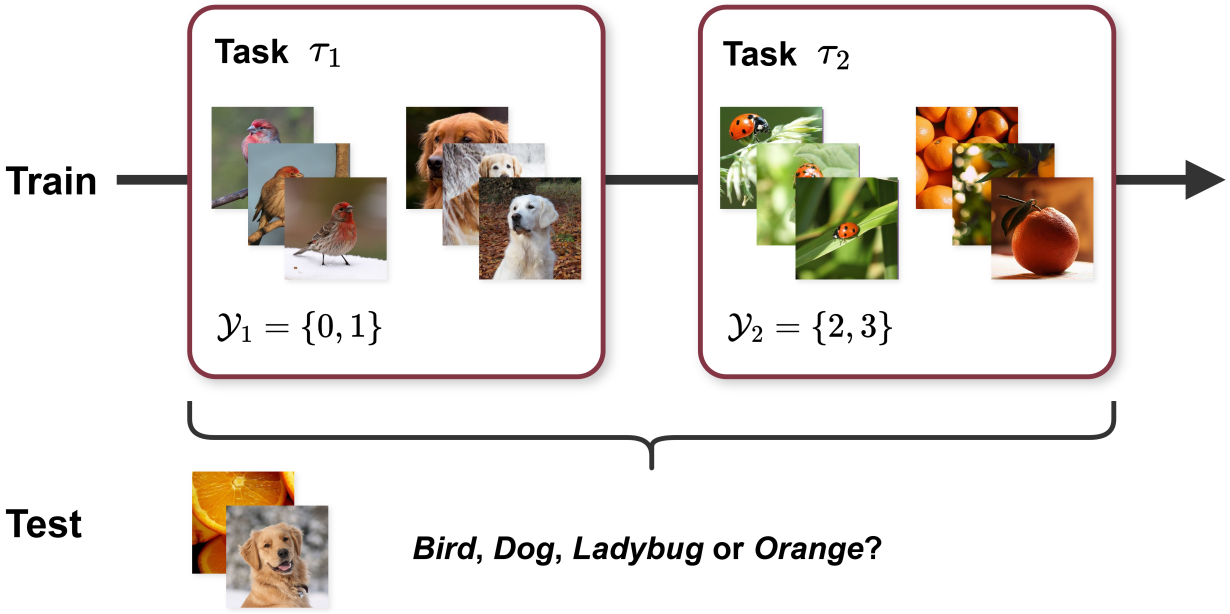

Figure 2.3: **Class-incremental learning (Class-IL) scenario.** Each task i introduces a distinct subset of classes $\mathcal{Y}_i$. At inference time, since task identity is not available, the model must discriminate among all classes encountered so far.

that have never been observed together, particularly those belonging to different tasks, as the model tends to bias its predictions toward the most recently learned classes. This difficulty is further exacerbated in settings in which storing and replaying examples from previous tasks is constrained or not permitted, for instance due to privacy constraints or limited memory resources. Owing to this increased complexity, Class-IL is considered the most challenging continual learning setting and is thus commonly adopted as the primary benchmark.

2.3 Benchmarks

Benchmarking continual learning methods in image classification typically relies on well-established vision datasets originally developed for standard supervised classification, which are then appropriately partitioned into meaningful sequences of tasks.

Early approaches relied on low-resolution digit-recognition benchmarks, including SVHN [55] and MNIST-based variants [56], where controlled transformations of the input images give rise to sequences of tasks or domains. Among these, the following benchmarks were frequently employed:

- **Split MNIST**: constructed from the MNIST dataset, which contains 70,000 28×28 grayscale images of handwritten digits across 10 classes. The label space is generally partitioned into 5 sequential binary classification tasks.

- **Permuted-MNIST** [57]: tasks are constructed by applying distinct, fixed random permutations to the pixels of all images in each task, while preserving the original label space. As the resulting sequence exhibits substantial domain shifts, Permuted-MNIST has become a standard benchmark for Domain-IL evaluation.

- **Rotated-MNIST**: tasks are generated by rotating all images by different angles (e.g., between 0° and 180°). As with Permuted-MNIST, the label space remains fixed across tasks, while the input distribution changes systematically, providing a controlled testbed for studying Domain-IL.

- **Split SVHN**: derived from Google Street View images [55], SVHN contains over 600,000 32×32 RGB images of house-number digits captured in real-world environments. It comprises 10 classes corresponding to the digits 0-9, which are usually split into 5 sequential tasks. Split SVHN provides a more challenging benchmark than MNIST due to cluttered backgrounds, illumination variability, and natural scene complexity.

As the complexity of proposed continual learning methods has increased, more recent works have also employed ImageNet-based subsets [58]. These datasets offer testbeds that more closely approximate real-world variability and enable the evaluation of scalability. The following benchmarks are derived from ImageNet:

- **Split Mini-ImageNet** [59]: contains 60,000 84×84 RGB images across 100 classes; for each class, 500 images are used for training and 100 for evaluation. The classes are typically organized into 20 sequential tasks, each comprising 5 classes.

- **Split Tiny-ImageNet** [60]: comprises 100,000 64×64 RGB images from 200 classes, usually divided into a sequence of 20 tasks of 10 classes each. For each class there are 500 images for training, 50 images for validation and 50 images for testing.

- **Split ImageNet-100** [61][1]: contains high-resolution RGB images of 100 animal classes belonging to 7 different species (*annelids*, *arachnids*, *birds*, *clams*, *fishes*, *reptiles*, *shellfish*), which reduces inter-class semantic diversity and makes the classification tasks more fine-grained. Each class contains 1,300 samples for training and 50 for evaluation. The classes are generally split into a sequence of 20 tasks of 5 classes each.

- **Split ImageNet-Rendition (ImageNet-R)** [62]: comprises artistic and non-photorealistic renditions of 200 ImageNet classes (e.g., paintings, sculptures, embroidery,

[1]Split ImageNet-100 is derived from https://www.kaggle.com/datasets/ambityga/imagenet100

cartoons), with 150 samples per class, introducing strong intra-class variations. The class set is partitioned into 10 tasks of 20 classes each. It serves as a benchmark for evaluating model performance under significant domain shifts between tasks.

In addition to those discussed above, other relevant datasets include:

- **Split CIFAR-10** [63]: contains 60,000 32×32 RGB images across 10 classes, with 5,000 training images and 1,000 test images per class. The label space is commonly split into 5 sequential binary classification tasks, each involving 2 classes.
- **Split CIFAR-100** [63]: obtained by typically dividing the 100 classes of the original dataset into 10 sequential tasks of 10 classes each. Each class consists of 500 training and 100 test 32×32 RGB images, resulting in 5,000 training images and 1,000 test images per task.
- **Split CUB-200** [64]: based on the CUB-200-2011 dataset, this benchmark consists of 11,788 high-resolution RGB images of 200 bird species. The classes are usually split into 10 sequential tasks of 20 classes each, with approximately 30 training and 30 test images per class, which are resized to 224×224.
- **Split DomainNet** [65]: a large-scale dataset for multi-domain image classification, comprising about 600,000 RGB images over 345 classes collected from 6 distinct domains: *real*, *clipart*, *painting*, *sketch*, *infograph*, and *quickdraw*. Each domain is treated as a separate task sharing the same label space, providing a challenging benchmark for Domain-IL under severe distribution shifts.

2.4 Evaluation metrics

To evaluate the quality of continual learning methods, several metrics have been proposed that quantify how training on one task influences performance on others over time. In the following, let a_i^t denote the accuracy on the i-th task after training up to task t, and let T be the total number of tasks.

Final Average Accuracy (FAA). It is the primary evaluation metric and measures the average accuracy across all tasks after learning the last task. The FAA is defined as:

$$\text{FAA} \triangleq \frac{1}{T} \sum_{i=1}^{T} a_i^T, \tag{2.2}$$

In an ideal scenario, acquiring new tasks would both facilitate future learning and leave previously acquired knowledge intact. In particular, learning earlier tasks would make it easier to learn subsequent ones, e.g., by providing reusable representations or more general features. At the same time, when new tasks are learned, performance on earlier tasks would remain unchanged, meaning that no interference occurs. Such desirable effects are captured by two metrics referred to as *forward* and *backward transfer*, respectively.

Final Backward Transfer (FBWT). First introduced in [66], this metric measures how learning subsequent tasks affects performance on previously learned ones. Final Backward Transfer is defined as:

$$\text{FBWT} \triangleq \frac{1}{T-1}\sum_{i=1}^{T-1}(a_i^T - a_i^i), \tag{2.3}$$

i.e., the average difference between the final accuracy on each task i (after training on all T tasks) and its accuracy immediately after it was learned. Positive values indicate *positive backward transfer*, meaning that training on later tasks has improved performance on earlier ones, whereas negative values reflect forgetting.

Final Forward Transfer (FFWT). This metric quantifies how learning preceding tasks influences performance on tasks that have not yet been trained on. Following [66], Final Forward Transfer is computed as:

$$\text{FFWT} \triangleq \frac{1}{T-1}\sum_{i=2}^{T}(a_i^{i-1} - a_i^{\text{random}}), \tag{2.4}$$

where a_i^{i-1} denotes the accuracy on the i-th task obtained by a model that has been trained on tasks $1, \ldots, i-1$, and a_i^{random} denotes the accuracy on the i-th task using the same architecture with random initialization. A positive value of FFWT indicates that prior training on earlier tasks leads to better-than-random performance on future tasks, capturing beneficial forward transfer.

While these metrics are designed to capture desirable properties of continual learning, in practice their behavior often departs from the ideal case. When models are trained sequentially, performance on earlier tasks typically deteriorates as additional tasks are learned, leading to *negative* backward transfer values as a direct manifestation of catastrophic forgetting. Moreover, genuine beneficial influence of past learning on future tasks is rarely observed. In Class-IL and Task-IL scenarios, where distinct classes are introduced in separate tasks, such beneficial influence would require the model to correctly classify classes

that have not yet been observed, making positive forward transfer essentially unattainable. By contrast, in Domain-IL settings, where the same classes reappear under different transformations or domains, positive values of forward transfer may be feasible, since knowledge acquired on one domain can, in principle, facilitate learning in subsequent domains.

Final Forgetting (FF). This metric quantifies the severity of performance degradation on previously learned tasks. It is defined as the average drop in accuracy between the best performance achieved on each past task and its final accuracy after learning all T tasks:

$$\mathrm{FF} \triangleq \frac{1}{T-1} \sum_{i=1}^{T-1} \max_{l \in \{1,\dots,T-1\}} a_i^l - a_i^T, \tag{2.5}$$

where a_i^t denotes the accuracy on the i-th task after training up to task t. A positive value of FF indicates that the model has forgotten part of what it initially learned on earlier tasks, whereas a negative value implies that subsequent training has improved performance on those tasks.

In addition to the evaluation metrics discussed above, criteria related to computational and architectural overhead may also be considered. Many approaches rely on storing task-specific parameters, replay buffers, or auxiliary structures, which directly affects the memory footprint of the method. The practicality of a given approach therefore depends on how much additional data and metadata must be maintained over time; this includes not only the size of the core model, but also the cumulative amount of stored examples required for replay. Similarly, the computational cost associated with training and inference is an important factor, for instance in terms of additional passes over previously seen data or extra modules that must be evaluated at test time.

Furthermore, an essential aspect is whether the algorithm assumes access to explicit task boundaries, that is, whether it must be informed when a task starts and ends or can instead operate in a fully online setting without such information.

Taken together, these requirements strongly influence the suitability of a method for real-time or streaming environments, in which computational resources are constrained and task changes are not explicitly reported.

2.5 State of the Art

In recent years, a significant number of methods has been proposed to mitigate catastrophic forgetting in continual learning. Most approaches can be organized into four broad fami-

lies, depending on how they store, constrain, or otherwise exploit task-specific information throughout sequential training [67]. This section briefly surveys the main representatives of each family, which will serve as reference baselines in the experimental analysis.

2.5.1 Regularization-based methods

Regularization-based approaches address catastrophic forgetting by constraining parameter updates during training on new tasks, so as to preserve knowledge that is estimated to be important for previously learned tasks. To this end, additional penalty terms are introduced into the loss function, to discourage large deviations from parameter configurations that were beneficial in the past or to promote shared representations across tasks. This strategy is particularly effective when tasks share a substantial portion of their underlying feature representations, since many of the features learned on earlier tasks can be reused and selectively adapted for subsequent ones.

Regularization techniques can be divided into two subcategories: *functional* and *structural* methods.

Functional methods. Inspired by knowledge distillation [68, 69], these methods operate by regularizing the model at the output or feature level rather than directly on its parameters. The core idea is to maintain consistency between the responses of the model on previously learned tasks and its responses after learning new tasks, by adding distillation losses on logits or intermediate representations.

- **Learning without Forgetting (LwF)** [70] is one of the most relevant functional methods. Before training on a new task, the outputs of a frozen copy of the previous model are computed on the new task data and then used as soft targets during optimization. A distillation loss encourages the current model to match these reference predictions associated with previous tasks, preserving prior knowledge while learning the new task. However, its effectiveness can degrade when the distribution of the new task is poorly aligned with that of earlier tasks, as the stored outputs may become biased or uninformative in the presence of strong domain shift.

- **Encoder-Based Lifelong Learning (EBLL)** [71] extends LwF by preserving low-dimensional feature representations of previous tasks. For each task, an undercomplete autoencoder is trained jointly with the backbone network to reconstruct the features extracted from that task's data. When a new task is introduced, an additional regularization term constrains the current feature projections not to deviate from those reconstructed by the task-specific autoencoders, helping retain discriminative features

learned on earlier tasks. As a consequence, a disadvantage of this design is that a separate autoencoder must be stored for each task and evaluated during training on new tasks.

Structural methods. These approaches estimate a per-parameter importance measure after training on each task and then introduce a regularization term that discourages large updates to these parameters when learning subsequent tasks. The model is encouraged to retain useful knowledge from earlier tasks while still adapting its remaining degrees of freedom to new data.

- **Elastic Weight Consolidation (EWC)** [72] preserves performance on previously learned tasks by penalizing changes to parameters that are estimated to be important for those tasks. After training on a task, the diagonal of the Fisher Information Matrix is used to compute an importance score for each parameter. For subsequent tasks, EWC augments the loss with a quadratic penalty that keeps important parameters close to their previously learned values. Achieving this reduction in interference among tasks requires storing, for each task, the corresponding parameter values and their associated importance scores used in the regularization term.

- **Online Elastic Weight Consolidation (oEWC)** [73] is an online variant of EWC that avoids maintaining a separate Fisher matrix for every task. Instead, it keeps a single consolidated estimate of parameter importance, updated in a recursive manner as new tasks are learned, and applies a corresponding quadratic penalty during training. The memory and computational cost of oEWC remains independent of the number of tasks, while often achieving performance comparable to the original EWC.

- **Synaptic Intelligence (SI)** [74] is conceptually similar to EWC, but computes parameter importance in a fully online fashion during training. For each parameter, SI accumulates a path integral over time of the product of the parameter updates and the corresponding gradients, measuring how much that parameter contributed to reducing the loss on the current task. After training on the task, these importance scores are normalized and used to define a quadratic penalty that discourages large changes to important parameters when learning new tasks.

- **Memory Aware Synapses (MAS)** [75] is an online and unsupervised regularization method. For each parameter, MAS accumulates an importance score based on the sensitivity of the network's output to small perturbations of that parameter on a stream of unlabeled data. When a new task is learned, a quadratic penalty exploits these scores to discourage large updates to those deemed highly relevant.

2.5.2 Architectural methods

Architectural methods tackle catastrophic forgetting by modifying the network structure to accommodate new tasks while preserving knowledge of previous ones. In these approaches, disjoint or sparsely overlapping subsets of parameters are allocated to different tasks, either by dynamically expanding the network with new layers, neurons, or heads, or by isolating task-specific parameter subsets through masks, gates, or similar mechanisms. When a new task arrives, the model instantiates a new subnetwork or task-specific modules while keeping the parameters associated with earlier tasks fixed. A major disadvantage of this design is that it requires access to a task identifier or an external task-selection mechanism at test time, restricting these methods to Task-IL scenarios. Moreover, the number of parameters grows with the number of tasks, which limits scalability and reduces their suitability for resource-constrained settings.

- **Progressive Neural Networks (PNN)** [76] were proposed in a deep reinforcement learning setting, but the underlying idea can also be applied to supervised continual learning. For each new task, a new task-specific parallel network, or *column*, is added to the architecture, while the parameters of previously trained columns are frozen. Lateral connections from earlier to later columns allow the new task to reuse features learned on previous tasks while preventing interference in the opposite direction.

- **Expert Gate** [77] implements a network of experts: each task is assigned its own expert model and an autoencoder-based gate. At test time, each task-specific autoencoder computes the reconstruction error on the input, and the sample is routed to the expert whose autoencoder attains the lowest error. The main limitations of this design are the need to store a separate expert–gate pair per task and the reliance on accurately identifying the active task at test time.

- **PackNet** [78] allows several tasks to be "packed" into a fixed-capacity network with limited additional memory overhead. After training on a task, a subset of low-magnitude weights is pruned and the remaining weights are fine-tuned to recover performance. The freed weights are then reused to learn new tasks, with binary masks indicating which weights are reserved for previous tasks and which remain available for adaptation. As more tasks are added, the pool of free weights shrinks, which can eventually limit performance on later tasks.

- **Hard Attention to the Task (HAT)** [79] is a task-incremental method that employs per-task attention masks to activate specific units of the network. During training on

a new task, the model jointly learns the weights and a mask that gates activations and gradients, so that parameters identified as important for previous tasks are protected from further updates.

- **Dynamically Expandable Network (DEN)** [80] learns a shared but partially task-specific representation by selectively expanding the architecture. After training on a new task, sparse regularization and selective retraining are used to identify parameters that are sufficient to fit the new data. If this capacity is insufficient, additional units and connections are added to specific layers, while redundant neurons are split or pruned to keep the model compact.

2.5.3 Replay-based methods

Replay-based or *rehearsal* methods mitigate catastrophic forgetting by replaying data from previous tasks while learning new ones. In most implementations, a fixed-size buffer stores a subset of past examples, which are then interleaved with the current task data during training. These methods often outperform regularization-based approaches and, unlike most architectural methods, do not require knowledge of the task identity at inference time. Due to their simplicity and empirical effectiveness, rehearsal-based approaches are widely employed to preserve previously acquired knowledge in dynamic real-world scenarios.

The main limitation of rehearsal methods is the need to retain real samples in memory, which may be impractical in scenarios with limited storage or stringent privacy constraints. To mitigate this issue, a subcategory of replay approaches, known as *generative methods*, replaces stored samples with synthetic data generated by a learned generative model, such as GANs [81], VAEs [82, 83], or diffusion-based models [84, 85]. This strategy, however, introduces additional complexity: the generative model itself must be trained in a continual fashion and is prone to its own form of forgetting, distribution drift, or mode collapse. Moreover, training and sampling from such models can be computationally expensive, and the quality and diversity of the generated samples often fall short of real data. On commonly used vision benchmarks with moderate buffer sizes, simple rehearsal with real exemplars remains a very strong baseline, and generative replay has so far struggled to consistently match its performance, even though recent advances in latent and diffusion-based generative replay are beginning to narrow this gap.

- **Experience Replay (ER)** [86] is one of the earliest rehearsal strategies: during training on a new task, mini-batches are formed by mixing samples from the current data stream with examples drawn from a replay buffer. When the buffer has limited capacity, reservoir sampling [87] can be used to maintain an approximately uniform sample

of all previously seen data. Despite its simplicity, ER has inspired many subsequent replay-based methods and remains a strong baseline in continual learning.

- **Incremental Classifier and Representation Learning (iCaRL)** [88] uses a distillation loss to preserve the network's predictions on previously learned classes. Classification is performed using a nearest-mean-of-exemplars rule in the learned feature space. The fixed-size buffer is updated using a herding strategy to approximate the class mean of each category and provide a compact summary of past data.

- **Gradient Episodic Memory (GEM)** [66] augments rehearsal with gradient-based constraints that prevent performance degradation on past tasks. At each update, GEM projects the gradient for the current mini-batch onto the closest direction that does not increase the loss on any episodic memory stored in the buffer. This mechanism requires storing separate memories for all past tasks and solving a constrained optimization problem at every training step.

- **Function Distance Regularisation (FDR)** [89] adds a loss term that penalizes the L^2 distance between the outputs of the current network and those of the previous network on samples stored in the replay buffer, effectively performing self-distillation on stored exemplars and limiting interference with prior knowledge.

- **Bias Correction (BiC)** [90] introduces a small bias-correction layer after the classifier, whose parameters are fitted on a held-out, class-balanced validation set. This post hoc calibration compensates for the tendency to overpredict new classes.

- **Gradient-based Sample Selection (GSS)** [91] updates the replay buffer to maximize gradient diversity, so that the stored samples remain informative about past tasks. More specifically, sample selection is formulated as a constraint-reduction problem where the goal is to retain a fixed-size subset whose gradients closely approximate the feasible region induced by all past updates.

- **Averaged Gradient Episodic Memory (A-GEM)** [92] is an efficient variant of GEM that enforces a single constraint on the average loss over replay samples instead of maintaining separate constraints for each past task. At each update, the gradient for the current mini-batch is projected so that it does not increase the loss on a randomly sampled batch from the episodic memory.

- **Learning a Unified Classifier Incrementally via Rebalancing (LUCIR)** [93] combines rehearsal with a cosine-normalized classifier, a feature-distillation loss, and

a margin-based inter-class separation loss. This rebalancing strategy encourages class-discriminative representations, mitigating the bias towards new classes.

- **Hindsight Anchor Learning (HAL)** [94] learns a small set of anchor points for each class via bilevel optimization, selecting samples that are estimated to be most susceptible to forgetting. A regularization term then keeps the model's predictions on these anchors close to their past values, improving retention especially when the replay buffer is small.

- **Greedy Sampler and Dumb Learner (GDumb)** [95] maintains a fixed-size, approximately class-balanced memory while performing no parameter updates on the data stream. Incoming samples are greedily stored in the buffer to preserve class balance, and the model is re-trained from scratch on the buffered examples whenever evaluation is required.

- **Dark Experience Replay (DER)** [96] extends experience replay by storing, for each buffered example, the logits produced by the model when the example is first observed. During replay, the model is trained not only with a cross-entropy loss on the current labels but also with an auxiliary self-distillation loss that regresses towards these stored logits, promoting consistency with its past predictions across tasks. **DER++** [96] further augments this objective by adding an additional cross-entropy loss on replayed samples.

- **eXtended Dark Experience Replay (X-DER)** [97] builds on DER/DER++ by revising both how logits are stored and how the classifier is trained. First, the stored logits of buffered examples are periodically updated to incorporate *future past* information, i.e., relationships between stored samples and classes introduced in later tasks. Second, future-preparation and bias-mitigation losses warm up logits associated with unseen classes and reduce the bias towards the current task.

- **Regular Polytope Classifier (RPC)** [98] mitigates classifier bias by pre-allocating and freezing classifier weights for all classes that will appear across tasks, arranging them as vertices of a regular polytope in the feature space. During each incremental step, only the feature extractor is updated using classification and distillation losses, while the fixed classifier reduces the model's prediction bias towards new classes.

- **DualNet** [99] introduces a dual-backbone architecture that decouples incremental classification from representation learning. A *slow learner* backbone is trained with a self-supervised objective on i.i.d. samples drawn from the replay buffer, aiming to build

a transferable representation. A separate *fast learner* backbone is then trained in a standard supervised manner on the continual stream, solving the incremental tasks while leveraging the features provided by the slow learner.

- **Continual Prototype Evolution (CoPE)** [100] performs classification using class prototypes in the learned feature space. As new data arrive, prototypes are incrementally updated to balance the influence of recent samples with the need to preserve previously acquired structure in the latent space, mitigating abrupt shifts in decision boundaries.

- **Contrastive Continual Learning (CO^2L)** [101] replaces the standard cross-entropy objective with a supervised contrastive loss [102] applied to samples from the replay buffer and the current stream. By optimizing this contrastive objective, CO^2L encourages class-discriminative, well-clustered representations that better exploit replayed examples. For evaluation, a linear classifier is trained on the features of the buffered samples.

- **Experience Replay with Asymmetric Cross-Entropy (ER-ACE)** [103] refines standard ER by decoupling the cross-entropy contributions of current and replay samples. In ER-ACE, the loss is designed so that gradients from current data do not overwrite already learned logits associated with past classes, while replay samples focus on reinforcing previous decision boundaries. This asymmetric treatment of stream and replay examples alleviates the bias towards the most recent classes.

2.5.4 PEFT-based methods

Parameter-efficient fine-tuning (PEFT) methods adapt large pretrained models by freezing most backbone parameters and learning only a small number of additional task-specific parameters.

A prominent line of work operates on pretrained Vision Transformers (ViTs) and keeps the backbone frozen while learning prompt vectors that are injected as additional tokens or inputs to the transformer [104, 105, 106]. Other approaches generalize PEFT to a broader class of modules, such as adapters or low-rank updates [107, 108]. Although these methods can achieve strong performance in rehearsal-free or low-buffer regimes on benchmarks, they rely on the availability of a large pretrained model and on the alignment between its pretraining distribution and the downstream continual learning tasks, which makes direct comparison with methods trained from scratch non-trivial.

- **Learning to Prompt (L2P)** [104] is one of the first prompt-based continual learning methods for pretrained ViTs. While the backbone is kept frozen, a pool of learnable prompts is maintained. For each incoming sample, a query mechanism selects a subset of prompts based on the input features and prepends them to the token sequence. This mechanism makes L2P rehearsal-free and removes the need for task identity at test time.

- **DualPrompt** [105] extends L2P by decoupling shared and task-specific conditioning in the prompt space. At each transformer layer, it introduces two complementary types of prompts: general prompts, which capture task-invariant knowledge shared across tasks, and expert prompts, which encode task-specific information.

- **CODA-Prompt** [106] employs a decomposed, attention-based prompting mechanism. Instead of selecting a single prompt per input, it learns a set of prompt components and uses input-conditioned attention weights to assemble them into prompts, generating input-dependent prompt vectors.

- **Learning-Accumulation-Ensemble (LAE)** [107] proposes a unified framework that can incorporate different PEFT modules (e.g., adapters, LoRA, prefix tuning) into continual learning. LAE decomposes adaptation into three stages: *Learning*, which tunes an online PEFT module on the current task; *Accumulation*, which transfers task-specific knowledge into an offline PEFT module via momentum updates; and *Ensemble*, which combines the predictions of online and offline experts at inference time.

- **InfLoRA** [108] injects low-rank adaptation [109] branches into a frozen pretrained backbone to reparameterize its weights. Each branch is decomposed into a fixed projection matrix and a learnable expansion matrix. Before training on a new task, the projection matrix is constructed using estimates of the gradient subspaces of old and new tasks, so that it is approximately orthogonal to gradients of previous tasks while lying within the gradient subspace of the current one. During training, only the expansion matrix of the current branch is updated, while the backbone and all projection matrices remain frozen.

Chapter 3

Architectural Inductive Bias for Mitigating Catastrophic Forgetting

Catastrophic forgetting arises when features learned for earlier tasks are indiscriminately updated to accommodate subsequent tasks, a problem amplified by the high plasticity of standard neural network layers. From a neuro-inspired perspective, hippocampal *pattern separation* reduces representational overlap between highly similar experiences by transforming them into more distinct internal codes, limiting mutual interference in episodic memory [33, 34]. Beyond this mechanism, *sparse and selective neural coding* limits representational collisions by concentrating strong activation on a relatively small subset of units [27, 28, 29]. Hence, an ideal continual learning model would develop robust, task-specific features and, to a certain degree, "protect" them from being overwritten. This motivates methods that instill an inductive bias for feature selectivity directly into the model's architecture, encouraging the learning of representations that are naturally more compartmentalized and resistant to destructive updates.

In this chapter, we argue that architectural choices at the neuron level can provide such an inductive bias. We propose the use of B-cos networks [110], which replace standard linear operations with a cosine similarity-based transform. The underlying idea is that this modification fundamentally changes the condition under which a neuron activates in a way that is suitable to the goals of continual learning. Unlike a standard linear unit that responds to any input in a broad half-space defined by its weights, a B-cos unit fires strongly only when an input vector is closely aligned with the neuron's weight vector. This architectural modification has the effect of narrowing a neuron's activation cone, encouraging the model to learn features that are highly selective and task-specific. We hypothesize that this inherent selectivity acts as a protective mechanism: when the model is trained on a new task, features learned for previous tasks are less likely to activate, thereby shielding their weights from

destructive gradient updates introduced by later tasks and that may cause forgetting.

We incorporate the B-cos transform into standard deep learning backbones, such as ResNet [111], resulting in "B-cosified" architectures that retain the advantages of their original structure, while adding the alignment requirements of neuron activation typical of B-cos networks. To support learning over time, we combine these models with a standard rehearsal-based continual learning method, employing a replay buffer that stores a limited number of samples from previous tasks. This setup allows us to rigorously evaluate the architectural contribution of B-cos layers to mitigating forgetting, complementing the memory-based protection offered by experience replay.

To assess the effectiveness of our approach, we conduct experiments on three benchmarks of increasing complexity: Split CIFAR-10 [74], Split Mini-ImageNet [59], and Split ImageNet-100 [61], within a class-incremental learning setting, where task identity is unavailable at inference time. We compare our B-cosified models with traditional replay methods, evaluating them on class-incremental accuracy. Results show that B-cosified networks achieve a significant increase in accuracy, demonstrating superior resistance to catastrophic forgetting, which we hypothesize is due to the structural alignment promoted by the B-cos transform, leading to more stable internal representations. To further investigate the role of architectural choices, we conduct a generalization study, comparing different backbone configurations and quantifying their impact on performance.

Moreover, since each B-cos layer implements an input-dependent linear transform, the network's output can be summarized as a single linear mapping that can be directly visualized as a contribution map for the given input. We therefore analyze how these built-in explanations behave in a continual learning scenario, assessing whether they remain stable and faithful as new tasks are learned.

Our contributions can be summarized as follows:

- We identify feature selectivity as a key architectural principle for mitigating catastrophic forgetting and propose B-cos networks as a direct and effective implementation of this principle.
- We demonstrate empirically that B-cosified ResNets, when combined with a simple replay buffer, significantly outperform standard architectures on challenging class-incremental learning benchmarks.
- We provide an analysis suggesting that the performance gains stem from the B-cos layers' tendency to learn more selective, non-overlapping representations, which are inherently more robust within a sequential training paradigm.

- We investigate interpretability in continual learning by analyzing contribution maps across tasks, combining qualitative evidence with quantitative metrics of temporal stability and faithfulness.

This work opens a promising direction for developing continual learning methods with strong architectural inductive biases. Emphasizing feature-level selectivity, motivated by biological mechanisms that limit representational overlap, can reduce gradient cross-talk across tasks, supporting robust continual learning systems.

3.1 Related work

Architectural inductive biases for continual learning. Our approach is centered on the idea that modifying the intrinsic properties of network layers can create an inductive bias that is beneficial for continual learning. We propose using B-cos networks [110], which replace standard linear transforms with cosine-based operations, encouraging highly selective neurons that fire only for a narrow cone of inputs. This principle of reducing interference by shaping feature representations is shared by other lines of work. For instance, several methods aim to enforce orthogonality between either the features or the parameter updates of different tasks. Approaches like Orthogonal Weights Modification (OWM) [112] project gradients to maintain independence between task solutions, thereby minimizing destructive interference. Similarly, our goal of learning selective features is conceptually related to methods that promote sparsity. By ensuring that any given input only activates a small and specific subset of neurons, methods like SupSup [113] also limit the representational overlap between tasks, effectively shielding inactive neurons from gradient updates.

Unlike methods that require complex gradient manipulations or explicit sparsity regularizers, our B-cos approach instills the desired property of feature selectivity through a simple, local architectural change. To our knowledge, we are the first to analyze and leverage B-cos networks from this perspective, demonstrating that their inherent inductive bias for selective features provides a powerful and elegant mechanism for mitigating catastrophic forgetting in continual learning.

Interpretability in deep neural networks. Interpretability methods aim to make neural networks more transparent by identifying the input evidence that drives model decisions. Post-hoc attribution techniques, including gradient-based methods [114, 115], class-activation approaches such as Grad-CAM [116], and local surrogate models like LIME [117], attempt to explain individual predictions by analyzing gradients or perturbing inputs. However, post-hoc explanations may be weakly faithful to the underlying computation and can

be sensitive to noise or randomization [118]. More recent work has explored inherently interpretable models, including ProtoPNet [119], BagNets [120], and CoDA-Nets [121], which introduce structural constraints so that explanations emerge directly from the forward pass rather than from a post-hoc approximation. Other inherently interpretable approaches rely on explicit intermediate semantics. Concept Bottleneck Models (CBM) [122] constrain predictions to be mediated by human-interpretable concepts. GlanceNets [123] extend this idea by promoting alignment between learned concepts and underlying generative factors, while combining disentangled representation learning with open-set recognition to mitigate concept leakage.

In continual learning, interpretability has been investigated along several independent directions. LwM [124] exploits explanations during training by introducing an attention distillation loss that penalizes changes in attention maps across the task sequence. CLEX [125] instead examines explanation drift in class-incremental learning by comparing how different attribution methods evolve over time on low-resolution benchmarks. ICICLE [126] adopts a prototype-based framework that encourages the reuse of visual concepts as new tasks arrive, reducing interpretability concept drift and making explanations more consistent across tasks. Yu et al. [127] employ CBMs and, to mitigate forgetting, enhance them through semantic-guided prototype augmentation that generates pseudo-features for previously learned classes.

Differently from these approaches, which exploit explanations (or concept/prototype supervision) during training to improve retention over time, our contribution mainly focuses on the architectural inductive bias induced by the B-cos transformation and its impact in continual learning. This study is complemented by an analysis of how the intrinsic interpretability of B-cos Networks evolves over time.

3.2 Method

We frame our investigation within the standard paradigm of continual learning for classification, as detailed in Sec. 2.1. Our work then focuses on rehearsal-based methods, which are a dominant and effective family of approaches for mitigating catastrophic forgetting. These methods augment the training process with a small, fixed-size memory buffer, $\mathcal{M}$, which stores a representative subset of samples from past tasks. When the model F is learning task τ_i, it does not only train on the new data from $\mathcal{D}_i$ but also on samples replayed from the buffer $\mathcal{M}$. After training on task τ_i is complete, the buffer is updated with samples from $\mathcal{D}_i$ to be used in future tasks.

The general training objective at task τ_i is to update the model parameters $\boldsymbol{\theta}$ by minimizing a compound loss function that balances performance on the current task with knowledge

retention of past tasks. This can be expressed generally as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_i}\Big[\mathcal{L}_{\text{task}}(F(\mathbf{x};\boldsymbol{\theta}),y)\Big] + \\ + \mathbb{E}_{(\mathbf{x}_m,y_m)\sim\mathcal{M}}\Big[\mathcal{L}_{\text{replay}}(F(\mathbf{x}_m;\boldsymbol{\theta}),y_m)\Big]\end{aligned} \tag{3.1}$$

where $\mathcal{L}_{\text{task}}$ represents the loss on the new data (e.g., standard cross-entropy), while $\mathcal{L}_{\text{replay}}$ is a loss term applied to the replayed samples from the buffer. The specific formulations of these loss terms vary across different state-of-the-art rehearsal methods; our contribution is however architectural and orthogonal to the specific loss formulation. The ultimate goal is to produce a final model that performs well on the union of all classes seen during training, $\mathcal{Y}_{\text{all}} = \bigcup_{i=1}^{T} \mathcal{Y}_i$.

3.2.1 B-cos networks

At the core of our architectural modification is the B-cos transform, introduced by Böhle et al. [110]. This transform replaces the standard linear operations found in typical neural network layers. To understand its impact, we first consider the standard dot product operation performed by a single neuron with input $\mathbf{x}$ and weight vector $\mathbf{w}$:

$$f(\mathbf{x};\mathbf{w}) = \mathbf{w}^\intercal\mathbf{x} = \|\mathbf{w}\|\,\|\mathbf{x}\|\cos(\angle(\mathbf{x},\mathbf{w})). \tag{3.2}$$

Geometrically, this neuron produces a positive activation for any input $\mathbf{x}$ that lies in the open half-space defined by the hyperplane to which $\mathbf{w}$ is the normal vector, with the specific response depending on the activation function that receives the value $f(\mathbf{x};\mathbf{w})$. This broad activation condition allows features to respond to a wide range of inputs, which can contribute to representational interference between tasks.

The B-cos transform modifies this operation to make activation far more selective. It is defined as:

$$\text{B-cos}(\mathbf{x};\mathbf{w}) = \|\hat{\mathbf{w}}\|\,\|\mathbf{x}\|\,|\cos(\angle(\mathbf{x},\hat{\mathbf{w}}))|^B \cdot \text{sgn}(\cos(\angle(\mathbf{x},\hat{\mathbf{w}}))), \tag{3.3}$$

where $\hat{\mathbf{w}} = \mathbf{w}/\|\mathbf{w}\|$ is the unit-normalized weight vector, $\text{sgn}(\cdot)$ is the sign function, and $B \geq 1$ is the main hyperparameter of the transformation, whose effect will be described shortly. The authors show the expression in Eq. 3.3 is equivalent to a rescaled dot product:

$$\text{B-cos}(\mathbf{x};\mathbf{w}) = \hat{\mathbf{w}}^\intercal\mathbf{x}\,|\cos(\angle(\mathbf{x},\hat{\mathbf{w}}))|^{B-1}. \tag{3.4}$$

The critical component is the scaling factor $|\cos(\angle(\mathbf{x},\hat{\mathbf{w}}))|^{B-1}$. Since the cosine similarity between the input and the normalized weight is always in the range $[-1,1]$, its absolute

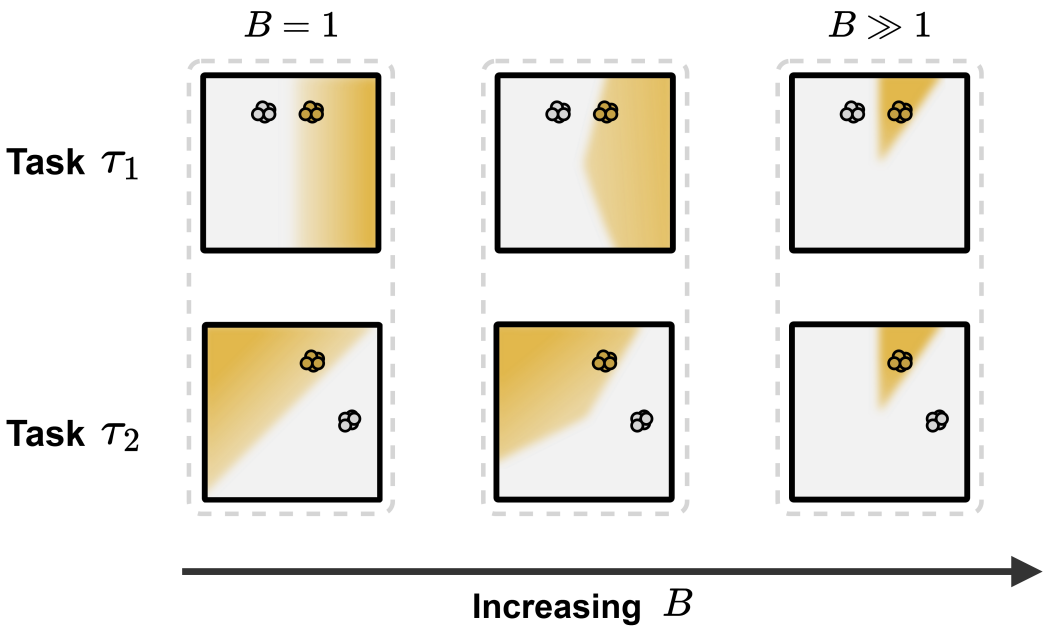

Figure 3.1: **Visualization of the classification tasks for different values of B.** For $B = 1$ (first column), the neuron behaves as a standard linear unit, activating over the positive semi-hyperplane defined by its weight vector. Assuming the yellow cluster is the same in both tasks, the optimal separating direction differs substantially across tasks (compare rows within the same column). As B increases, the neuron's activation hypercone narrows around its weight vector, making activations more selective and the resulting decision regions more consistent across tasks (last column), reducing task-dependent representational interference.

value is in $[0, 1]$. For any non-perfect alignment (i.e., when $|\cos(\angle(\mathbf{x}, \hat{\mathbf{w}}))| < 1$), raising this value to the power of $B - 1$ (for $B > 1$) suppresses the output. As the hyperparameter B increases, this suppression becomes exponentially stronger for inputs that are not closely aligned with the weight vector.

This has a powerful and intuitive geometric effect: it narrows the neuron's activation cone. While a standard linear neuron activates for any input in the positive semi-hyperplane defined by the weights, a B-cos neuron only produces a significant output for inputs that fall within a narrow hypercone around its weight vector $\mathbf{w}$, with the "width" of this cone being inversely controlled by B. By encouraging neurons to learn such highly selective features, the B-cos architecture provides a strong inductive bias against the representational interference that causes catastrophic forgetting. Fig. 3.1 provides a visual interpretation of the effect of B-cos activations.

3.2.2 B-cosified architectures

An advantage of the B-cos transform is its versatility as a drop-in replacement for standard linear operations. This allows us to adapt well-established and powerful architectures, such as ResNet [111] and DenseNet [128] models, for our continual learning experiments. We refer to these adapted models as "B-cosified" architectures. The conversion process, guided by the principles outlined in [110], involves several specific modifications:

- Every standard 2D convolutional layer within the models, including those in the residual blocks, bottleneck layers, and downsampling paths of ResNet, as well as the dense blocks and transition layers of DenseNet, is replaced with a B-cos convolutional layer. The final fully connected classification layer is also replaced with a 1×1 B-cos convolutional layer, which acts as a convolutional classifier.
- The B-cos transform with $B > 1$ is inherently non-linear, which enables the removal of standard non-linear activation functions, most notably ReLU. This leads to a desirable explainability property, that is the possibility of collapsing the network's computation into a single linear transform. However, it is still possible to add non-linearities: specifically, the authors of [110] employ MaxOut [129] activations, which act as "conditional" linear paths, increasing model capacity while keeping explainability properties. In our implementation, we also adopt MaxOut, setting the number of units $k = 2$.
- Batch normalization is removed, as it would interfere with the magnitude of the feature vectors passed between layers, disrupting the scaling mechanism of the B-cos transform.
- In the initial feature extraction block of the standard DenseNet-121 architecture, the max-pooling layer is replaced with an average pooling, to ensure that the transformations remain smoother and less aggressive.

Essentially, while the high-level structure, such as the residual connections in ResNet or the feature concatenation in DenseNet, is preserved to benefit from their advantages in gradient flow and feature reuse, the underlying operations are entirely replaced to promote feature selectivity based on B-cos transform principles.

3.3 Experimental results

3.3.1 Datasets and metrics

Our experiments consider three continual learning benchmarks obtained by splitting standard image classification datasets into a sequence of disjoint tasks: Split CIFAR-10 [74], Split

Mini-ImageNet [59], and Split ImageNet-100 [130].

To evaluate performance, we employ two standard continual learning metrics: Final Average Accuracy (FAA) and Final Average Forgetting (FAF). Since lower FAF indicates better knowledge retention and greater resistance to forgetting, it is relevant for evaluating our hypothesis that B-cos architectures promote more stable and protected representations. Further details on these benchmarks and metrics are provided in Sec. 2.3 and Sec. 2.4, respectively.

3.3.2 Baselines

We assess our architectural proposal by integrating it into a rehearsal-based method and comparing its performance against a set of well-established baselines: Experience Replay (ER) [131], DER++ [96], ER-ACE [103]. All methods use the same memory buffer size for a fair comparison. While we report the results of all three methods as baselines, we specifically choose to integrate B-cos layers with ER-ACE, due to its stronger performance (see Tables 3.1, 3.2, and 3.3 later in this section).

3.3.3 Training procedure

For our experiments, we use ResNet-18, ResNet-50 [111], and DenseNet-121 [128] backbones. Training is carried out for 50 epochs per task, using the Adam optimizer [132] initialized with a learning rate of 0.001. Batch size is set to 32 for Split CIFAR-10 and to 8 for all other datasets. For all B-cos models, we vary the hyperparameter B between 1.4 and 2, as we empirically found this range to provide the most promising results. Given the lower number of classes in Split CIFAR-10, we adopt smaller buffer sizes of 70, 100, and 200 samples. For Split Mini-ImageNet and Split ImageNet-100, we evaluate larger buffers of 200, 500, and 1000 samples to accommodate their increased complexity.

All results are obtained in the class-incremental setting and reported as mean and standard deviation over three runs for both metrics.

3.3.4 Results

We hereby discuss our results, analyzing the performance of the compared methods separately for each dataset, with each table reporting both FAA and FAF metrics. Tables 3.1, 3.2, and 3.3 present results for Split CIFAR-10, Split Mini-ImageNet, and Split ImageNet-100, respectively.

	FAA (↑)			FAF (↓)		
Buffer size	**70**	**100**	**200**	**70**	**100**	**200**
ER [103]	35.64±1.96	41.53±1.39	48.26±2.47	68.87±2.04	60.42±3.40	53.08±2.27
DER++ [96]	53.80±2.85	56.77±2.55	62.62±1.66	43.64±3.00	36.64±1.90	29.38±1.32
ER-ACE [131]	46.06±1.97	51.48±3.92	59.73±2.18	**25.77**±4.23	**22.70**±1.35	**14.01**±1.23
↪ **B=1.40**	51.64±2.86	51.82±3.28	60.89±5.40	38.69±3.38	35.22±3.31	27.36±0.56
↪ **B=1.50**	54.45±2.97	**59.02**±2.32	**66.25**±0.91	36.53±4.52	30.92±0.81	25.01±0.83
↪ **B=1.60**	**55.03**±1.74	58.25±1.33	63.66±1.65	39.09±1.71	33.90±1.56	27.19±1.35
↪ **B=1.70**	54.72±1.68	57.19±1.11	63.41±1.12	39.94±1.64	35.79±0.90	29.11±0.91
↪ **B=1.80**	51.54±1.92	55.74±1.85	62.32±1.05	39.19±4.70	34.68±1.52	26.33±0.39
↪ **B=1.90**	50.89±2.09	56.00±1.26	62.25±0.53	40.34±3.13	36.14±1.98	32.03±0.80
↪ **B=2.00**	49.95±1.94	53.21±1.05	60.51±2.69	41.09±2.38	35.55±2.05	32.49±1.23

Table 3.1: **Results on Split CIFAR-10 in class-incremental learning with varying buffer sizes.** We report *final average accuracy* (FAA ↑) and *final average forgetting* (FAF ↓). We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column.

Across all datasets and buffer sizes, integrating B-cos layers yields consistent and significant improvements over the standard rehearsal baselines. On Split CIFAR-10 with a buffer of 200, our best B-cos model achieves an accuracy of 66.25%, substantially outperforming both ER-ACE (59.73%) and the strong DER++ baseline (62.62%). This trend is amplified on the more difficult datasets: on Split Mini-ImageNet (buffer size 1000), our top-performing model reaches 28.43%, again surpassing ER-ACE (18.80%) and DER++ (14.81%) by a large margin.

The performance gains are particularly remarkable on Split ImageNet-100, our most challenging benchmark. With a buffer size of 1000, our B-cos model achieves 27.40% accuracy, more than doubling the performance of DER++ (10.33%) and delivering a relative improvement of over 50% against ER-ACE (17.79%). This demonstrates that the architectural pressure toward feature selectivity is especially crucial in fine-grained scenarios with high inter-class similarity, where representational overlap is a major contributor to forgetting.

Our experiments also reveal the sensitivity to the hyperparameter B. The best results are consistently achieved with moderate values, typically between $B = 1.4$ and $B = 1.8$. This suggests an important trade-off: higher values of B enforce stricter feature selectivity, which helps mitigate forgetting, but excessive values ($B \geq 2$) can overly regularize the model, hindering its plasticity and ability to learn new concepts effectively. Furthermore, the performance gains from using B-cos layers are generally more pronounced as the buffer size

	FAA (↑)			FAF (↓)		
Buffer size	**200**	**500**	**1000**	**200**	**500**	**1000**
ER [103]	4.97±0.28	5.76±0.43	7.53±0.76	86.21±0.29	84.27±0.36	82.97±0.34
DER++ [96]	9.65±1.15	12.67±1.32	14.81±8.60	68.99±5.47	64.66±3.15	58.94±7.42
ER-ACE [131]	12.22±0.71	16.27±0.46	18.80±0.35	43.13±1.90	45.39±0.70	46.72±0.50
↪ **B=1.40**	13.19±0.86	**20.69**±1.08	25.94±3.00	42.00±4.23	**35.66**±0.78	**36.98**±4.65
↪ **B=1.50**	13.27±0.71	19.95±1.50	27.68±1.94	**40.31**±1.23	37.78±2.29	37.12±4.66
↪ **B=1.60**	**13.85**±0.76	20.42±0.64	25.15±3.20	43.35±2.42	39.39±2.80	40.46±6.07
↪ **B=1.70**	13.04±1.13	20.03±0.52	28.15±0.89	42.33±0.15	44.45±0.93	36.61±0.95
↪ **B=1.80**	12.54±0.44	20.33±0.97	**28.43**±2.11	38.55±2.68	44.32±1.38	39.30±0.59
↪ **B=1.90**	12.65±0.51	19.39±0.45	26.18±0.72	44.56±2.69	44.75±1.36	38.89±0.97
↪ **B=2.00**	11.39±0.85	20.25±1.27	25.89±1.00	48.15±5.00	42.95±2.01	41.08±0.96

Table 3.2: **Results on Split Mini-ImageNet in class-incremental learning with varying buffer sizes.** We report *final average accuracy* (FAA ↑) and *final average forgetting* (FAF ↓). We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column.

increases, indicating that our architectural approach scales well and effectively complements the information provided by a larger replay capacity.

Next, we analyze the results in terms of *final average forgetting*, providing a direct measure of how well each method preserves past knowledge, with a lower value indicating less forgetting. The results reveal a nuanced story that reinforces our hypothesis. On the more complex Split Mini-ImageNet and Split ImageNet-100 datasets, B-cosified models consistently demonstrate lower forgetting than the ER-ACE baseline, especially with larger buffer sizes. For example, on Split Mini-ImageNet with a buffer size of 1000, the best B-cos model achieves an FAF of 36.98%, a marked improvement over ER-ACE's 46.72%. This directly supports our claim that the feature selectivity induced by B-cos layers provides a protective effect against the representational interference common in complex, multi-task scenarios.

Interestingly, on the simpler Split CIFAR-10 benchmark, the B-cos models exhibit slightly higher forgetting than ER-ACE, despite achieving higher final accuracy. A possible explanation of this behavior could be due to the low resolution of CIFAR-10 images (32×32), which may inherently limit the degree of feature specialization possible. The lack of fine-grained detail might force the network to learn features that are more general, reducing the effectiveness of the B-cos selectivity mechanism. Further investigation of this phenomenon is needed to better understand its causes. Nevertheless, the substantial reduction in forget-

	FAA (↑)			FAF (↓)		
Buffer size	**200**	**500**	**1000**	**200**	**500**	**1000**
ER [103]	4.34±0.14	4.50±0.06	4.72±0.07	63.89±0.37	63.14±0.17	63.88±0.79
DER++ [96]	6.20±0.98	8.69±1.04	10.33±1.61	60.68±7.58	63.89±2.00	59.76±5.44
ER-ACE [131]	10.39±0.51	14.38±1.45	17.79±0.58	43.81±1.71	47.77±2.26	50.44±2.12
↪ **B=1.40**	**13.47**±0.33	20.15±1.63	**27.40**±2.60	44.93±2.36	46.32±5.84	43.62±7.74
↪ **B=1.50**	12.25±1.34	**22.09**±1.63	25.69±2.62	44.58±1.43	44.73±5.53	43.26±1.17
↪ **B=1.60**	11.18±0.49	19.51±1.87	24.12±2.44	53.63±4.72	46.35±4.35	41.82±1.46
↪ **B=1.70**	12.07±1.35	19.09±0.31	26.48±2.18	**41.84**±4.80	45.77±1.91	**40.31**±1.11
↪ **B=1.80**	10.33±0.38	17.09±2.00	23.07±1.42	49.23±3.25	45.47±2.03	43.28±0.53
↪ **B=1.90**	11.31±0.99	17.62±0.43	24.93±0.04	44.38±3.25	**44.39**±3.46	43.58±0.92
↪ **B=2.00**	10.55±0.16	16.46±0.43	24.01±0.20	49.18±2.36	49.61±2.32	44.80±1.40

Table 3.3: **Results on Split ImageNet-100 in class-incremental learning with varying buffer sizes.** We report *final average accuracy* (FAA ↑) and *final average forgetting* (FAF ↓). We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column.

ting on more challenging, fine-grained datasets confirms that the architectural bias is most impactful on more complex tasks, which is a desirable property that makes the approach more suitable for real-world problems.

3.3.5 Generalization analysis

To assess how B-cosification generalizes across architectural backbones, we extend our evaluation beyond the ResNet-18 backbone used in Sec. 3.3.4. In particular, we consider two widely adopted, higher-capacity convolutional networks: ResNet-50 [111] and DenseNet-121 [128]. These models introduce substantially higher representational capacity and depth, allowing us to assess the robustness of our method under increased architectural complexity. The ER-ACE model [131], identified as the top-performing method when combined with our approach, is used as the baseline model for this study. All experiments are performed on the Split Mini-ImageNet dataset [59].

With a ResNet-50 backbone, B-cosification consistently improves upon the standard ER-ACE model. For a buffer size of 500, the best B-cos model achieves a final accuracy of 13.05%, a significant relative improvement compared to the 10.29% from its standard counterpart. This advantage becomes more pronounced with a buffer size of 1000, where the best configuration achieves 16.33% accuracy versus 12.95% for the standard ER-ACE. The

	ResNet-50			DenseNet-121		
Buffer size	**200**	**500**	**1000**	**200**	**500**	**1000**
ER [103]	4.38±0.08	5.40±0.60	5.69±0.35	5.56±0.03	7.81±0.46	9.74±0.08
DER++ [96]	6.98±1.53	7.86±1.79	6.13±0.21	7.66±0.94	9.86±1.29	11.66±0.42
ER-ACE [131]	8.12±0.31	10.29±0.86	12.95±0.06	8.24±1.34	13.75±0.74	19.49±1.27
↪ **B=1.40**	8.61±0.09	10.93±1.32	14.93±0.20	9.59±0.16	14.84±0.83	19.51±0.52
↪ **B=1.50**	**9.67**±0.18	12.48±0.09	15.47±0.21	9.15±0.66	**15.34**±0.29	**20.14**±0.38
↪ **B=1.60**	9.01±0.48	12.66±0.77	**16.33**±0.05	9.32±0.13	14.38±0.67	19.38±0.15
↪ **B=1.70**	8.85±0.23	12.80±0.74	15.80±0.66	9.20±0.38	14.39±0.38	17.48±0.44
↪ **B=1.80**	9.53±0.47	**13.05**±0.04	15.94±1.34	**9.79**±0.25	14.58±0.16	17.66±1.05
↪ **B=1.90**	8.69±0.46	12.76±1.19	15.80±0.33	8.90±0.83	13.86±0.30	17.39±0.51
↪ **B=2.00**	8.04±0.23	12.22±0.18	14.24±1.32	8.79±0.37	13.81±0.42	16.50±0.92

Table 3.4: **Results on Split Mini-ImageNet in class-incremental learning using ResNet-50 and DenseNet-121 as backbones.** We report *final average accuracy* (FAA ↑) for different buffer sizes. We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column.

same trend is observed for buffer sizes of 200 and 500 with the DenseNet-121 architecture, which relies on a fundamentally different feature-reuse mechanism (dense concatenation). For a buffer size of 1000, DenseNet-121 exhibits a more nuanced behavior. While the best B-cos model still achieves the top result with 20.14% accuracy, accuracy drops below the ER-ACE baseline for $B \geq 1.60$. A plausible explanation is that, when the rehearsal memory is large, ER-ACE already enforces a strong stability signal, reducing the marginal gains from additional feature selectivity. In this setting, increasing B too much can over-constrain representations, undermining within-task learning. This effect may be amplified in DenseNet-121 due to its reliance on feature reuse via dense concatenation: overly selective activations can make shared features less broadly useful across layers, underutilizing model capacity and resulting in a drop in accuracy. Overall, performance gains are observed for values of the B-cosification parameter B within the range from $B = 1.40$ to $B = 1.60$, with DenseNet-121 at larger buffer sizes favoring the lower end of this range. The consistent improvement across these varied and deeper architectures provides evidence that instilling the inductive bias for feature selectivity via B-cos layers is a robust and architecture-agnostic principle for improving continual learning performance.

A noteworthy, and perhaps counter-intuitive, observation is that the absolute performance on Split Mini-ImageNet is higher with the shallower ResNet-18 than with the deeper

ResNet-50 and DenseNet-121 backbones. This holds true for both the standard ER-ACE baseline and our B-cosified models. For instance, the best B-cos ResNet-18 with a buffer size of 500 reaches 20.69% accuracy, while the best B-cos ResNet-50 and DenseNet-121 top out at 13.05% and 15.34% respectively. This phenomenon highlights a known challenge in continual learning: larger, higher-capacity models are often more prone to catastrophic forgetting, as their larger parameter space is more difficult to regularize effectively, especially with the limited information provided by a small memory buffer. On top of this, complex models are more likely to suffer from overfitting on small tasks. The consistent relative improvement provided by B-cos across all these architectures reinforces the claim that our architectural approach provides a robust benefit, independent of model depth.

3.4 Interpretability

A key advantage of B-cos networks is that their design enables model-inherent explanations that are directly tied to the model's computation, rather than relying on post-hoc surrogates. In this section, we describe how these explanations are computed and then analyze their behavior and implications in a continual learning context.

3.4.1 Explanations

Let $\mathbf{W}_j$ denote the weight matrix of layer j, with the k-th row given by the weight vector of the k-th neuron. Following [110], a B-cos layer computes an input-dependent linear transform:

$$\widetilde{\mathbf{W}}_j(\mathbf{a}_j) = \left|\cos(\mathbf{a}_j; \widehat{\mathbf{W}}_j)\right|^{B-1} \odot \widehat{\mathbf{W}}_j, \tag{3.5}$$

where $\widehat{\mathbf{W}_j}$ denotes $\mathbf{W}_j$ with each row scaled to unit norm, and $\odot$ denotes element-wise multiplication (with row-wise broadcasting when applied to a vector and a matrix).

The output of a B-cos network can be expressed as a sequence of input-dependent linear transforms applied to the input $\mathbf{x}$:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \widetilde{\mathbf{W}}_L(\mathbf{a}_L)\, \widetilde{\mathbf{W}}_{L-1}(\mathbf{a}_{L-1}) \cdots \widetilde{\mathbf{W}}_1(\mathbf{a}_1 = \mathbf{x})\, \mathbf{x}. \tag{3.6}$$

Such a formulation allows the network to be rewritten as a single linear mapping of the input. In particular, the model output is given by

$$f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{W}_{1\to L}(\mathbf{x})\, \mathbf{x}, \tag{3.7}$$

where $\mathbf{W}_{1\to L}(\mathbf{x})$ summarizes the sequence of layer-wise transforms applied to $\mathbf{x}$. Under this formulation, the class-c logit can be written as the inner product between the input $\mathbf{x}$ and the corresponding class-specific weight vector. Accordingly, the c-th row of $\mathbf{W}_{1\to L}(\mathbf{x})$ indicates which input patterns contribute positively or negatively to the prediction for class c.

The resulting spatial *contribution maps* can be used to quantitatively evaluate the explanations. For a target neuron indexed by j at layer l, we compute the input contributions as

$$\mathbf{s}_j^l(\mathbf{x}) = [\mathbf{W}_{1\to l}(\mathbf{x})]_j^\top \odot \mathbf{x}. \tag{3.8}$$

For image inputs, the contribution from a single pixel location (x, y) is obtained by summing over color channels k, i.e., $\sum_k [\mathbf{s}_j^l(\mathbf{x})]_{(x,y,k)}$.

3.4.2 Qualitative results

To analyze interpretability in a continual learning setting, we qualitatively inspect how explanations evolve across the sequence of tasks. Specifically, we compare the proposed B-cos contribution maps against Grad-CAM explanations [116] for standard rehearsal-based baselines. We assume that, when the model effectively learns a target class in the task in which it is introduced, its explanations should highlight the most discriminative visual cues for recognizing that class. Under successful knowledge retention, these highlighted regions are expected to remain stable over time; conversely, substantial drift or degradation in the maps may signal interference and the loss of class-relevant representations.

Given an example image taken from the first task of Split Mini-ImageNet, Figure 3.2 shows the corresponding explanation maps computed for different values of B and visualized throughout the subsequent task sequence. Over time, Grad-CAM maps for ER and DER++ progressively lose localization, either becoming broadly diffuse or exhibiting frequent shifts, whereas ER-ACE maps instead remain more object-focused while still showing a soft relevance spillover into the surrounding context. In contrast, the behavior of B-cos explanations varies with the value of B. For lower values of B, despite higher accuracy (as shown in Sec. 3.3.4), the narrower neuronal activation cone leads to contribution maps that are granular and sparse, and therefore harder to interpret. Increasing B widens the activation cone and produces more coherent and stable maps, improving readability but reducing overall accuracy.

Since B-cos explanations for $B = 2$ are the most informative, we provide additional qualitative comparisons with Grad-CAM for ER-ACE, the strongest rehearsal baseline in our setting (Fig. 3.3). Like the other baselines, ER-ACE Grad-CAM maps can still drift and spread to contextual regions over time. We also observe that deviations from the more

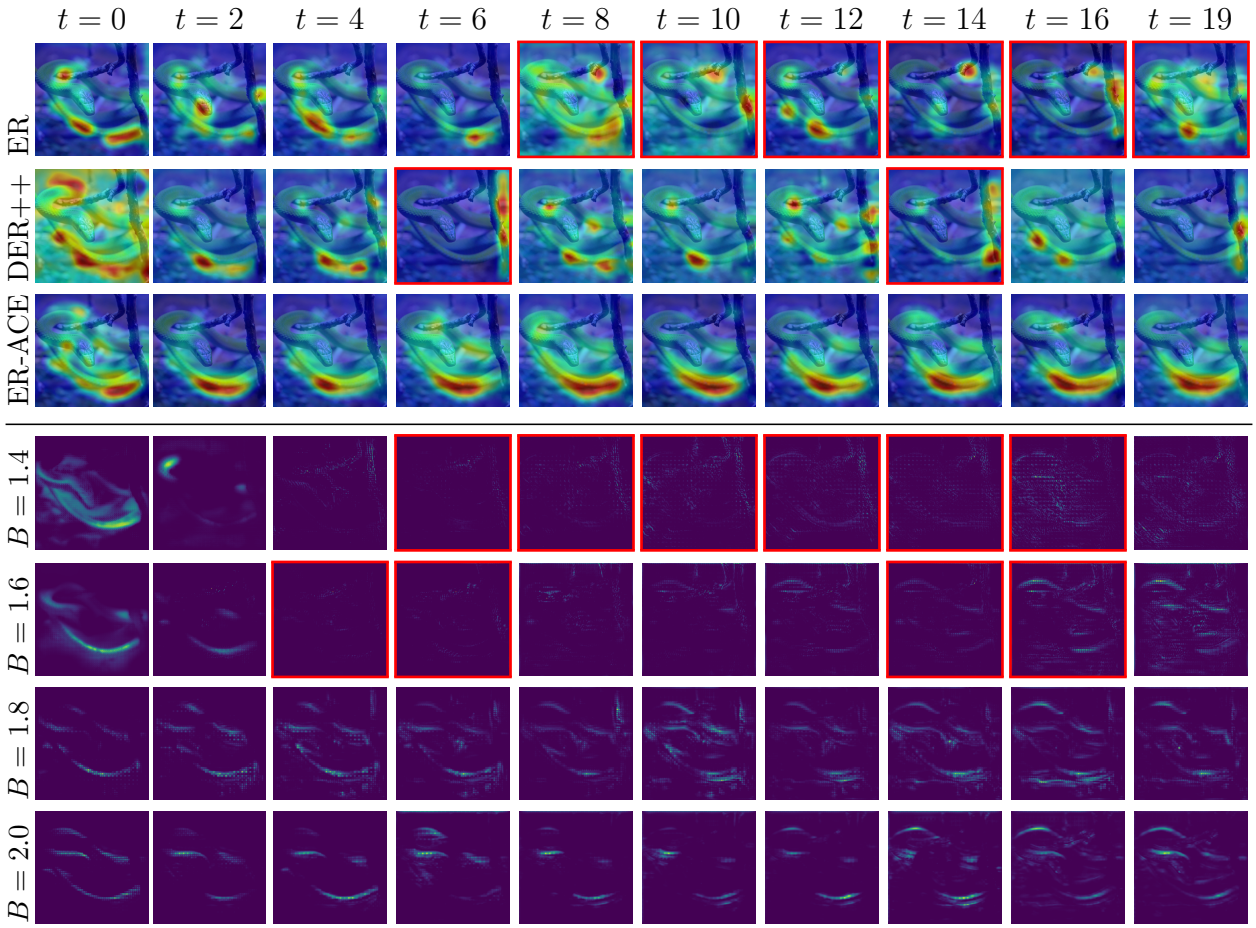

Figure 3.2: **Explanation degradation across tasks.** Comparison of Grad-CAM maps for ER, DER++, and ER-ACE (top) and B-cos explanations for different values of B (bottom) over the task sequence. Red borders indicate instances in which the model produces an incorrect prediction.

informative B-cos patterns seen in earlier tasks often co-occur with performance drops and misclassifications.

Figure 3.4 shows examples from later tasks, together with the model's outputs on the same inputs in earlier tasks. As expected, for classes not yet seen during training (orange borders), explanations are largely uninformative, suggesting that the attribution signal is tied to learned representations rather than spurious patterns.

3.4.3 Quantitative results

We report quantitative results aimed at characterizing the *temporal stability* of explanation maps as new tasks are learned. To this end, we employ three complementary similarity metrics that capture different aspects of agreement between explanation maps: *Pearson*

	PCC	IoU	Dice	Insertion
ER [103]	0.434±0.052	0.308±0.029	0.461±0.033	0.611±0.061
DER++ [96]	0.396±0.080	0.334±0.038	0.486±0.041	0.232±0.062
ER-ACE [131]	0.541±0.065	0.367±0.033	0.521±0.035	0.540±0.093
↪ **B=1.40**	0.489±0.044	0.339±0.023	0.502±0.028	0.877±0.035
↪ **B=1.50**	0.517±0.035	0.337±0.019	0.501±0.021	**0.945**±0.008
↪ **B=1.60**	0.688±0.060	0.409±0.025	0.576±0.027	0.770±0.060
↪ **B=1.70**	0.740±0.029	0.476±0.020	0.641±0.020	0.777±0.036
↪ **B=1.80**	0.673±0.041	0.456±0.026	0.619±0.025	0.791±0.039
↪ **B=1.90**	0.725±0.029	**0.515**±0.024	**0.673**±0.022	0.810±0.038
↪ **B=2.00**	**0.795**±0.044	0.493±0.027	0.653±0.027	0.753±0.048

Table 3.5: **Interpretability evaluation results on Split Mini-ImageNet in class-incremental learning with buffer size of 500.** We compare standard experience replay methods with our B-cosified models using different values of the B parameter. Bold values indicate the best performance in each column.

Correlation Coefficient (PCC), *Intersection over Union* (IoU), and *Dice coefficient* (Dice). PCC measures the linear agreement between explanation maps across time: higher values indicate that the overall relevance pattern is preserved, whereas lower values suggest substantial drift. For each image, we compute these metrics across consecutive tasks by comparing the current explanation to the corresponding one from the previous task, which serves as the reference, and averaging the resulting scores over the task sequence. To focus on the most discriminative evidence, IoU and Dice are computed on binarized maps obtained by retaining the top-20% most relevant pixels. We report both metrics since IoU is union-normalized and is therefore more conservative, while Dice behaves as an F1-like measure and is typically less punitive under partial overlap.

Beyond stability, we assess the *faithfulness* of the produced explanations, i.e., whether highlighted regions are causally relevant for the prediction, via the *Insertion* metric [133]. Starting from a zero-initialized reference image, we construct a sequence of partially revealed inputs by inserting pixels in decreasing order of relevance induced by the explanation map (from most to least relevant), and we record the target-class score after each insertion step. The area under the resulting insertion curve (AUC) is reported, where higher values indicate that the ranking identifies evidence that is causally aligned with the model's prediction. The metric is computed using the model after the last task, to test whether evidence highlighted at earlier tasks remains predictive after learning the full task sequence.

All metrics are computed *per image*, then averaged within each class and finally across

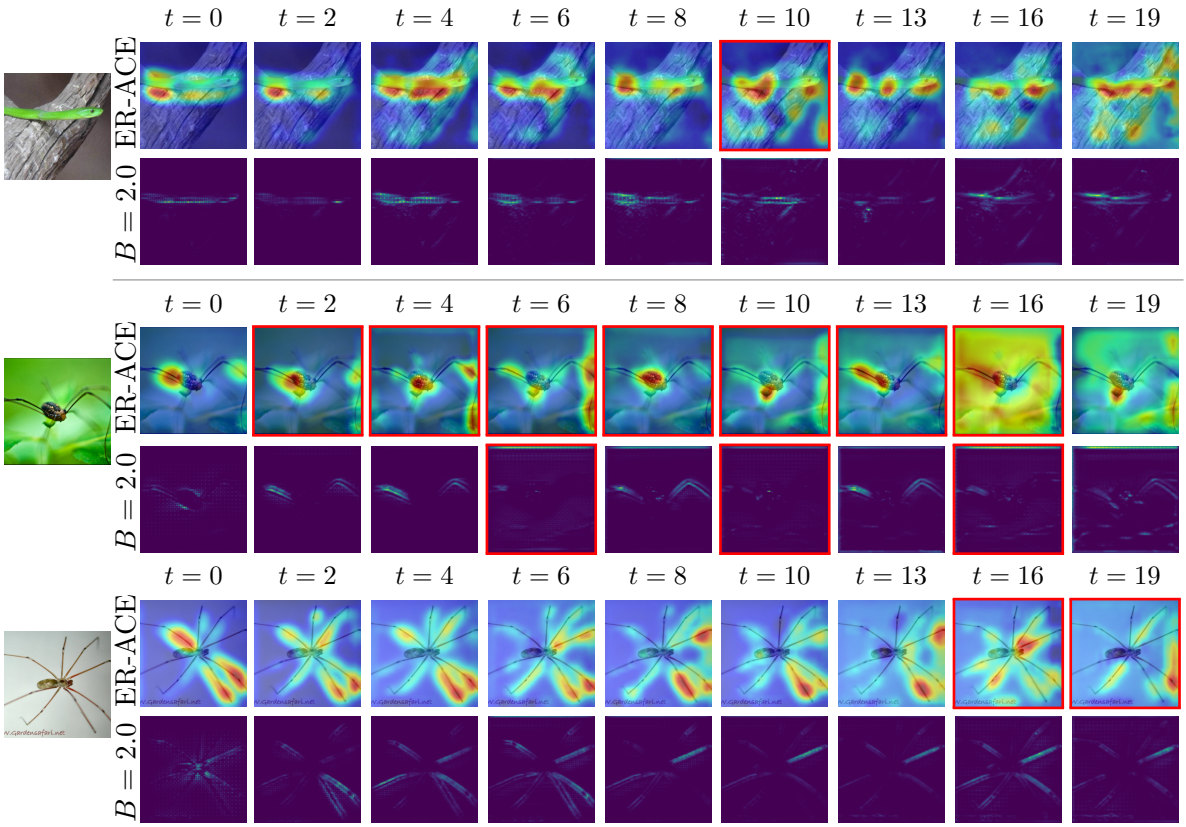

Figure 3.3: **Explanation degradation across tasks using images from the first task.** Comparison of Grad-CAM maps for ER-ACE and B-cos explanations for $B = 2$ over the task sequence. Red borders indicate instances in which the model produces an incorrect prediction.

classes. As reported in Table 3.5, B-cosification preserves the spatial attribution pattern more reliably over time than rehearsal-based baselines. Interestingly, the configuration that maximizes FAA ($B = 1.40$) is not the most stable: it attains the lowest stability overall, below all other B settings and ER-ACE. This is consistent with the effect of B: smaller values increase plasticity and can improve end-of-stream performance, but they also allow stronger task-to-task reorganization, leading to larger attribution drift. Increasing B progressively mitigates this effect, improving inter-task agreement with only marginal FAA differences.

Faithfulness follows the opposite trend within the B-cos family, as insertion generally peaks at smaller B. This suggests more causally effective relevance rankings, even though temporal stability is reduced. Nevertheless, all B-cos settings outperform replay baselines in insertion AUC, suggesting improved causal alignment between attributions and the model decision. Overall, these results point to a trade-off between maximizing temporal consistency (larger B) and maximizing the causal effectiveness of the relevance ranking (smaller B).

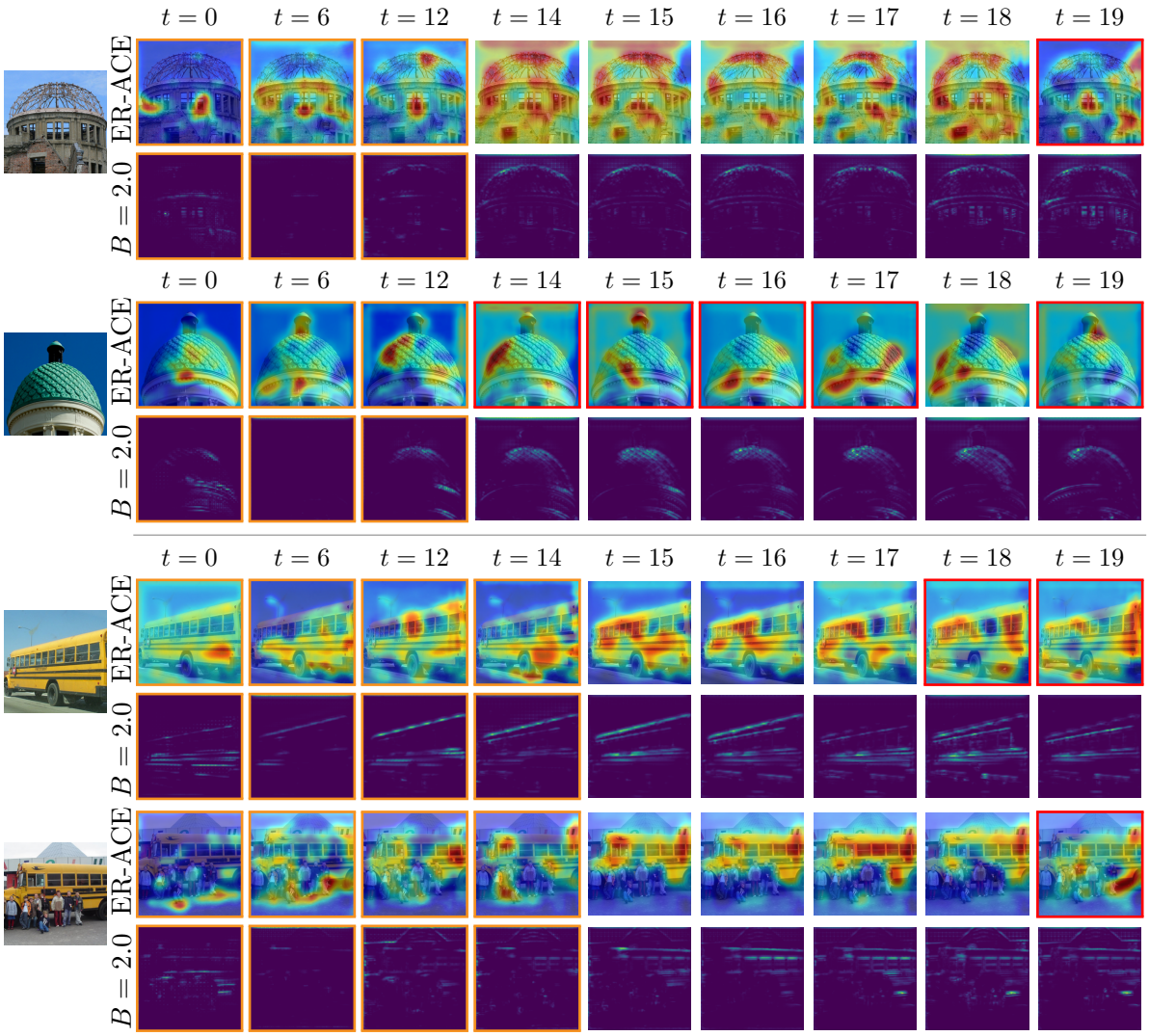

Figure 3.4: **Explanation degradation across tasks using images from later tasks in the sequence.** Comparison of Grad-CAM maps for ER-ACE and B-cos explanations for $B = 2$ over the task sequence. Red borders indicate instances in which the model produces an incorrect prediction, while orange borders indicate instances for which the model has not been trained on the corresponding task yet.

3.5 Discussion

In this chapter, we address catastrophic forgetting by drawing inspiration from brain-inspired mechanisms such as pattern separation and neural coding. By replacing standard linear layers with a cosine-similarity transform, our models develop narrow activation cones that foster feature selectivity and reduce destructive interference across tasks. When integrated into ResNet backbones and combined with a simple rehearsal buffer, these "B-cosified" architectures consistently outperform their vanilla counterparts on class-incremental benchmarks

of varying complexity, demonstrating a marked improvement in both average accuracy and forgetting metrics.

Our empirical analysis shows that, besides memory replay, the performance gains also derive from the architectural bias introduced by B-cos layers: by learning highly aligned, task-specific representations, the network protects previously acquired knowledge from being overwritten during subsequent training. The generalization analysis further confirms that the selectivity induced by the B-cos transform is a critical driver of this effect, with greater alignment yielding more stable class boundaries over time.

Beyond accuracy, we studied the impact of continual learning on the inherent interpretability of B-cos Networks. Both qualitative evidence and quantitative metrics reveal a systematic trade-off, guided by the choice of B, between predictive performance and the temporal consistency of explanations. Smaller B increases plasticity and often improves final average accuracy, but leads to highly selective yet fragmented contribution maps; conversely, larger B produces more coherent and temporally stable explanations, at a moderate cost in predictive performance.

Looking forward, we envisage several directions for further investigation. First, extending B-cos networks to other backbone families (e.g., Vision Transformers) and continual learning paradigms (e.g., task-agnostic or unsupervised settings) could further validate the generality of our approach. Second, a deeper theoretical investigation into the relationship between angular margin, feature sparsity, and forgetting could yield refined transforms with tunable selectivity. On a related note, it would be interesting to better study the effect of hyperparameter B: our results show that a suitable choice of B for a specific combination of dataset and buffer size is necessary in order to achieve the best performance. For this reason, we intend to investigate the causes of this phenomenon, so that a principled choice of B can be made for specific use cases. Finally, combining B-cos layers with complementary strategies, such as dynamic expansion or parameter isolation, may lead to even greater resilience in lifelong learning systems.

3.6 Publications

Sorrenti, A., Bellitto, G., Calcagno, S., Hendrix, R., Spampinato, C., & Palazzo, S. (2025). "B-cos Networks as an Architectural Inductive Bias for Mitigating Catastrophic Forgetting". Workshop of Continual Learning in Computer Vision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (pp. 3550-3559), Workshop on Continual Learning in Computer Vision.

Chapter 4

Selective Freezing for Efficient Continual Learning

Considering the environmental and resource implications of training large-scale deep learning models, developing learning strategies that are both effective and resource-efficient has become an increasingly important priority for the research community. State-of-the-art architectures, particularly those operating at the frontier of performance across multiple domains, often require substantial computational resources for training. As these demands increase, so does the associated energy consumption, leading to a significant environmental footprint [134]. Therefore, identifying ways to reduce the computational requirements without a substantial performance loss has become an imperative.

In this chapter, we draw inspiration from neuroscientific accounts of synaptic consolidation, which emphasize that learning unfolds across multiple timescales and that stability emerges by progressively constraining plasticity for representations that have proven reliable over experience [30, 31, 32]. Building on this perspective, we first present an exhaustive empirical analysis of the effect of layer freezing in a continual learning context. Our results reveal that there exist configurations that allow us to freeze a significant portion of the model without suffering a notable decrease in accuracy. This shows that a substantial portion of the representations learned on earlier tasks remains reusable over time, enabling transfer across tasks without requiring all layers to be continually updated. Motivated by these findings, we propose a method designed to address the twin challenges of continual learning and computational efficiency. Our approach leverages the intuition that suitable freezing of layers in deep learning models during training can support both objectives, by providing a mechanism to preserve important features from previous tasks, while reducing the amount of computation required, which directly affects the energy impact of model training. In detail, at the beginning of each continual learning task at training time, we perform a fast adaptation stage of

the model on the new task, by testing different freezing strategies to dynamically identify a subset of layers that optimizes plasticity on the new task and stability on the previous ones. Experimental results show that our approach achieves performance that is competitive with manually-tuned optimal freezing strategies. This eliminates the need for time-consuming and expertise-intensive hyperparameter tuning, further augmenting the computational efficiency of our approach. Additionally, we show that our freezing strategy effectively reduces the amount of computation and energy requirements, which we quantitatively estimate in terms of number of parameters and number of updates required to train a model.

The results presented herein may serve as a step towards more sustainable and efficient deep learning, suggesting that the regulation of plasticity inspired by multi-timescale synaptic consolidation in the human brain can preserve the benefits of continual learning. This underscores the value of the brain as a principled source of inspiration for designing machine learning systems that are both environmentally responsible and effective.

4.1 Related work

Several approaches in the continual learning literature have investigated strategies that rely on freezing subsets of network parameters. Once a task has been learned, parameter freezing allows the acquired knowledge to remain fixed, thus reducing forgetting of previous tasks. Most works have focused on defining *mask-based* methods, where a learned mask for individual weights or groups of weights is used to selectively freeze or constrain certain parameters. Piggyback [135] uses a binary mask on the weights of a pre-trained model to create different sub-networks, introducing an overhead of 1 bit per network parameter for each task. Similarly, Kang et al. [136] introduced Winning SubNetworks (WSN), which sequentially learns and selects an optimal subnetwork for each task by means of an *accumulated binary mask*. WSN updates only weights that have not been selected in previous tasks, resulting in a task-specific subnetwork. HAT [79] learns near-binary attention vectors by using gated task embeddings for each task. These vectors are then used to define hard attention masks for each task, which are used to constrain the network's weight updates for the following task. Masana et al. [137] proposed a ternary mask-based approach for the task-incremental learning scenario. Differently from the above-mentioned approaches, these masks are applied to the features of each layer rather than weights, reducing the number of mask parameters for each new task. Sparse-MAML [138] proposes a meta-learning approach, where a subset of weights is frozen during the inner-loop learning process. Similarly to the previous methods, a binary mask is learned and then multiplied element-wise with gradient updates. The PathNet algorithm [139] employs agents embedded in the neural network to identify which

parts can be reused for new tasks. Task-relevant paths that evolved during the previous tasks can be frozen or partially reused in the following tasks. Shi et al. [140] introduced the BLIP approach, which preserves the information gain on model parameters provided by each task through a guided bit freezing. In particular, weight quantization is exploited to determine the weights to be frozen to prevent forgetting. Jung et al. [141] defined an algorithm that prevents the model from drifting by freezing the weights associated with the information learned in previous tasks, called *nodes*, while learning future tasks. Specifically, the loss function involves two group sparsity-based regularization terms that are used to define the importance of a node for carrying out the preceding tasks. Yang et al. [142] proposed a progressive task-correlated layer freezing method to be used in the context of self-supervised continual learning (SSCL). More specifically, a task correlation ratio, based on the gradient projection norm, is defined to formally characterize the correlation between current and previous tasks.

4.2 Method

Following the formulation described in Sec. 2.1, we frame our setting as a supervised classification problem on a non-i.i.d. stream of data $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, under the assumption that *task boundaries*, which mark changes in the data distribution, are known at training time. Given a classifier $F : \mathcal{X} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta}$, the training objective is to minimize a classification loss over the sequence of tasks, such that the final model achieves high accuracy on both current and past tasks.

The classification model may also keep a limited *memory buffer* $\mathcal{M}$ of past samples, to reduce forgetting of features from previous tasks. The model update step between tasks can be summarized as:

$$\langle F, \boldsymbol{\theta}_{i-1}, \mathcal{M}_{i-1} \rangle \xrightarrow{\mathcal{D}_i} \langle F, \boldsymbol{\theta}_i, \mathcal{M}_i \rangle, \tag{4.1}$$

where $\boldsymbol{\theta}_i$ and $\mathcal{M}_i$ represent the set of model parameters and the memory buffer at the end of task τ_i.

4.2.1 Selective freezing

In accordance with synaptic consolidation [30, 31], we propose a method for parameter freezing to maximize stability and plasticity. Specifically, we propose to train the model at the beginning of each task for a limited number of iterations under varying parameter freezing settings, providing an opportunity to the model to find the optimal configuration that combines retaining of previous knowledge and learning of the new task.

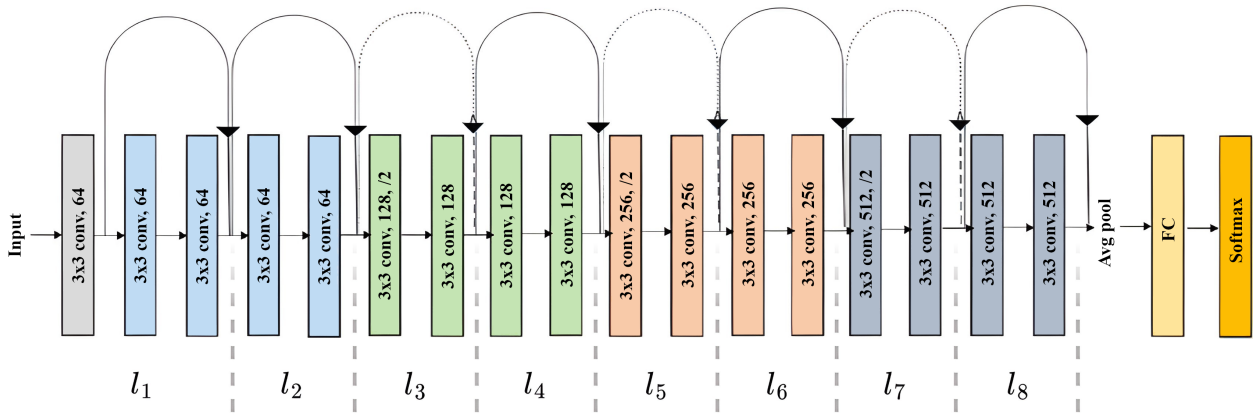

Figure 4.1: **ResNet-18 architecture and selective freezing strategy**. After an initial convolutional layer (apart from the last fully-connected layer), each colored portion represents a network's *main block*, consisting of two residual *basic blocks*, each applying two convolutions. For our selective freezing strategy, we treat each basic block as the smallest unit of freezing, named *layer*.

Formally, we want to model the joint probability between task data $\mathcal{D}_i$, previous experience $\mathcal{M}_{i-1}$, model parameters $\boldsymbol{\theta}_i$ and a binary freezing mask $\mathbf{m}_i$, with the same dimensions as $\boldsymbol{\theta}_i$ and such that $m_{i,j} = 1$ indicates that parameter $\theta_{i,j}$ should be frozen:

$$P(\mathbf{x}, y, \boldsymbol{\theta}_i, \mathbf{m}_i) = P\left(y \mid \mathbf{x}, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i, \mathbf{m}_i) P(\mathbf{x}), \tag{4.2}$$

where $\mathbf{x}$ and y represent samples and labels from $\mathcal{D}_i \cup \mathcal{M}_{i-1}$. The first term of the decomposition of Eq. 4.2 is the likelihood of correct labels given the input and the model prediction, while the joint distribution $P(\boldsymbol{\theta}_i, \mathbf{m}_i)$ describes the relation between model parameters $\boldsymbol{\theta}_i$ and the freezing strategy defined by $\mathbf{m}_i$, and can be expressed as:

$$P(\boldsymbol{\theta}_i, \mathbf{m}_i) = P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{m}_i), \tag{4.3}$$

where

$$P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) = \prod_j \mathcal{N}\left(\theta_{i,j}; \theta_{i-1,j}, \sigma_i^2\right)\Big)^{1-m_{i,j}}. \tag{4.4}$$

In this formulation, we model the distribution of each parameter $\theta_{i,j}$ as a Gaussian distribution depending on the corresponding mask value $m_{i,j}$, which removes a term from the overall probability when $m_{i,j} = 1$. Note that the mean of each parameter is set to $\theta_{i-1,j}$, i.e., its value at the end of the previous task (or to 0 for the first task, based on common initialization strategies).

In order to model $P(\mathbf{m}_i)$ in a practically feasible way, we employ some simplifying as-

sumption based on the layered structure of deep learning models. Given $F = l_1 \circ l_2 \circ \ldots \circ l_L$, where each l_k represents a network layer with parameters $\boldsymbol{\theta}_{|k}$ and $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_{|1}, \ldots, \boldsymbol{\theta}_{|L}\right]$, let us similarly define $\mathbf{0}_{|k}$ and $\mathbf{1}_{|k}$ as two tensors with the same size as $\boldsymbol{\theta}_{|k}$, with all values set to 0 and 1, respectively. Then, we impose that possible values for $\mathbf{m}_i$ must be parameterized by a value l as follows:

$$\mathbf{m}_i(l) = \left[\mathbf{1}_{|1}, \ldots, \mathbf{1}_{|l}, \mathbf{0}_{|l+1}, \ldots, \mathbf{0}_{|L}\right] \vee \mathbf{m}_{i-1} \tag{4.5}$$

with $l \in \{1, \ldots, L\}$. In practice, parameters frozen at previous tasks must remain so at the current task, and a layer's parameters can only be frozen altogether if all previous layers are also frozen.

Given these constraints, our goal is to find the optimal binary mask $\mathbf{m}_i$ that maximizes the likelihood of the labels y given the inputs $\mathbf{x}$ from current task $\mathcal{D}_i$ and from long-term memory $\mathcal{M}_{i-1}$. This is expressed as the following optimization problem:

$$\arg\max_{\mathbf{m}_i, \boldsymbol{\theta}_i} P\left(y \mid \mathbf{x}, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{m}_i) P(\mathbf{x}), \tag{4.6}$$

where the optimization is over parameters $\boldsymbol{\theta}_i$ and all feasible binary masks $\mathbf{m}_i$. Hence, the search over $\mathbf{m}_i$ is restricted to the discrete set of masks induced by Eq. 4.5 and performed by testing each admissible configuration individually, retaining the one that best balances stability and plasticity according to the objective.

The choice of $\mathbf{m}_i$ is thus formalized through the maximization of this likelihood, carried out via the optimization of the selective freezing loss function $\mathcal{L}_{\text{sf}}$:

$$\mathcal{L}_{\text{sf}} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_i}\left[\mathcal{L}\left(y, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right] + \alpha \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{M}_{i-1}}\left[\mathcal{L}\left(y, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right], \tag{4.7}$$

where $\mathbf{m}_i$ varies as described above, and α is a weighting factor between data sources. It is important to notice that, while optimizing for $\mathbf{m}_i$ necessarily requires updating $\boldsymbol{\theta}_i$ as well (since freezing, per se, does not alter inference performance), the objective is to prepare the model by identifying the optimal set of parameters that should be kept from previous tasks in a way that ensures both knowledge retention and room for plasticity. For this reason, optimization is carried out for a single epoch over $\mathcal{D}_i$. Note that the choice of $\mathcal{L}$ is arbitrary: the proposed formulation allows for plugging in any existing continual learning method, enhancing it with the proposed training strategy.

4.3 Experimental results

4.3.1 Datasets and metrics

In our experiments, in order to define a set of continual learning tasks $\{\tau_1, \ldots, \tau_T\}$, we employ the Split CIFAR-10 [96] dataset, which is obtained by splitting the CIFAR-10 dataset [63]. For the experiments with a pre-trained backbone, the datasets used for the pre-training phase are ImageNet [143], CIFAR-100 [63] and ImageNette [144].

The proposed approach and the related baselines have been evaluated in the standard *class-incremental learning* (Class-IL) setting, in which the model is required to gradually solve the complete problem while classes become available at different times. For the performance evaluation, we report the average classification accuracy over all dataset classes in the test set; we assume that the model has no access to task identity when classifying a given input.

4.3.2 Baselines

We carry out a set of experiments, freezing different portions of the backbone network according to different criteria, while employing the well-known continual learning DER++ [96] and ER-ACE [103] approaches. As a backbone for both approaches, we employ a ResNet-18 network [111], illustrated in Fig. 4.1. In standard implementations of the model[1], the feature extraction portion of the network consists of four *main blocks* (depicted with distinct colors in the figure), each of which includes two *basic blocks*; in turn, each basic block applies two convolutions with a residual connection. In the following experiments, we will treat each block as the smallest unit of freezing, named *layer*. Consequently, the network is divided into eight *layers*, aligning with the original structure of the basic blocks. The first convolution of the network is assumed to be part of the first layer.

4.3.3 Training procedure

In our experiments, we follow [96] and adopt the same training settings. All models are trained by means of Stochastic Gradient Descent (SGD) with a fixed learning rate of 0.03. Training is performed separately for 50 epochs for each task. Training batches are composed by mixing data retrieved from the current task and previous samples from the replay buffer. Buffer sizes 200 and 500 were used in our experiments. The training dataset is normalized

[1] We refer to the PyTorch implementation.

Pretraining	**ImageNet-1k**		**CIFAR-100**		**ImageNette**	
Layer	**200**	**500**	**200**	**500**	**200**	**500**
l_1	69.84	80.31	55.63	67.42	67.87	74.98
l_2	70.18	78.44	56.07	67.39	69.52	75.09
l_3	69.27	78.95	57.96	68.15	70.02	74.34
l_4	70.34	80.08	54.79	66.51	68.02	74.17
l_5	70.10	80.85	54.62	67.29	71.82	75.40
l_6	69.06	76.69	53.66	63.56	71.64	75.48
l_7	62.02	75.68	43.08	55.39	71.49	77.23
l_8	10.13	9.09	7.92	10.18	71.44	74.47
not frozen	72.96	81.96	57.64	68.78	67.51	75.97

Table 4.1: **Effect of layer freezing when using a pre-trained backbone with DER++.** Results are computed at the end of the last task on Split CIFAR-10, for different pre-training modalities, in the Class-IL setting.

using the mean and the standard deviation values computed on data used for the training process. The data augmentation strategy includes random crop and horizontal flip.

For evaluating the freezing strategies, the optimization of Eq. 4.7 is carried out on a distinct portion of the training set, which we refer to as *validation set*. Specifically, for a given task τ_i, we optimize $\mathcal{L}_{\text{sf}}$ on the corresponding validation $\mathcal{D}_{i,\text{val}} \cup \mathcal{M}_{i-1,\text{val}}$, where $\mathcal{D}_{i,\text{val}}$ contains 10% of the training set $\mathcal{D}_i$, and $\mathcal{M}_{i-1,\text{val}}$ includes 10% of the buffer $\mathcal{M}_{i-1}$. Additionally, as alternative approaches, we also try using only the data from the current task ($\mathcal{D}_{i,\text{val}}$), or exclusively with the data contained in the buffer ($\mathcal{M}_{i-1,\text{val}}$).

All experiments were run on a single NVIDIA RTX 3090 GPU and implemented using the PyTorch framework.

4.3.4 Static freezing

We first present a set of experiments aimed at assessing the overall impact of different *static* freezing strategies (i.e., where the choice of which layers to freeze and the task at which they are frozen is performed *a priori*, independently of the model's performance). This preliminary analysis provides useful hints on how continual learning performance is affected by feature freezing in general; moreover, it allows us to establish a reference for comparison in our experiments with selective freezing.

We first analyze the effect of layer freezing with a pre-trained ResNet-18 backbone. A common setting of continual learning methods assumes that model features are trained from

scratch, in order to assess a method's capability to learn strong features that support forward transfer and are not spurious or task-specific. In our scenario, it is interesting to evaluate the extent to which layer freezing interacts with the availability of pre-trained features: indeed, this setting should reduce the need for forward transfer (as we can assume that pre-trained features mitigate spurious feature learning) and robustness to forgetting (due to freezing). The results of these experiments are shown in Tab. 4.1 (for DER++) and Tab. 4.2 (for ER-ACE). We report the class-incremental learning (Class-IL) accuracy computed after the last task in Split CIFAR-10, for buffer sizes 200 and 500, when freezing up to a certain layer (in the table, the row corresponding to l_i refers to when the i-th layer w.r.t. Fig. 4.1 is frozen, along with all previous layers); we take into account classification pre-training on three different datasets, CIFAR-100, ImageNette[2] and ImageNet-1k [143]. It is interesting to note that the Class-IL accuracy does not decrease significantly when freezing up to layer l_6, in several cases even improving accuracy with respect to the non-frozen case. As mentioned above, this can be expected, since freezing pre-trained features helps reduce forgetting while reusing already good features, at the cost (in this case, negligible) of giving up learning task-specific low-level features. However, going deeper with freezing, performance drops when using pretrained features from CIFAR-100 and ImageNet-1k, showing that a certain degree of feature customization may be in order. Interestingly, this drop is mitigated when pre-training on ImageNette (see Tables 4.1 and 4.2). A possible explanation lies in the fact that ImageNette only has ten classes: as such, it is tailored towards learning features that are less class-specific and that can be effectively leveraged by the final fully-connected layer, even when the entire backbone of the model is frozen.

The previous results show that layer freezing does not seem to incur a significant performance loss in a continual learning setting, thanks to pre-training. However, it is more interesting to see how this changes when the entire backbone is trained from scratch. To this aim, Figures 4.2, 4.3, 4.4, 4.5 show the results for DER++ and ER-ACE without a pre-trained backbone. Accuracy is reported in terms of Class-IL for different buffer sizes, on Split CIFAR-10. In detail, each table reports these metrics when the backbone is frozen up to a certain layer at the end of a certain task: during previous tasks, the entire backbone is updated. This setting thus takes into account not only the depth but also the *moment* at which backbone features are frozen, allowing us to study the trade-off between freezing early (encouraging feature preservation but reducing plasticity) or late (encouraging adaptability to new tasks but risking catastrophic forgetting). As expected, in the absence of a pre-training stage, freezing tends to reduce model performance compared to full training of the entire backbone. A general trend can be identified for both DER++ and ER-ACE

[2]ImageNette is available at: https://github.com/fastai/imagenette

Pretraining	**ImageNet**		**CIFAR-100**		**ImageNette**	
Layer	**200**	**500**	**200**	**500**	**200**	**500**
l_1	79.87	84.94	66.78	73.02	59.69	59.18
l_2	79.62	85.80	65.38	73.41	60.32	61.65
l_3	79.62	86.03	65.22	72.90	55.93	57.98
l_4	80.31	84.47	66.14	71.00	57.79	56.97
l_5	80.06	85.76	65.08	69.72	53.45	54.02
l_6	79.80	85.56	64.60	70.24	50.76	52.25
l_7	79.01	83.96	60.62	67.35	50.71	49.61
l_8	11.74	10.47	10.35	11.17	38.57	39.96
not frozen	80.70	85.11	64.94	73.70	59.33	65.66

Table 4.2: **Effect of layer freezing when using a pre-trained backbone with ER-ACE.** Results are computed at the end of the last task on Split CIFAR-10, for different pre-training modalities, in the Class-IL setting.

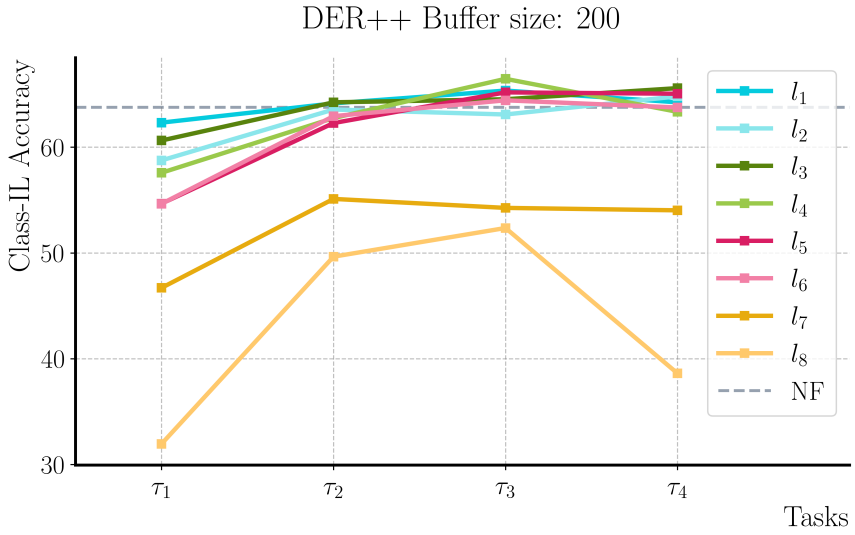

Figure 4.2: **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the DER++ method and a buffer size of 200.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen.

and for both tested buffer sizes: later freezing (with respect to the sequence of tasks) yields higher accuracy, as the model can work at its full capacity for a longer time; deeper freezing (with respect to the sequence of layers in the backbone), tends to reduce accuracy, for a similar reason, since the model is forced to reuse features that — unlike in the pre-training experiments — may not be representative for future tasks. Nevertheless, it is possible to find

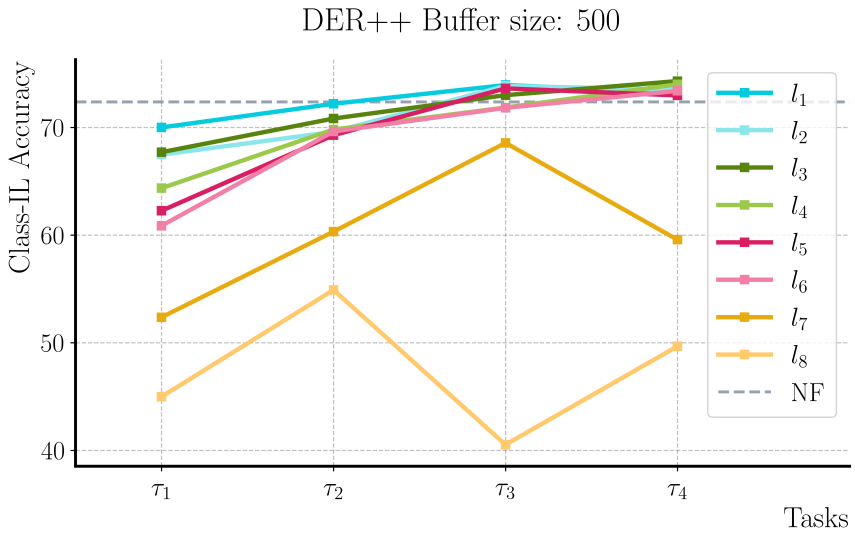

Figure 4.3: **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the DER++ method and a buffer size of 500.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen.

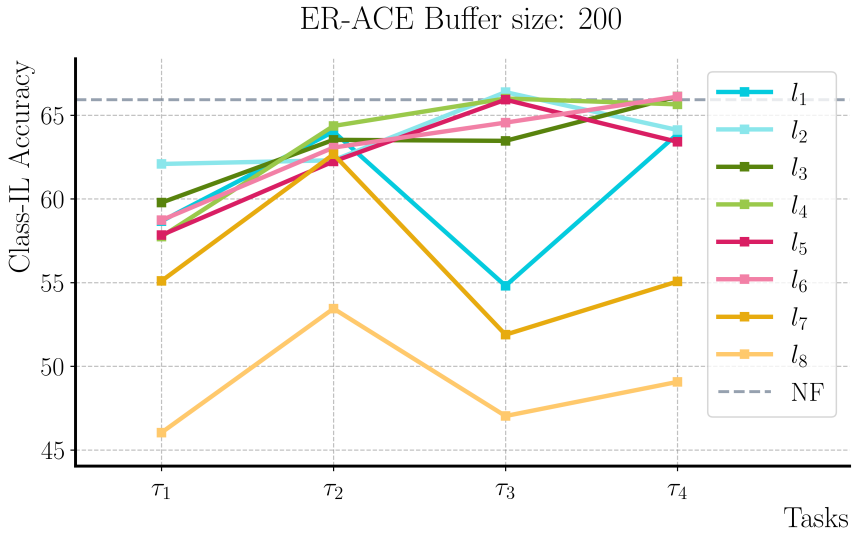

Figure 4.4: **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the ER-ACE method and a buffer size of 200.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen.

a "sweet spot" among the different freezing configurations, representing good compromises between accuracy and efficiency. For instance, while freezing after the first task may be too early to learn good features, freezing after the second task, even up to l_3, leads to a relatively small decrease in accuracy, on both the Class-IL and Task-IL metrics. This behavior appears

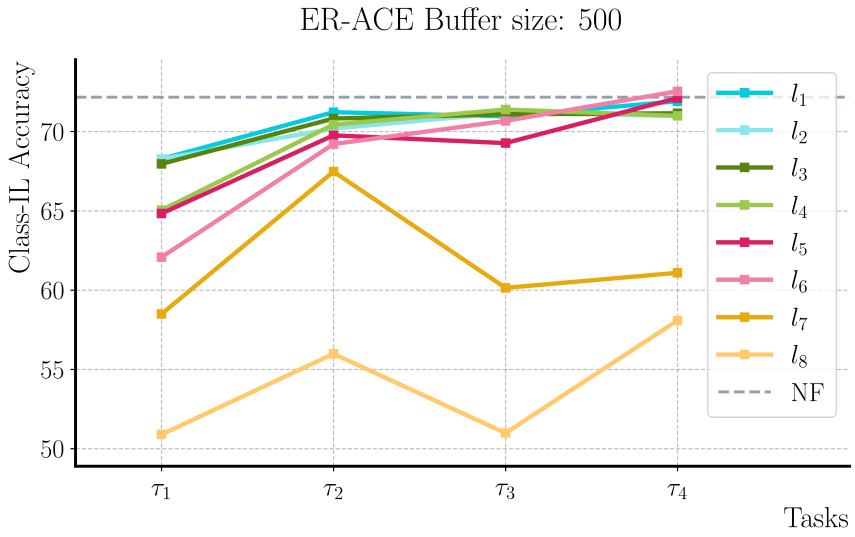

Figure 4.5: **Effect of layer freezing, when training from scratch, performed at end of different tasks when using the ER-ACE method and a buffer size of 500.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting. "NF" refers to the baseline setting where no layer is frozen.

Validation set	**DER++**		**ER-ACE**	
	200	**500**	**200**	**500**
$\mathcal{D}_{i,\text{val}} \cup \mathcal{M}_{i-1,\text{val}}$	60.20	66.92	59.52	64.65
$\mathcal{D}_{i,\text{val}}$	62.20	69.89	58.52	66.26
$\mathcal{M}_{i-1,\text{val}}$	58.84	62.35	57.50	64.62
vanilla training	64.83	72.13	63.81	71.86

Table 4.3: **Effect of selective freezing obtained using DER++ and ER-ACE methods.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting, for both 200 and 500 buffer sizes.

to be consistent for DER++ and ER-ACE, for buffer sizes 200 and 500.

4.3.5 Selective freezing

In this section, we present the results obtained while applying the proposed selective freezing strategy to DER++ and ER-ACE on the Split CIFAR-10 dataset, for different buffer sizes. In detail, Tab. 4.3 reports Class-IL accuracy at the end of the final task with the three variants of selective freezing described in Sect. 4.3.3, i.e., when the validation set of freezing decision includes either training samples only (from the new task), or buffer samples only, or both.

Validation set	DER++		ER-ACE	
	200	**500**	**200**	**500**
$\mathcal{D}_{i,\text{val}} \cup \mathcal{M}_{i-1,\text{val}}$	57.57	67.90	66.21	71.86
$\mathcal{D}_{i,\text{val}}$	58.11	67.56	64.84	71.02
$\mathcal{M}_{i-1,\text{val}}$	56.46	64.77	64.23	69.53
vanilla training	58.21	69.00	66.33	73.77

Table 4.4: **Effect of selective freezing with DER++ and ER-ACE methods using a backbone pre-trained on the CIFAR-100 dataset.** Results are computed at the end of the last task on Split CIFAR-10 in the Class-IL setting, for both 200 and 500 buffer sizes.

It can be observed that generally the results obtained with selective freezing are slightly lower compared to the case of vanilla training, with a drop in accuracy in the range of 3 to 6 percentage points. The overall performance aligns with the optimal strategy identified through static freezing, which involves freezing after task 2 up to l_5. Moreover, it seems that training on new task data only (i.e., without buffer samples) leads to improved performance, indicating that adaptation to new data during selective freezing plays a more important role than recalling past knowledge, which is instead handled during standard training.

Tab. 4.4 reports results from the same experiments, carried out starting from a pre-trained backbone. It is interesting to notice that, in this scenario, ER-ACE seems to benefit from pre-training significantly more than DER++. This may indicate that ER-ACE exhibits better properties at feature reuse and forward transfer; interestingly, this interpretation is also supported by the results shown in the following, according to which ER-ACE has a stronger tendency to selectively freeze larger parts of the backbone at earlier tasks.

We finally assess the computational efficiency of our selective freezing strategy on DER++ and ER-ACE compared to standard training. Fig. 4.6 illustrates the prevailing freezing scheme observed in DER++ during training on CIFAR-10, over 10 different experimental runs, while Figure 4.7 shows the resulting efficiency in terms of the number of parameter updates required during the entire training process. Notably, the efficiency gain shows an increasing trend with the growth of the number of epochs per task. When the number of epochs is relatively low (e.g., 10), selective freezing leads to a higher number of updates compared to standard training. This is due to the fact that during the first epoch of the second task, up to 7 different model configurations are trained and evaluated in parallel. However, as the number of epochs increases, the overhead introduced during this phase becomes progressively more marginal, leading to a considerable efficiency gain of 12.34% with 100 epochs per task.

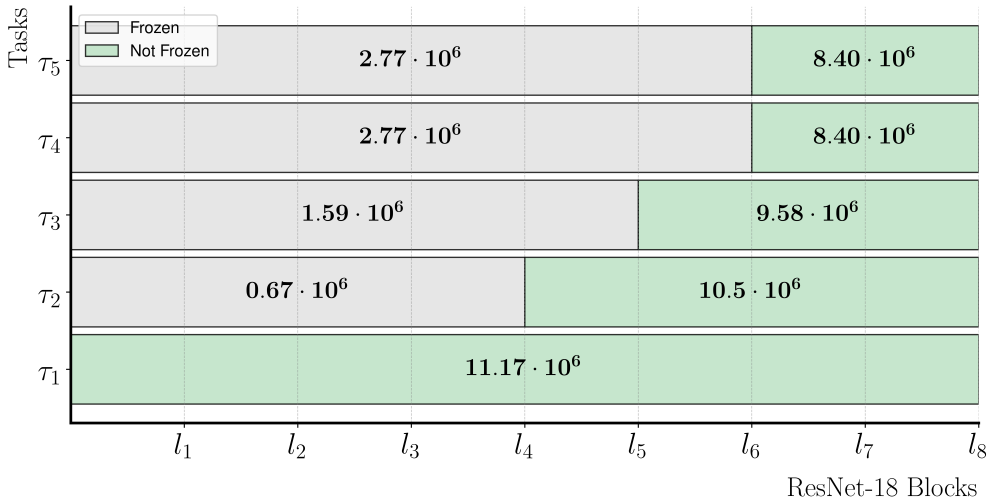

Figure 4.6: **Most frequent automatically learned freezing scheme for DER++ on Split CIFAR-10.** Results are computed over 10 runs. Values within each bar segment indicate the number of parameters.

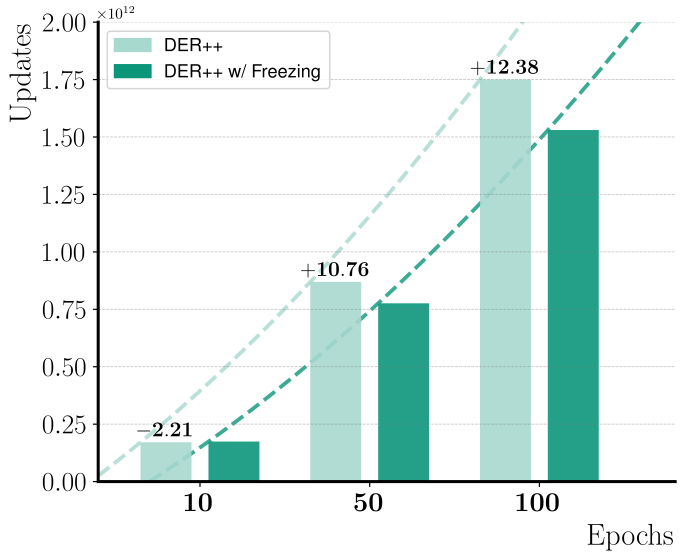

Figure 4.7: **Total number of parameter updates for DER++ with and without selective freezing.** The number of updates is computed over the complete training for different numbers of epochs. Values above the vanilla DER++ bars report the relative change (%) with respect to the selective freezing strategy, where positive values denote more updates.

It is worth noting that the specific behaviors of freezing can also depend on the particular methods of continual learning employed in the experiments. As an example, ER-ACE tends to freeze more layers in the initial phase, showing a preference for a more conservative configuration right at the beginning of the second task, as depicted in Figure 4.8, while

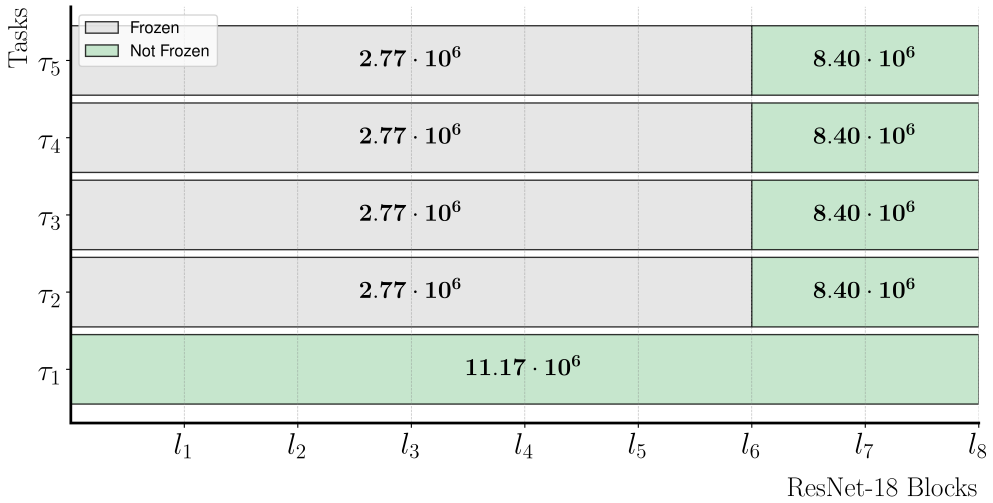

Figure 4.8: **Most frequent automatically learned freezing scheme for ER-ACE on Split CIFAR-10.** Results are computed over 10 runs. Values within each bar segment indicate the number of parameters.

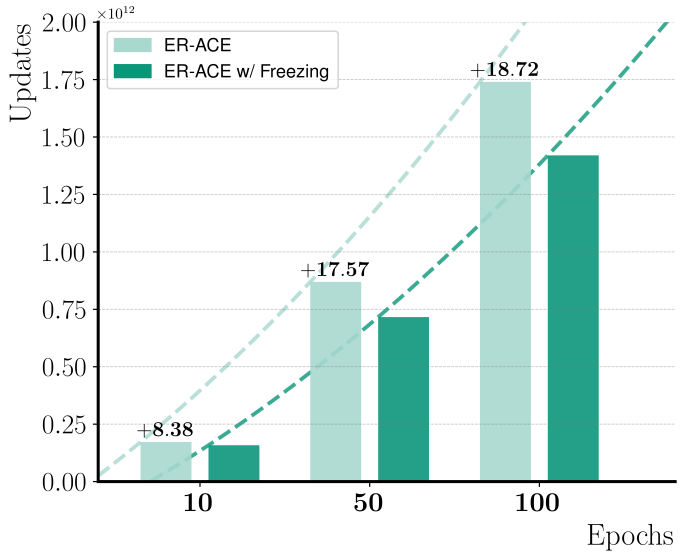

Figure 4.9: **Total number of parameter updates for ER-ACE with and without selective freezing.** The number of updates is computed over the complete training for different numbers of epochs. Values above the vanilla ER-ACE bars report the relative change (%) with respect to the selective freezing strategy, where positive values denote more updates.

DER++ adopts a more gradual freezing strategy. Indeed, freezing more layers in an early stage has a positive impact on efficiency: for ER-ACE the efficiency gain ranges from 8.38% in the case of 10 epochs per task, to a remarkable 18.72% extending the number of epochs to 100, as shown in Figure 4.9.

4.4 Discussion

In this chapter, we have presented a neuro-inspired approach that performs selective layer freezing to address the challenges of continual learning and computational efficiency in deep learning models. Building on ideas from multi-timescale synaptic consolidation, our method explicitly regulates where plasticity is allowed to remain high and where it is progressively constrained, mirroring the interplay between fast adaptive changes and slower stabilizing mechanisms in the human brain. Our findings highlight the potential for freezing a significant portion of the model without sacrificing accuracy: by dynamically identifying a subset of layers to freeze during training, we achieve a balance between plasticity on new tasks and stability on previous ones. Experimental results demonstrate the competitiveness of our approach compared to manually tuned freezing strategies. We have also quantitatively estimated the reduction in computation and energy requirements achieved through our freezing strategy, showing that biologically inspired control of plasticity can mitigate catastrophic forgetting while lowering the resource consumption of large-scale deep learning models.

While the proposed approach is able to achieve promising results, its simplicity introduces certain limitations that need to be addressed. First, our experiments focus on a specific network architecture, ResNet-18. Although this backbone is a *de facto* standard in much of the continual learning literature, it is important to assess the impact of selective freezing on different and deeper architectures. Second, our analysis and experiments are conducted on a specific dataset and task setup: the effectiveness of our approach may vary with different task characteristics, dataset sizes, and variation in task sequence.

In the future, we also aim to explore more sophisticated freezing strategies, for instance by introducing a finer selection of parameters, rather than working at the layer level. Secondly, investigating the interplay between freezing strategies and other techniques, such as regularization methods, knowledge distillation or architectural modifications, could provide additional insights into improving continual learning and computational efficiency.

4.5 Publications

Sorrenti, A., Bellitto, G., Salanitri, F. P., Pennisi, M., Spampinato, C., & Palazzo, S. (2023). "Selective freezing for efficient continual learning". In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (pp. 3550-3559), Workshop on Visual Continual Learning.

Chapter 5

Wake-Sleep Consolidated Learning

The gap between artificial neural networks and humans in learning over time can be attributed to inherent differences in network structure and optimization approaches. According to the Complementary Learning Systems (CLS) theory [35, 36], effective human learning occurs through the interplay between the hippocampus and the neocortex, where the former extracts representations from experience and the latter supports consolidation and long-term retention. This view has inspired continual learning methods [99, 105] which translate CLS concepts into computational frameworks, suggesting that grounding artificial neural networks to cognitive neuroscience may result in improved performance. Though promising, these approaches are rather rigid as the structures of the two learning parts are defined *a priori*, while neural networks in primates perform fast adaptation by flexibly re-configuring synapses while learning from new experience. Moreover, prior work does not consider the role of offline brain states such as sleep. Current theories suggest that sleep and dreaming play a crucial role in consolidating memories and facilitating learning, by increasing knowledge generalization [38, 39, 40]. During sleep, neurons are spontaneously active without external input and generate complex patterns of synchronized activity across brain regions [41, 42]. Such dynamics are believed to be due to the brain replaying and consolidating memories, while reorganizing synaptic connections [43].

In this chapter, we propose Wake-Sleep Consolidated Learning (WSCL), a CLS-grounded formulation structured around the alternation of wakefulness and sleep, and built on the fast adaptation mechanism introduced in Chapter 4. This integration is achieved by introducing a sleep phase at training time that mimics the offline brain states during which synaptic connection, memory consolidation and dreaming occur. In WSCL, a deep neural network (DNN) is used to emulate the functions of the neocortex, while a two-layered buffer for short-term and long-term memory mimics the role of the hippocampus. Training is organized in two main phases: 1) a *wake phase*, where fast adaptation of the DNN to new sensory

experience is carried out and episodic memories are stored in the short-term memory; 2) a *sleep phase*, consisting of two alternating stages: a) Non-Rapid Eye Movement (NREM), where the network replays episodic memories collected during the wake step, consolidates past experiences in the long-term memory, and optimizes its neural connections to support synaptic plasticity; b) Rapid Eye Movement (REM), where dreaming simulates new experience, preparing the brain for future events. The hypothesis is that dreaming serves as an "anticipatory" mechanism, helping the brain to identify relationships between different types of information and making it easier to learn and remember new information.

Our computational formulation of the wake-sleep process is tested on several benchmarks, including Split CIFAR-10, Split CIFAR-100, Split Tiny-ImageNet and Split ImageNet-100. In all cases, our method outperforms the baselines and prior work, yielding a significant gain in classification tasks. Remarkably, the WSCL approach is the first continual learning method yielding positive forward transfer, demonstrating its ability to prepare synapses to future knowledge. We also show that all three steps are necessary: the wake stage is essential to ensure efficiency and to favor network plasticity by the NREM stage, while the REM stage helps to increase feature transferability and reduce the forgetting of acquired knowledge.

5.1 Related work

As introduced in Sec. 2.5.3, rehearsal-based approaches emerge as the most promising avenue for combating catastrophic forgetting in continual learning scenarios especially in dynamic real-world applications. Unlike static models prone to overwriting prior knowledge, rehearsal-based techniques capitalize on past experiences by storing select samples in a small buffer, which the model continues to train on even when presented with new tasks. While current solutions help reduce forgetting, real-world applications prove difficult, as typical CL evaluations are carried out on oversimplistic benchmarks [145, 146]. Most approaches tackling this challenging scenario combine a replay strategy [147, 24, 86, 131] with regularization on logits sampled throughout the optimization trajectory [96]. Some works focus on memory management: GSS [91] introduces a specific optimization of the basic rehearsal formula meant to store maximally informative samples; HAL [148] individuates synthetic replay data points that are maximally affected by forgetting. CaSpeR [147] adopts a rehearsal-based strategy enforcing latent space regularization on buffer samples through geometric constraints. Other works propose tailored classification schemes: CoPE [149] uses class prototypes to ensure a gradual evolution of the shared latent space; ER-ACE [103] makes the cross-entropy loss asymmetric to minimize imbalance between current and past tasks. Recent works introduce a surrogate optimization objective: CR [150] employs a supervised contrastive learning ob-

jective and OCM [151] leverages mutual information: both aim at learning features that are less subject to forgetting.

In the literature, a few neuroscience-informed CL methods have been proposed. Elastic Weight Consolidation (EWC) [72] and Synaptic Intelligence [74], introduced in Sec. 2.5.3, both employ regularization to preserve important weights learned during previous tasks while allowing the network to adapt to new tasks, emulating fast adaptation happening in the neocortex. FearNet [152] adopts an auxiliary network (in line with CLS theory) to detect catastrophic forgetting and trigger knowledge-preserving regularization. Co2L [101] learns stable representations through contrastive learning and self-supervised distillation.

Existing approaches similarly inspired by CLS theory are DualNet [99], DualPrompt [105] and CLS-ER [153]. DualNet [99] employs two learning networks: a slow learner that emulates the memory consolidation process in the hippocampus and a fast learner that adapts current representations to new observations. DualPrompt [105] adapts transformer models via learnable prompts for rapid task adaptation, mirroring the complementary specialization of the hippocampus and the neocortex. CLS-ER [153] implements semantic memory using two separate neural networks to model short-term and long-term memory dynamics. While these approaches yield good results, they ignore offline states that appear fundamental in human learning. Alternating between wake and sleep phases has already been shown to have the potential for learning improved and robust semantic representations [154, 155]. Sleep Replay Consolidation [156] employs sleep-based training using local unsupervised Hebbian plasticity rules for mitigating catastrophic forgetting of ANN. The recent SIESTA [157] introduces alternating waking and sleeping and it primarily focuses on online learning with intermittent consolidation phases.

WSCL further unfolds the sleep phase by detailing the NREM and REM stages, integrating the dreaming process into the learning loop. This integration, which appears to contribute significantly to human learning, has a positive impact on the training of neural networks (as shown in the results). The computational formulation of the wake-NREM-REM of WSCL is inspired by [43], where the role of adversarial dreaming for learning visual representations is preliminary investigated. However, simple strengthening of existing connections through unsupervised learning as proposed in [43, 156] does not seem sufficient to build robust representations during sleep [40]: our work thus explores more sophisticated restructuring of neural connections in the neocortex guided by the hippocampus.

In addition, WSCL adopts the parameter selective freezing strategy introduced in Chapter 4. This strategy aims exclusively to enhance model efficiency, mirroring human fast adaptation, and operates during training without task-specific parameter selection during inference. This strategy is fundamentally different from architectural approaches such as the

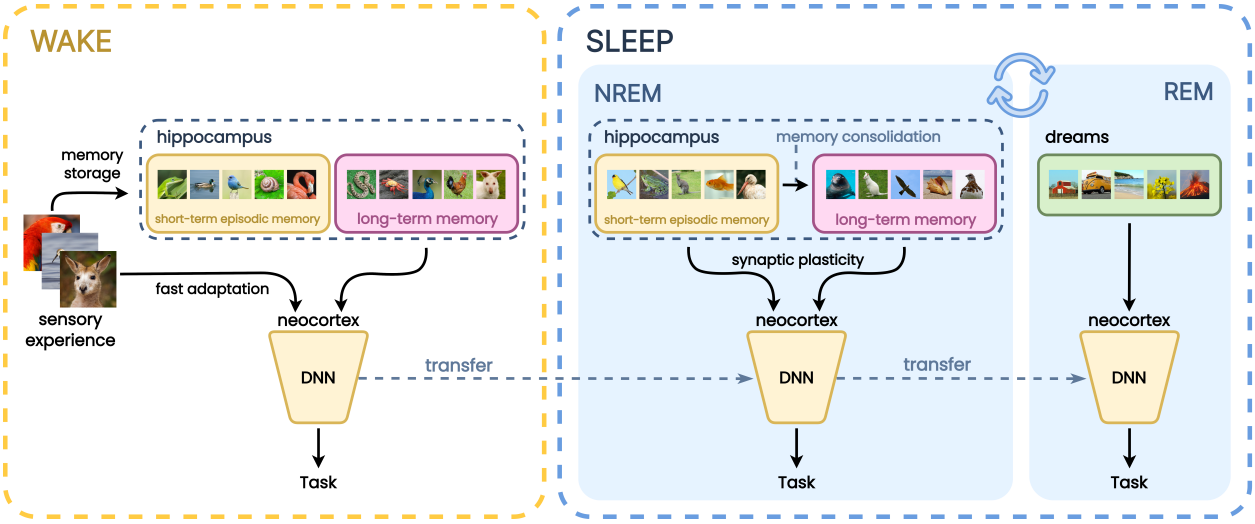

Figure 5.1: **Wake-Sleep Consolidated Learning.** In the wake stage, the model (which emulates the neocortex) fast adapts to the new sensory experience, storing episodic memories (as in the hippocampus) in the short-term memory to be replayed during sleep. The sleep phase foresees two alternating processes: 1) the NREM stage, where the DNN model consolidates its synapses based on the replayed (recent and past) samples and the long-term memory is updated; 2) the REM stage, where the DNN is trained with dreamed samples to prepare the model for future sensory inputs.

ones in [73, 78, 113, 158, 159] that, instead, learn task-specific parameters which are then selected, at inference time, based on task identifiers [78] or mask entropy [113] or scaling parameters [159]: WSCL maintains a unified network for all tasks, eliminating the necessity for task-specific masks or parameters during inference.

5.2 Method

In accordance with CLS theory [35, 36], a classifier network $F : \mathcal{X} \rightarrow \mathcal{Y}$ is used as an abstraction of neocortical processing, and the hippocampal episodic memory is modeled through a *long-term memory* and a *short-term memory*. Formally, $\mathcal{M}_i$ refers to the long-term memory at the end of task τ_i, while $\mathcal{M}_s$ is used to denote the short-term memory[1].

The objective is to train the model F over a sequence of T tasks $\{\tau_1, \ldots, \tau_T\}$, assuming that, during task τ_i, training data are available only from the corresponding data distribution $\mathcal{D}_i$, i.e., $(\mathbf{x}, y) \sim \mathcal{D}_i$. As defined in Sec. 2.1, the training objective is to minimize the classification loss $\mathcal{L}$ over the sequence of tasks, while balancing *plasticity* and *stability* [23].

An overview of the WSCL approach is presented in Fig. 5.1, showing how the training stage on a new task is divided into two phases: a *wake phase* and a *sleep phase*. During

[1]For brevity, we drop task index i from short-term memory $\mathcal{M}_s$, as it is re-created at each task.

the wake phase, the model is exposed to the new task, with the objective of performing fast adaptation of existing knowledge to the task characteristics. In this stage, the model quickly updates its parameters in order to find a balance between previously-acquired knowledge and new information, storing the latter in a short-term memory for later reuse during the sleep stage. In implementation terms, this balance is achieved by dynamically and adaptively freezing layer representations, identifying plasticity requirements for learning the new task while enforcing stability. Thus, during the wake stage, WSCL focuses primarily on quickly learning general and transferable representations by combining both current and past experience, as well as identifying which part of the network has to be trained. In the sleep phase, the model consolidates newly acquired knowledge by revisiting the hippocampus short-term memory containing the task data, merging it into existing knowledge by updating synaptic connections, moving it into a long-term memory for future reference, and exploring the representational space through task-agnostic "dreaming". These stages are mapped into our training procedure by means of supervised training on task data, buffering task information, and employing an auxiliary dataset (uncorrelated to task data) as a surrogate for the generative process associated to dreaming.

5.2.1 Wake phase

According to the established cognitive foundation, we define the waking stage in the proposed learning paradigm as the combination of two simultaneous processes, *short-term memorization* and *fast model adaptation*.

Short-term memorization has the objective of storing part of the current task experience, for later reuse — in particular, for processing and consolidation during the sleep stage. In a continual learning setting, we model short-term memorization into $\mathcal{M}_s$ as a sampling of task data $\mathcal{D}_i$:

$$\mathcal{M}_s = \{(\mathbf{x}_j, y_j) \sim \mathcal{D}_i\}_{j=1}^{N_s}, \tag{5.1}$$

where N_s is the amount of samples collected from the $\mathcal{D}_i$ distribution. Note that $\mathcal{M}_s$ is reset during each wake phase and is distinguished from the *long-term memory* $\mathcal{M}_i$, which includes a smaller permanent number of samples N_l from past tasks (in practice, the buffer of rehearsal-based methods).

Fast model adaptation. Inspired by synaptic consolidation [30, 31], we employ the selective freezing strategy introduced in Chapter 4 to enable fast model adaptation during the wake stage. This strategy operates only during the model's training, as it aims at adapting

quickly the model to the current data distribution, while it does not activate at inference time when the model's knowledge is already consolidated. Specifically, fast model adaptation works by training the model for a limited number of iterations under varying parameter freezing settings, providing an opportunity to the model to rapidly learn new information in the wake stage while retaining the previous knowledge; in-depth consolidation of task information is carried out separately in the sleep stage. Unlike approaches such as DualNet, where the structure of the slow and fast networks is predefined, in WSCL the part of the network that reuses past knowledge and the part accounting for plasticity are identified on-line during the wake phase.

Following nomenclature from Chapter 4, let $\boldsymbol{\theta}_i$ denote the model parameters after training on task τ_i, and let $\mathbf{m}_i$ be a binary freezing mask with the same dimensions as $\boldsymbol{\theta}_i$, where $m_{i,j} = 1$ indicates that parameter $\theta_{i,j}$ should be frozen. The joint distribution over data samples from $\mathcal{D}_i \cup \mathcal{M}_{i-1}$, the model parameters $\boldsymbol{\theta}_i$, and the freezing mask $\mathbf{m}_i$ is defined as:

$$P(\mathbf{x}, y, \boldsymbol{\theta}_i, \mathbf{m}_i) = P\left(y \mid \mathbf{x}, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i, \mathbf{m}_i) P(\mathbf{x}). \tag{5.2}$$

The joint distribution between $\boldsymbol{\theta}_i$ and $\mathbf{m}_i$ can be written as $P(\boldsymbol{\theta}_i, \mathbf{m}_i) = P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{m}_i)$ where:

$$P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) = \prod_j \mathcal{N}\left(\theta_{i,j}; \theta_{i-1,j}, \sigma_i^2\right)\Big)^{1-m_{i,j}}. \tag{5.3}$$

The distribution of each parameter $\theta_{i,j}$ is a Gaussian distribution depending on the corresponding mask value $m_{i,j}$, so that frozen parameters ($m_{i,j} = 1$) do not contribute to the update.

Fast adaptation is then performed by minimizing the following objective:

$$\begin{aligned} \mathcal{L}_{\text{fma}} = \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_i}\Big[\mathcal{L}(y, F(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i))\Big] \\ + \alpha\, \mathbb{E}_{(\mathbf{x},y)\in\mathcal{M}_{i-1}}\Big[\mathcal{L}(y, F(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i))\Big], \end{aligned} \tag{5.4}$$

where α weights the contribution of replay samples from $\mathcal{M}_{i-1}$.

Since freezing does not change the model output by itself, searching over $\mathbf{m}_i$ requires updating $\boldsymbol{\theta}_i$. The resulting objective selects parameters to preserve from past tasks while maintaining plasticity, and is optimized for a single epoch on $\mathcal{D}_i$. The choice of $\mathcal{L}$ is generic, enabling integration with arbitrary continual learning methods.

Further details on the problem formulation of the freezing strategy are provided in Sec. 4.2.

5.2.2 Sleep phase

During sleep, the brain cycles multiple times through two phases, known as rapid eye movement (REM) and non-rapid eye movement (NREM) sleep. In the NREM phase, the hippocampus replays and consolidates the information acquired at waking time by facilitating its transfer to the neocortex, where long-term memory storage occurs [38, 160]. REM sleep is thought to play a role in creativity and problem-solving [161, 162], allowing the brain to form new connections and generate novel ideas. In our WSCL approach, we analogously distinguish between two alternating training modalities, conceptually mapped to the NREM and REM phases.

During the former, we access examples from the current task (stored in the short-term memory) and from previous tasks (retrieved from long-term memory) to train the model — partially frozen during the wake stage — and stabilize present knowledge. In the REM stage, we emulate the dreaming process by providing the model with examples from an external data source, with classes unrelated to any continual learning task. This approach allows the model to learn task-agnostic features which can be interpreted as prior knowledge supporting task-specific learning and forward transfer.

NREM stage. The main objective of this stage is to transfer information from the short-term memory $\mathcal{M}_s$, built in the previous wake phase, to the model, strengthening the synaptic connections associated to the current task and thus enforcing plasticity, while retaining previously acquired knowledge thanks to long-term memory $\mathcal{M}_{i-1}$. In this setting, we apply parameter freezing mask $\mathbf{m}_i$ (defined in the wake phase), which is however not updated in the process.

Formally, in this stage we model the same distribution as in Eq. 5.2, but optimize for $\boldsymbol{\theta}_i$ alone, while leaving $\mathbf{m}_i$ constant. The objective thus becomes:

$$\arg\max_{\boldsymbol{\theta}_i} P\left(y \mid \mathbf{x}, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{x}), \tag{5.5}$$

where the prior on parameters $P\left(\boldsymbol{\theta}_i \mid \mathbf{m}_i\right)$ is essentially the same as in Eq. 5.3, with the difference that the mean of the distribution is the value of $\boldsymbol{\theta}_i$ as computed at the end of the wake stage, rather than $\boldsymbol{\theta}_{i-1}$. Optimizing the above objective amounts to minimizing a variant of the loss in Eq. 5.4:

$$\begin{aligned} \mathcal{L}_{\text{NREM}} =& \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{M}_s}\left[\mathcal{L}\left(y, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right] \\ &+ \alpha \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{M}_{i-1}}\left[\mathcal{L}\left(y, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right], \end{aligned} \tag{5.6}$$

where $\mathcal{M}_s$ is employed instead of the whole dataset $\mathcal{D}_i$.

In this stage, we also gradually update long-term memory $\mathcal{M}_i$, using reservoir sampling [66] to inject task experience from short-term memory $\mathcal{M}_s$ into $\mathcal{M}_i$, so that it becomes available to future tasks.

REM stage. We approximate the sleeping mechanism performed by the human brain in the REM stage by providing the model with an additional source of previously unseen knowledge (a "dreaming" dataset with no semantic overlap with CL classes), that can help the model to generalize better to new and unseen data, as suggested by cognitive literature [43].

Let $\mathcal{D}_{\text{dream}}$ be the dreaming dataset from which we can sample data points $(\mathbf{x}, y)$, with $\mathbf{x} \in \mathcal{X}$ and class label $y \in \mathcal{Y}_{\text{dream}}$. We assume that $\mathcal{Y}_{\text{dream}} \cap \mathcal{Y} = \emptyset$ (the latter being the set of continual learning classes), to prevent any overlap between auxiliary and continual learning classes. Given this premise, the proposed optimization objective becomes:

$$\arg\max_{\boldsymbol{\theta}_i} P\left(y \mid \mathbf{x}, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right) P(\boldsymbol{\theta}_i \mid \mathbf{m}_i) P(\mathbf{x}), \tag{5.7}$$

where $(\mathbf{x}, y) \sim \mathcal{D}_{\text{dream}}$, while the other terms are the same as in Eq. 5.5. This objective is then mapped to a training loss function defined as:

$$\mathcal{L}_{\text{REM}} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{\text{dream}}}\left[\mathcal{L}\left(y, F\left(\mathbf{x}, \boldsymbol{\theta}_i, \mathbf{m}_i\right)\right)\right]. \tag{5.8}$$

During REM stage, training with two distinct class label sets, $\mathcal{Y}$ from the continual learning problem and $\mathcal{Y}_{\text{dream}}$ from the dreaming dataset has been addressed following the procedure reported in [163].

5.3 Experimental results

5.3.1 Datasets and metrics

We evaluate WSCL on a subset of the continual learning benchmarks described in Sec. 2.3. Since the REM stage requires additional dreaming samples, we additionally identify a dreaming counterpart for each benchmark:

- **Split CIFAR-10** [74]: the dreaming counterpart is a subset of 50 CIFAR-100 classes, selected after removing those semantically related to CIFAR-10.
- **Split CIFAR-100** [74]: the dreaming counterpart, referred to as *ImageNet*^{aux}, consists

of 100 ImageNet classes selected after removing those semantically related to CIFAR-100.

- **Split Tiny-ImageNet** [60]: we use the first 100 classes as the main training dataset *Tiny-ImageNet*$_{1/2}$ (split into 5 tasks of 20 classes) and the remaining 100 classes as the dreaming dataset, referred to as *Tiny-ImageNet*^{D}.
- **Split ImageNet-100**[2]: the dreaming counterpart, referred to as *ImageNet*^{NO}, consists of 100 additional ImageNet classes selected after removing all synsets descending from *organism* ($^{\text{NO}}$ stands for "non-organism").

To evaluate performance, we report *final average accuracy* (FAA), *final average forgetting* (FAF) and *final forward transfer* (FFWT) after training on the last task in the class-incremental setting.

5.3.2 Training procedure

Our approach employs a ResNet-18 backbone for feature extraction and classification. ResNet-18 includes, at a high level, four *layers* with two blocks each, for a total of eight blocks[3] With reference to the definition of model f in Sect. 5.2.1, we map each layer l_i to each ResNet-18 block.

In the wake stage of task i, we train multiple instances of the model, starting from parameters $\boldsymbol{\theta}_i$, with all possible configurations of $\mathbf{m}_i$: if the deepest frozen layer is l_j, the number of possible values for $\mathbf{m}_i$ is $L-j+1$, with L being the total number of layers. Training is carried out for a single epoch with mini-batch SGD and a learning rate of 0.03. Batch size is set to 32 for Split CIFAR-10 and Tiny-ImageNet$_{1/2}$, and to 8 for Split ImageNet-100. The α hyperparameter in Eq. 5.4 is set to 1, and the N_s dimension of the short-term buffer to 5,000. It is important to mention that, in our implementation, the optimization of Eq. 5.4 (fast model adaptation loss $\mathcal{L}_{\text{fma}}$) and Eq. 5.6 (NREM loss $\mathcal{L}_{\text{NREM}}$) on long-term memory $\mathcal{M}_i$ is carried out on disjoint portions of the whole set of stored samples. In particular, 10% of $\mathcal{M}_i$ is used when optimizing $\mathcal{L}_{\text{fma}}$, while the remaining 90% is used for $\mathcal{L}_{\text{NREM}}$. This separation mitigates the risk of overfitting of $\mathcal{L}_{\text{NREM}}$ on data that will be used, in the wake phase, to determine to which extent model layers should be frozen: indeed, in case of overfitting, the wake phase would encourage model freezing, as it would more easily minimize the corresponding loss term.

[2]Split ImageNet-100 is derived from https://www.kaggle.com/datasets/ambityga/imagenet100
[3]https://pytorch.org/vision/master/_modules/torchvision/models/resnet.html

Target dataset	**CIFAR-10**		**CIFAR-100**		**Tiny-ImageNet$_{1/2}$**		**ImageNet-100**	
Dreaming dataset	*CIFAR-100*		*ImageNet^{aux}*		*Tiny-ImageNet^{D}*		*ImageNet^{NO}*	
Buffer size	**200**	**500**	**200**	**500**	**200**	**500**	**200**	**1000**
GDumb [95]	33.81±1.52	46.01±3.26	6.71±1.55	10.95±0.20	5.74±0.45	9.85±0.50	4.54±0.23	9.79±1.18
↪**WSCL**	**66.85**±1.60	**72.35**±0.72	**14.79**±0.28	**23.68**±0.72	**17.79**±2.34	**26.84**±0.84	**7.70**±0.14	**16.43**±0.10
ER [103]	48.76±0.57	59.75±2.51	14.31±0.47	20.71±0.95	16.25±0.85	21.07±1.43	4.23±0.15	5.05±0.51
↪**WSCL**	**51.86**±4.40	**63.71**±1.35	**16.91**±1.26	**24.25**±0.44	**18.81**±0.48	**23.63**±0.85	**6.01**±0.64	**15.26**±3.59
DER++ [96]	57.35±5.47	69.06±1.24	14.93±2.23	23.26±3.19	16.62±1.76	23.40±1.66	5.95±0.49	8.59±1.11
↪**WSCL**	**63.97**±3.38	**72.33**±0.99	**24.00**±0.94	**31.96**±1.19	**23.70**±0.91	**31.81**±0.70	**6.48**±1.22	**11.70**±0.14
ER-ACE [131]	59.98±2.65	67.17±1.54	25.85±1.93	32.85±4.02	27.81±1.24	32.10±2.21	9.42±0.78	11.58±3.59
↪**WSCL**	**71.15**±2.15	**74.18**±1.28	**34.44**±0.40	**39.78**±0.36	**35.68**±1.18	**41.25**±1.75	**12.51**±0.86	**20.51**±0.56
DualNet [99]	31.31±2.05	43.20±2.81	9.49±0.28	11.04±4.39	16.45±0.39	18.98±0.71	9.78±1.24	16.54±0.85
CoPE [149]	21.20±0.28	23.64±1.56	13.71±0.43	21.03±0.49	16.50±0.62	20.50±0.47	6.23±0.61	12.57±3.69
CLS-ER [153]	34.97±4.83	45.17±4.20	23.85±1.16	30.22±0.76	15.38±0.43	18.19±0.85	9.06±1.32	15.15±1.48

Table 5.1: **Final average accuracy of rehearsal-based methods with and without WSCL in the Class-IL setting.** Results are reported for different buffer sizes. Bio-inspired methods are included for comparison. Bold values indicate the best performance in each column.

In the sleep stage, we train the model using $\mathcal{L}_{\text{NREM}}$ and the $\mathcal{L}_{\text{REM}}$ losses on alternating batches. We perform 10 epochs of training, with the same optimizer settings and hyperparameters as above. All the reported results are computed in the class-incremental setting and reported as mean and standard deviation computed over 5 runs.

5.3.3 Results

We first evaluate how WSCL contributes to classification accuracy of state-of-the-art models. To accomplish this, we select recent rehearsal-based methods, namely, ER [131], DER++ [96], and ER-ACE [103], and compare their performance when the WSCL training strategy is employed, by plugging them in as the $\mathcal{L}$ loss term in Eq. 5.4, 5.6, 5.8. We address rehearsal-based methods only, as WSCL requires a memory buffer to model long-term memory. We further provide a lower bound, consisting of training without any countermeasure to forgetting (*Fine-tune*), and an upper bound given by training all tasks jointly (*Joint*). Results in Table 5.1 show that, on all four benchmarks, WSCL leads to a significant performance gain that varies from about 2 percent points on Split ImageNet-100 to 12 percent points on Split CIFAR-10, substantiating our claims on the importance of leveraging human learning strategies for building better computational methods. Table 5.1 also reports the comparison with: a) DualNet [99], which leverages CLS theory and the same backbone, i.e., ResNet-18; b) CoPE [149] that integrates contrastive learning — another technique inspired by cogni-

Target dataset	**CIFAR-10**		**CIFAR-100**		**Tiny-ImageNet$_{1/2}$**		**ImageNet-100**	
Dreaming dataset	*CIFAR-100*		*ImageNet^{aux}*		*Tiny-ImageNet^{D}*		*ImageNet^{NO}*	
Buffer size	**200**	**500**	**200**	**500**	**200**	**500**	**200**	**1000**
ER [103]	−7.36±5.75	−12.20±0.74	−0.80±0.48	−0.92±0.52	−1.00±0.33	−1.32±0.08	−1.05±0.06	−1.02±0.09
↪**WSCL**	1.68±2.16	6.03±2.58	7.07±1.37	6.49±2.01	12.41±0.98	12.60±0.43	3.82±0.64	3.17±0.73
DER++ [96]	−12.29±0.18	−6.23±5.63	−0.87±0.51	−0.72±0.43	−0.84±0.49	−1.06±0.47	−0.08±0.00	−1.05±0.42
↪**WSCL**	1.06±6.05	2.83±5.27	8.83±0.30	7.52±0.84	12.16±1.25	12.24±1.52	1.78±0.43	2.31±0.07
ER-ACE [131]	−8.58±5.03	−8.97±2.21	−1.01±0.61	−1.02±0.14	−0.73±0.51	−0.94±0.47	−1.04±0.02	−1.17±0.14
↪**WSCL**	0.48±5.53	−1.87±3.12	6.25±0.81	6.06±0.43	8.60±0.96	9.06±1.22	1.83±0.69	1.19±0.67

Table 5.2: **Final forward transfer of rehearsal-based methods with and without WSCL in the Class-IL setting.** Results are reported for different buffer sizes. Bold values indicate the best performance in each column.

tive neuroscience [43] — for better feature transferability to later tasks[4]; c) CLS-ER [153], another method inspired by CLS theory that implements a semantic memory with two separate deep neural networks to model short-term and long-term memory dynamics. We do not include DualPrompt [105] as it uses a large pre-trained ViT [164] as a backbone, leading to an unfair comparison with the simpler ResNet-18. All methods combined with our WSCL strategy improve over DualNet (up to about 40 percent points), CoPE and CLS-ER, demonstrating how mimicking offline brain states improves performance even in a purely discriminative supervised learning regime. We also measure *forward transfer (FWT)* of WSCL, a desirable property in CL that indicates how much a model leverages previous knowledge to learn a new task [66]. Forward transfer is estimated as the average difference between the accuracy of a task when learning it in a CL setting and when learning it from random initialization (details in [66]). Table 5.2 shows how WSCL tends to enhance FWT, turning it from negative to positive values. This is highly remarkable as the majority of existing CL methods show a negative forward transfer.

Furthermore, it is equally important to measure forgetting (the lower, the better) to assess how well an approach tackles no-iid data. Cross-checking results in Table 5.3 with those available in Table 5.1 highlights how WSCL effectively reduces forgetting, while enhancing forward transfer skills and accuracy performance in a way sensibly higher than the baselines.

Previous results have primarily been derived from experiments conducted on common continual learning (CL) benchmarks, where dreaming datasets typically exhibit a semantic distribution shift, meaning their image classes do not overlap with those of the target task. In order to further validate the efficacy of WSCL, we extend our evaluation to scenarios involving domain distribution shifts. Specifically, we assess the performance of WSCL strategy,

[4]Results for DualNet and CoPE are computed using their original implementations and hyperparameters.

Target dataset	**CIFAR-10**		**CIFAR-100**		**Tiny-ImageNet$_{1/2}$**		**ImageNet-100**	
Dreaming dataset	*CIFAR-100*		*ImageNet^{aux}*		*Tiny-ImageNet^{D}*		*ImageNet^{NO}*	
Buffer size	**200**	**500**	**200**	**500**	**200**	**500**	**200**	**1000**
ER [103]	56.66±2.64	43.21±3.41	73.13±1.04	65.49±1.78	62.63±3.43	58.16±1.13	74.04±2.09	73.45±2.06
↪**WSCL**	50.23±4.94	36.04±2.52	68.66±1.04	60.80±1.60	56.71±1.68	50.63±1.37	76.79±0.67	63.93±0.91
DER++ [96]	31.23±2.07	22.63±1.68	67.76±2.87	54.76±5.15	62.15±1.27	50.81±2.56	67.10±2.83	63.63±4.12
↪**WSCL**	35.53±3.28	23.52±2.29	54.63±1.42	46.33±0.81	51.30±2.26	43.91±0.75	59.84±1.59	52.39±2.01
ER-ACE [131]	16.55±2.03	15.21±2.05	38.37±0.86	30.77±6.28	34.41±1.35	28.15±1.96	32.61±3.56	36.44±2.46
↪**WSCL**	11.78±1.61	10.69±2.02	28.20±1.05	25.91±0.89	28.23±1.52	23.29±4.47	27.24±0.55	33.53±0.76

Table 5.3: **Final forgetting of rehearsal-based methods with and without WSCL in the Class-IL setting.** Results are reported for different buffer sizes. Bold values indicate the best performance in each column.

Target dataset	**CORe50**							
Buffer size	**200**				**500**			
ER [103]	19.97±0.02				19.93±0.10			
DER++ [96]	19.88±0.05				19.94±0.06			
ER-ACE [131]	19.90±0.03				19.96±0.02			
Dreaming dataset	*CIFAR-10*		*CIFAR-100*		*Tiny-ImageNet^{D}*		*ImageNet^{NO}*	
Buffer size	**200**	**500**	**200**	**500**	**200**	**500**	**200**	**500**
ER [103] ↪**WSCL**	**36.40**±1.91	**44.03**±0.79	**34.40**±3.33	**51.71**±2.71	**37.30**±2.44	**56.07**±4.00	**43.89**±2.93	**60.94**±2.15
DER++ [96] ↪**WSCL**	**47.43**±2.41	**54.31**±4.28	**41.83**±2.57	**58.83**±1.25	**53.13**±4.99	**63.85**±2.34	**60.43**±3.86	**71.20**±5.01
ER-ACE [131] ↪**WSCL**	**59.46**±3.93	**68.48**±0.67	**61.62**±1.68	**71.12**±0.76	**60.53**±2.19	**71.02**±3.30	**66.85**±1.53	**76.94**±2.57

Table 5.4: **Final average accuracy on CORe50 with multiple dreaming datasets in the Class-IL setting.** Results are reported for rehearsal-based methods, with and without WSCL, across different buffer sizes. Bold values indicate the best performance in each column.

when combined to ER, DER++ and ER-ACE, on the challenging CORe50 dataset [165] with multiple dreaming datasets. As shown in Table 5.4, our approach achieves exceptional performance gains, with improvements of up to 50 percent points, across all dreaming datasets. These results not only underscore the effectiveness of the WSCL strategy but also highlight its applicability to extremely complex datasets where conventional approaches often fall short.

We further expand performance analysis by grounding WSCL to other prominent continual learning methods[5]. For this evaluation, we employ the model that yields the best results, i.e., ER-ACE combined to WSCL (as shown in Table 5.1). As shown in Table 5.5, ER-ACE w/ WSCL (indicated as "Ours") significantly outperforms all existing methods.

[5]Results obtained using the original code released along with the relative papers.

Method	CIFAR-10		CIFAR-100		Tiny-ImageNet$_{1/2}$		ImageNet-100	
Joint	85.15±1.99		61.83±0.47		50.81±1.65		43.39±1.76	
Fine-tune	19.47±0.10		9.11±0.11		13.84±0.55		3.88±0.33	
Buffer-free methods								
LwF [70]	19.33±0.16		8.44±0.39		13.87±1.11		3.83±0.11	
oEWC [73]	18.96±0.24		7.11±0.40		13.87±0.53		3.48±0.18	
SI [74]	19.27±0.23		7.86±1.08		13.12±1.63		3.75±0.35	
Ours (Wake+REM)	**41.58**±3.94		**18.17**±1.04		**25.68**±0.44		**6.27**±0.89	
Buffer-based methods								
Buffer size	**200**	**500**	**200**	**500**	**200**	**500**	**200**	**1000**
A-GEM [92]	19.45±0.25	20.21±0.38	8.34±0.97	8.02±1.25	13.75±0.37	13.56±0.39	4.00±0.20	4.15±0.06
BiC [90]	55.03±1.93	66.24±1.65	21.80±3.81	29.52±2.06	16.26±0.87	12.88±5.50	8.10±2.75	7.03±4.71
DER++ [96]	57.35±5.47	69.06±1.24	14.33±1.97	23.26±3.32	16.62±1.76	23.40±1.66	5.95±0.49	8.59±1.11
ER [103]	48.76±0.57	59.75±2.51	14.31±0.47	20.71±0.95	16.25±0.85	21.07±1.43	4.23±0.15	5.05±0.51
ER-ACE [131]	59.98±2.65	67.17±1.54	25.86±1.94	32.85±4.02	27.81±1.24	32.10±2.21	9.42±0.78	11.58±3.59
FDR [89]	38.72±8.93	31.91±5.08	13.41±1.02	19.34±2.29	17.67±1.04	23.17±1.69	4.44±0.77	3.91±0.22
GDumb [95]	33.81±1.52	46.01±3.26	6.71±1.55	10.95±0.20	5.74±0.45	9.85±0.50	4.54±0.23	9.79±1.18
GEM [66]	21.93±2.04	20.80±0.23	10.40±2.19	14.39±4.11	14.57±0.57	15.20±1.28	4.36±0.11	4.29±0.28
GSS [91]	41.36±6.46	48.83±4.41	11.11±0.63	12.78±0.18	15.92±0.88	18.15±0.61	4.05±0.42	4.46±1.20
iCaRL [88]	64.52±1.18	60.94±1.34	14.22±0.22	16.01±0.52	20.40±0.36	22.68±0.30	10.40±0.20	11.17±0.79
LUCIR [93]	53.48±7.62	63.01±3.40	24.06±1.81	32.54±1.16	22.65±1.18	32.15±0.88	6.08±0.32	13.19±0.32
RPC [98]	49.37±1.47	55.19±2.73	14.38±1.36	21.01±0.95	16.58±0.52	20.95±0.59	4.13±0.16	5.83±0.30
Ours	**71.15**±2.15	**74.18**±1.28	**34.44**±0.40	**39.78**±0.36	**35.68**±1.18	**41.25**±1.75	**12.51**±0.86	**20.51**±0.56

Table 5.5: **Final average accuracy of state-of-the-art continual learning methods in the Class-IL setting.** Results are reported for different buffer sizes where applicable. Bold values indicate the best performance in each column.

Notably, when excluding the buffer for training ER-ACE w/ WSCL (which means using the model without NREM stage, indicated in Table 5.5 as Wake+REM), it achieves a substantial performance improvement, from approximately 12% (on Tiny-ImageNet$_{1/2}$) to about 23% on Split CIFAR-10, over existing buffer-free methods, namely, LwF [70], SI [74], and oEWC [73].

5.3.4 Ablation study

In our ablation study, we adopt ER-ACE as the baseline, as it achieves the best performance in Table 5.1. We evaluate its performance on both the Tiny-ImageNet$_{1/2}$ and CORe50 datasets, examining scenarios involving semantic and domain shifts. Initially, we conduct ablations on the processing phases of WSCL. Results in Table 5.6 demonstrate that NREM and REM sleep states contribute equally to the final model performance across both benchmarks.

Target dataset	**Tiny-ImageNet$_{1/2}$**		**CORe50**	
Dreaming dataset	*Tiny-ImageNet^D*			
Method	**FAA**	**FWT**	**FAA**	**FWT**
Only Wake	4.70	-0.93	15.00	-7.49
Wake + REM	25.68	11.89	23.82	-11.80
Wake + NREM	27.61	-0.67	54.15	-12.61
Wake + REM + NREM	35.68	8.60	60.53	-5.62

Table 5.6: **Ablation on the WSCL processing stages.** Results refer to ER-ACE on Tiny-ImageNet$_{1/2}$ and on CORe50 datasets with buffer size of 200.

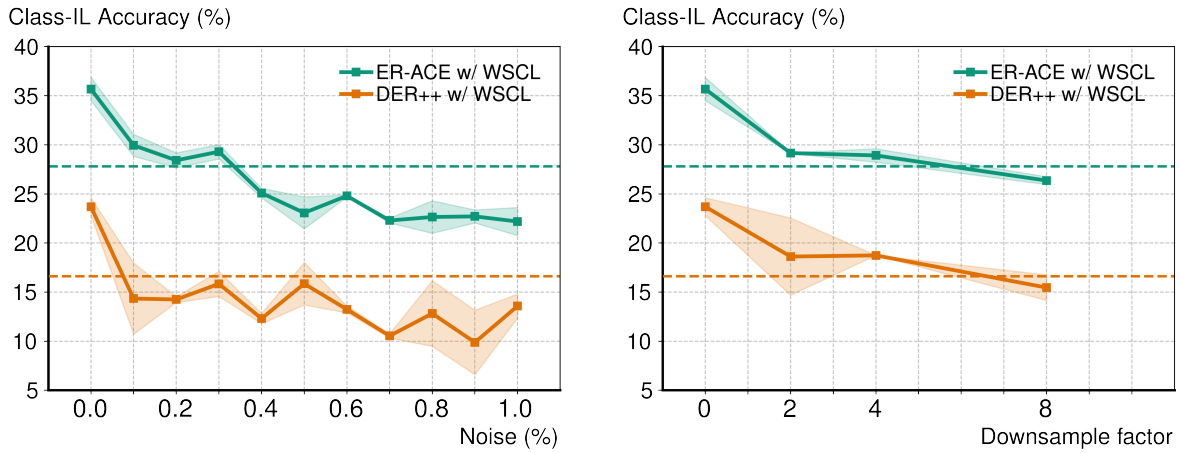

Figure 5.2: **Impact of dreaming quality**, in terms of noise (left) and image resolution (right). Results refer to ER-ACE and DER++ with WSCL (solid lines) and without it (dotted line).

Notably, on the Tiny-ImageNet$_{1/2}$ dataset, the REM phase exhibits positive forward transfer, aligning with cognitive neuroscience findings indicating REM's role in priming brain synapses for future experiences [161, 162]. This pattern of forward transfer is observed across the majority of the tested datasets (see Table 5.2), except for CORe50. This deviation can be attributed to the nature of the CORe50 benchmark, which inherently restricts feature reuse across tasks due to its strong background bias.

We then evaluate the impact of the quality of dreaming, by adding Gaussian noise (at different percentages) and reducing the spatial resolution of dreaming samples. Fig. 5.2 indicates that WSCL still outperforms the baseline when dreaming images are affected by noise up to 30% or scaled down by 6×, suggesting that the role of REM stage in consolidating knowledge is mostly independent from the visual details of the dreamed samples, which merely serve to learn additional reusable features.

Dataset	**CIFAR-10**	**CIFAR-100**
DER++ [96]	67.38	28.01
↪CaSpeR [147]	69.11	32.16
↪**WSCL**	**72.18**	**35.00**
ER-ACE [131]	66.13	34.99
↪CaSpeR [147]	69.58	36.70
↪**WSCL**	**73.56**	**39.33**

Table 5.7: **Comparison between WSCL and CaSpeR in terms of final average accuracy in the Class-IL setting.** Results are reported only for DER++ and ER-ACE, as the remaining methods on which CaSpeR is applicable manipulate future logits, hindering WSCL application. Dreaming datasets are CIFAR-100 and $ImageNet^{aux}$, respectively, for CIFAR-10 and CIFAR-100 benchmarks. Buffer size is 500.

We further investigate the impact of the size of the dreaming dataset on the results. Figure 5.3 illustrates how the dreaming stage allows for enhanced performance even when the additional dreaming dataset is reduced by approximately 70%.

Dreaming in WSCL serves as an implicit regularizer for the learned latent space within the model, maintaining consistent partitioning across classes, especially on low-dimensional buffers. In this study, we contrast our WSCL approach with a regularization method, namely CaSpeR [147], which explicitly encourages partitioning behavior in the learned space by imposing constraints on its Laplacian spectrum. To ensure a fair comparison, we replicated the experimental setup detailed in [147], conducting tests over the same number of epochs (20) and batch size of 64 on the Split CIFAR-10 and Split CIFAR-100 datasets with buffer 500 (as these represent the intersection between [147] evaluation and ours). The results, presented in Table 5.7, highlight how WSCL outperforms CaSpeR as a regularizer.

We finally assess the efficiency aspects of WSCL. Indeed, the human brain is capable of performing complex tasks with remarkable speed and accuracy, at a relatively low energy cost: cerebral parallel processing architecture, plasticity, and ability to adapt to changing environments are all factors that contribute to its efficiency [166, 167]. In WSCL, efficiency is encouraged in the wake stage, by letting the model selectively freeze different portions of the network: this is analogous and consistent to cognitive neuroscience evidence that a synchronization of neural activity across different brain regions and changes in the balance between excitation and inhibition enable efficient processing [168, 169].

Fig. 5.4 shows the most frequent (over 10 different runs) set of frozen backbone layers at each task, when training ER-ACE with WSCL on Tiny-ImageNet$_{1/2}$, as well as the total number of performed parameter updates using the training procedure presented in Sect. 5.3.2.

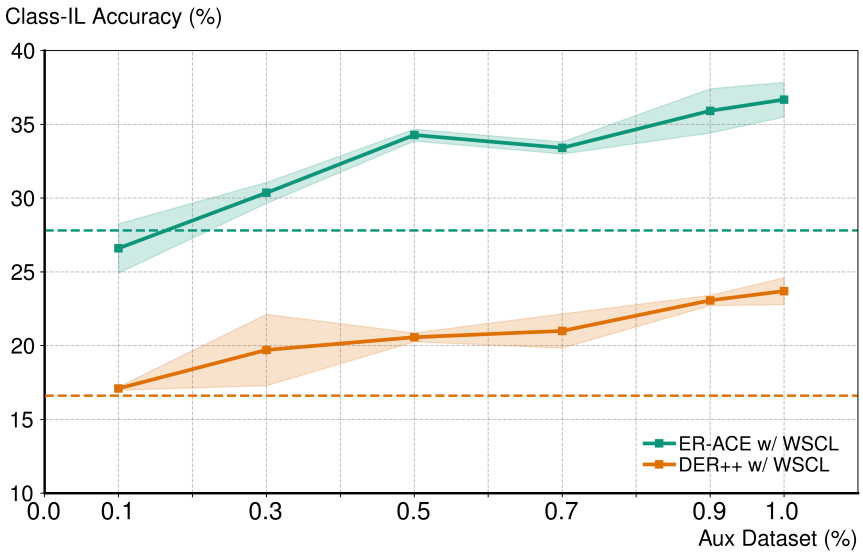

Figure 5.3: **Impact of dreaming dataset dimension**. Results refer to ER-ACE and DER++ with WSCL (solid lines) and without it (dotted line).

It is important to note, however, that the freezing strategy employed during the wake stage of WSCL depends on the specific continual learning method and the target dataset. Figure 5.5 illustrates the predominant freezing scheme of ER-ACE when evaluated on the Split ImageNet-100 dataset, as well as the resulting efficiency. Unlike Tiny-ImageNet$_{1/2}$, where ER-ACE typically freezes almost all layers after the completion of the first task, on Split ImageNet-100, ER-ACE gradually freezes the layers of its backbone network until task τ_5. Despite this more gradual freezing strategy, the efficiency gain achieved is approximately 17.14%, indicating fewer updates compared to the baseline model trained without the wake-sleep strategy in WSCL. Thus, WSCL's training procedure reduces the overall number of updates for the entire training of the ResNet-18 model, by a quantity that tends to increase with the number of training epochs (from 2% to about 17% less updates), thus confirming the suitability of the wake stage in supporting efficient training.

Furthermore, we conduct an analysis to assess how the freezing strategy scales with the backbone size and the number of epochs, focusing on the performance of ER-ACE on the Tiny-ImageNet$_{1/2}$ dataset. Our results, reported in Table 5.8, indicate that the efficiency gains achieved through selective freezing tend to increase with both the number of epochs and the depth of the considered model. For instance, when training ResNet-18 for 100 epochs, we observed a reduction of approximately 17% in parameter updates compared to the baseline model. In contrast, with ResNet-151 and the same number of epochs, the reduction in parameter updates reached about 24%. Notably, the selectively freezing strategy primarily targets reducing training computation costs and has minimal impact on performance during

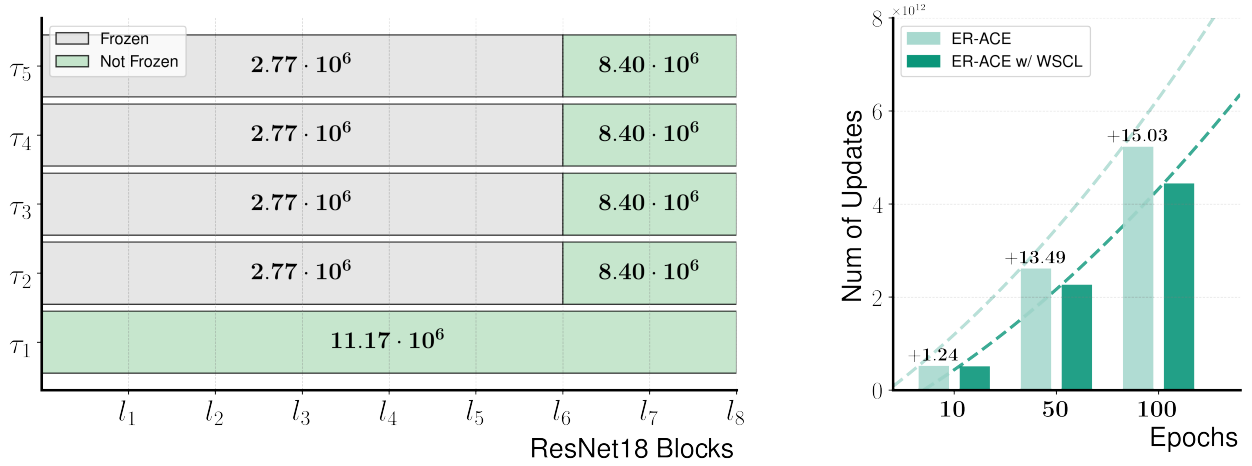

Figure 5.4: **WSCL model efficiency**: Left: the most frequent automatically learned freezing scheme (values within bars are number of parameters) during the wake phase for ER-ACE on Tiny-ImageNet$_{1/2}$. Right: number of parameter updates for the whole training of ER-ACE with and without WSCL on Tiny-ImageNet$_{1/2}$ (from 10 epochs to 100 training epochs).

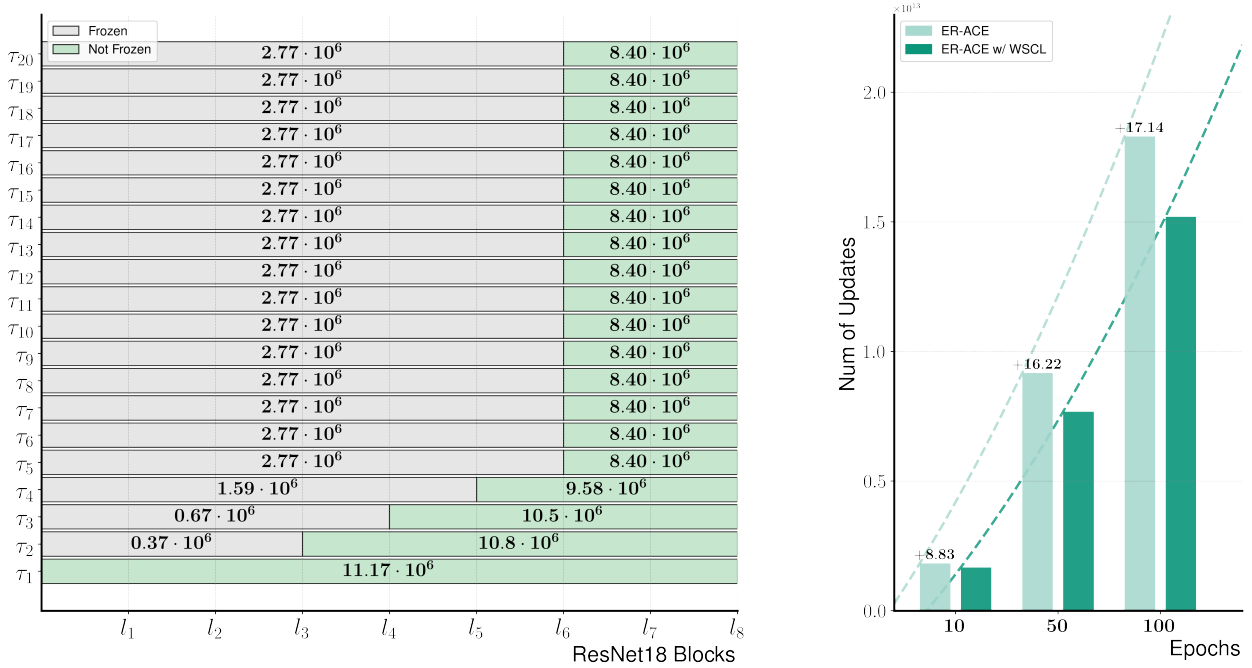

Figure 5.5: **WSCL model efficiency**: Left: the most frequent automatically learned freezing scheme (values within bars are number of parameters) during the wake phase for ER-ACE on Split ImageNet-100. Right: number of parameter updates for the whole training of ER-ACE with and without WSCL on Split ImageNet-100 (from 10 epochs to 100 training epochs). The numbers above the green bars represent the improvement in percent points with respect to the baseline alone.

Backbone	**ResNet-18**					
# epochs	**10**		**50**		**100**	
	FAA (↑)	**ΔU** (↓)	**FAA** (↑)	**ΔU** (↓)	**FAA** (↑)	**ΔU** (↓)
Selective Freezing	41.25	+8.83	44.50	-16.22	45.22	-17.14
No Freezing	41.22		44.24		44.26	
Backbone	**ResNet-151**					
# epochs	**10**		**50**		**100**	
	FAA (↑)	**ΔU** (↓)	**FAA** (↑)	**ΔU** (↓)	**FAA** (↑)	**ΔU** (↓)
Selective Freezing	35.96	+44.73	44.66	-16.43	43.68	-24.07
No Freezing	38.40		45.36		42.62	

Table 5.8: **Impact of the selective freezing strategy on ER-ACE with WSCL.** We report final average accuracy and parameter updates (ΔU), i.e., the percentage change in the number of parameter updates induced by selective freezing relative to the no-freezing baseline. Results are computed on Tiny-ImageNet$_{1/2}$ with buffer size 500 for different numbers of epochs.

inference. These findings underscore the scalability and effectiveness of the wake stage in WSCL for achieving efficient continual learning across a range of model architectures and training durations.

5.4 Discussion

The integration of Complementary Learning Systems (CLS) theory and sleep mechanisms in artificial neural networks holds great potential for enhancing continual learning capabilities. Inspired by the interaction between the hippocampus and neocortex in humans, Wake-Sleep Consolidated Learning (WSCL) introduces a sleep phase that mimics offline brain states during which memory consolidation and synaptic reorganization occur. By leveraging the wake phase for fast adaptation and episodic memory formation, and the sleep phase for memory consolidation and dreaming, WSCL shows superior performance compared to prior work on various benchmarks. Importantly, WSCL achieves positive forward transfer, exhibiting the ability to prepare synapses for future knowledge. These findings highlight the importance of all three stages — wake, NREM and REM — in supporting network plasticity and reducing forgetting for improved learning and memory.

The main limitations of WSCL concern memory and dreaming modeling techniques,

which currently rely on conventional rehearsal methods to facilitate memory retention and on the employment of external datasets for generating dream-like experiences. With regard to memory modeling, an interesting direction is to explore more nuanced and dynamic approaches that better capture the intricacies of memory formation, storage, and retrieval, by also devising mechanisms to account for memory decay and interference. Likewise, dream modeling can move beyond the current reliance on external datasets. To this end, Chapter 6 introduces a mechanism based on generative models capable of simulating dream-like experiences from the network's existing knowledge and latent representations. This design aims to enable the generation of diverse, creative, and contextually relevant dream scenarios, improving their realism.

5.5 Publications

Sorrenti, A., Bellitto, G., Salanitri, F. P., Pennisi, M., Palazzo, S., & Spampinato, C. (2024). "Wake-Sleep Consolidated Learning". *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 7, pp. 12668-12679, July 2025.

Chapter 6

Dream2Learn: Structured Generative Dreaming

During sleep, the brain replays and reorganizes recent experiences, generating dream content grounded in waking life and potentially supporting subsequent knowledge acquisition [38, 39, 40]. In contrast, deep learning models in continual learning often struggle to effectively balance stability and plasticity, limiting both long-term knowledge retention and the capacity for adaptation [149, 170]. Among existing approaches, rehearsal-based strategies are widely used due to their ability to stabilize learning by replaying stored examples. Yet, despite their effectiveness, such approaches diverge significantly from how the human brain consolidates memory. Rather than relying on the exact replay of past experiences, the brain engages in generative processes during dreaming, recombining perceptual elements from daily life to construct novel and plausible future scenarios [161, 162]. This process allows for efficient knowledge reinforcement, enabling the brain to improve generalization and anticipate future challenges. Translating this process into artificial neural networks is non-trivial, as it requires the ability to synthesize meaningful and structured representations of the previously learned knowledge without relying on external supervision.

To accomplish this task, in this chapter we present Dream2Learn (D2L), a generative dreaming process that synthesizes training samples directly from the classifier's internal representations. Unlike WSCL, which relies on surrogate real data to simulate dreams, and other sleep-based approaches that primarily reinforce existing representations [171, 157], D2L autonomously constructs future-adaptive representations (the *dreams*, indeed), ensuring task relevance and enhancing the model's ability to generalize to new tasks.

As illustrated in Fig. 6.1, D2L generates structured dreamed classes that serve as intermediate representations, facilitating continual learning. Instead of merely blending past class features, these dreamed samples form coherent yet distinct new concepts, expanding

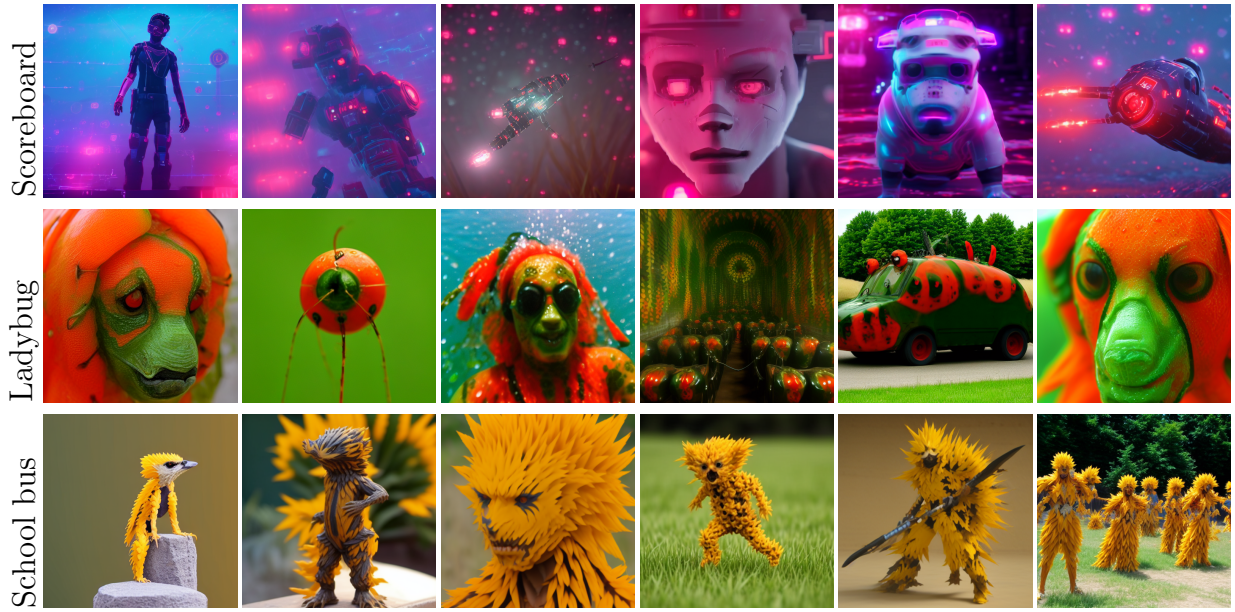

Figure 6.1: **Dreamed classes generated by D2L**. Examples of dreamed classes synthesized from their corresponding real classes (left). These samples emerge as semantically distinct yet structurally coherent representations in the generator's latent space, forming intermediate concepts that enhance the continual classifier's generalization to future tasks.

the representation space in a way that supports future task adaptation. By integrating "dreamed classes" into training, the classifier learns high-level reusable features, reinforcing forward transfer while mitigating catastrophic forgetting. This process mirrors the role of REM sleep, where synthetic experiences help refine learned representations, maintaining long-term adaptability as new data distributions emerge. Our experiments on Split Mini-ImageNet, Split ImageNet-100, and Split ImageNet-R show that our strategy significantly boosts performance when integrated with standard continual learning methods.

6.1 Related work

Generative Replay (GR) [81, 172, 173] has emerged as an alternative to buffer-based experience replay by synthesizing past samples, but early methods often faced mode collapse and underperformed compared to traditional replay. Although DDGR [85], SDDR [174], and DiffClass [175] improve sample fidelity, the role of GR remains retrospective: the generator mainly acts as a memory proxy, reconstructing prior distributions for rehearsal rather than proactively reorganizing representations. Inspired by cognitive neuroscience, several works explore memory mechanisms modeled on brain function. DualNet [99] and DualPrompt [105] introduce parallel learning pathways, while CLS-ER [153] and FearNet [176] implement short-

and long-term memory systems. These approaches focus on stabilizing representations during learning but do not incorporate offline processes for restructuring knowledge. Sleep-based learning offers a complementary perspective, drawing from evidence that wake-sleep cycles refine memory representations [177, 43]. Sleep Replay Consolidation [171] applies Hebbian plasticity, and SIESTA [157] introduces intermittent consolidation to support online learning.

The approach presented in Chapter 5 alternates wake and sleep cycles to emulate dream-like replay mechanisms for memory consolidation. However, instead of generating internal experiences during sleep phases, it shapes the latent space of the classifier using pre-defined representations, limiting the biological plausibility of the dreaming process.

D2L reframes generation as a prospective mechanism: rather than replicating past data for rehearsal, generation is used to proactively structure the representation space towards upcoming tasks. It introduces a self-sufficient generative dreaming mechanism, by generating additional training signals from the classifier's internal representations, ensuring task relevance and autonomous dreaming. Through soft prompt optimization, it identifies semantically distinct yet structurally coherent classes in the diffusion model's latent space, which act as intermediate anchors that bootstrap future learning dynamics. Thus, unlike GR techniques that focus on reconstructing or augmenting past distributions [174, 175], D2L actively shapes future-adaptive representations, transforming dreaming into a mechanism for fostering forward knowledge transfer and long-term retention, enabling the classifier to adapt more effectively to unseen tasks.

6.2 Method

We formulate our continual learning setting as the problem of training a model F, with parameters $\boldsymbol{\theta}$, over a sequence of T visual classification tasks $\{\tau_1, \ldots, \tau_T\}$, with each task τ_t associated to a dataset $\mathcal{D}_t$. Each observation $(\mathbf{x}, y) \sim \mathcal{D}_t$ consists of an image $\mathbf{x} \in \mathcal{I}$ and a class label $y \in \mathcal{C}_t$, with $|\mathcal{C}_t| = n_t$. The model's output layer has as many neurons as the total number of classes, i.e., $\sum n_t$.

We also assume the availability of a replay buffer $\mathcal{M}$, where we store a limited number of samples from past tasks for rehearsal, and of a pre-trained and frozen image generator G. Our approach requires that G can be conditioned from both textual prompts (with the possibility of adding learnable tokens) and input images; these requirements are easily satisfied by standard text-conditioned latent diffusion models (LDM) — e.g., Stable Diffusion conditioned by CLIP embeddings [178, 179] — with an image adapter [180][1]. Formally,

[1] We use h94/IP-Adapter

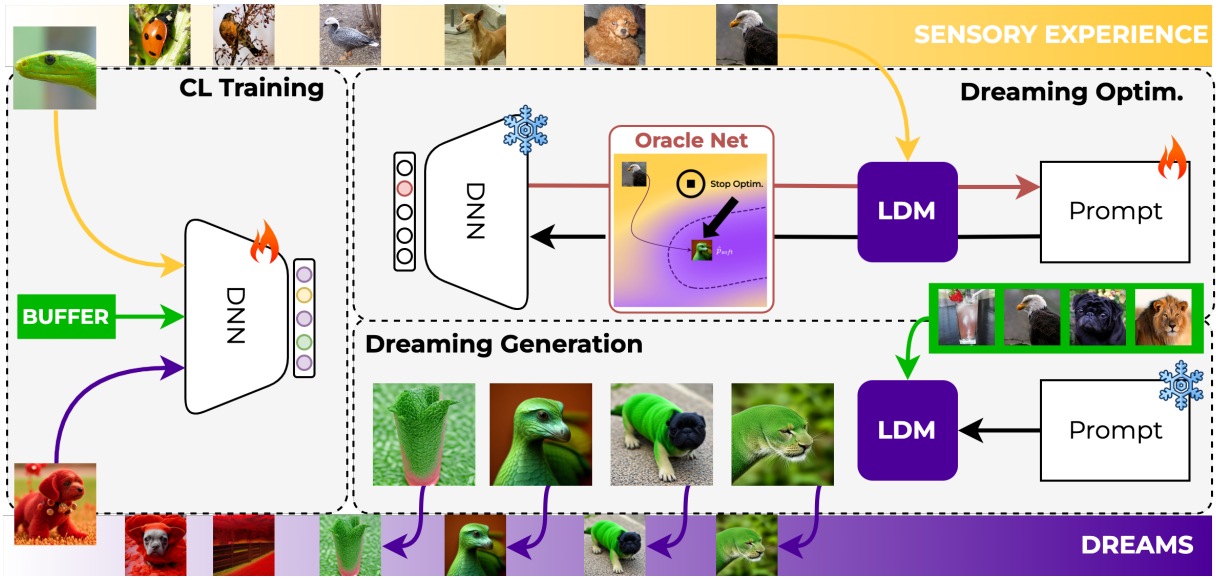

Figure 6.2: **Overview of Dream2Learn.** (1) During CL training, a deep neural network (DNN) learns from real sensory images (the current task distribution plus the buffer) and dreamed samples produced by a latent diffusion model (LDM). (2) The dreaming optimization process refines the LDM prompts, with an Oracle Network providing a stopping criterion that prevents collapse. (3) Prompts generate auxiliary classes: dreamed samples are not buffered, but rather enrich the representation space with coherent latent clusters that foster knowledge reuse and adaptation (see Algorithm 1)

$G : \mathcal{I} \times \mathcal{P} \to \mathcal{I}$, with $\mathcal{P}$ being the space of sequences of textual token embeddings[2]. We employ G, with appropriate conditioning, to synthesize dream images from past knowledge, thus creating an auxiliary synthetic data stream for preparation to future tasks. At the beginning of our procedure, the model F is trained to learn how to perform task τ_1. Since no knowledge is initially present (as $\boldsymbol{\theta}$ is randomly initialized), we bootstrap the model by training it on task data $\mathcal{D}_1$, optimizing a cross-entropy loss:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{CE}}(F, \mathcal{D}_1) = -\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_1}\Big[\log p(y|\mathbf{x};\boldsymbol{\theta})\Big], \tag{6.1}$$

with $p(y|\mathbf{x};\boldsymbol{\theta})$ being the likelihood of the correct class, given model parameters $\boldsymbol{\theta}$. During this bootstrap phase, we also populate the buffer $\mathcal{M}$ via reservoir sampling [66].

The bootstrapped classifier can now be used to synthesize new classes as "dream" variants of what the model has seen up to this point. Dream classes are created by optimizing a learnable prompt $\mathbf{p}_c$ for each class $c \in \mathcal{C}_1$, such that $G(\mathbf{x}, \mathbf{p}_c)$ transforms an input image $\mathbf{x}$

[2]In practice, G is also made stochastic by receiving a random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is omitted for brevity.

into a "dreamed" versions that vaguely resemble target class c, thereby creating a synthetic distribution for a novel "dream class". The details about this optimization process are given in Sec. 6.2.1. Using this procedure, at task τ_1 we introduce n_1 dream classes, i.e., as many as the current task's actual classes. We indicate with $\mathcal{D}_1^d$ the distribution of dream classes defined at this stage. Let $\mathcal{D}_{\tau_1}$ be the mixture distribution which equally samples from real data $\mathcal{D}_1$ and from the dream distribution $\mathcal{D}_1^d$. The classifier F is then fine-tuned on $\mathcal{D}_{\tau_1}$, replacing $\mathcal{D}_1$ in Eq. 6.1. On subsequent tasks τ_t, $t > 1$, we can exploit the model's knowledge on dream classes to ease its learning of new classes. At the beginning of τ_t, we forward samples from each new class $c \in \mathcal{C}_t$ through the model, and map c to the output dream neuron with the largest average likelihood, as in [163]. This allows to bootstrap each class maximizing the reuse of relevant features and preventing disrupting weight updates (details on this procedure in Sec. 6.2.1). The dream classes corresponding to the assigned classification heads are removed. We then train F on task data $\mathcal{D}_t$ and on $\mathcal{D}_{t-1}^{d*}$, the residual dream distribution obtained from $\mathcal{D}_{t-1}^d$ by removing the discarded dream classes $\mathcal{D}_t^r$, optimizing the following:

$$\min_{\boldsymbol{\theta}} \Big[\mathcal{L}_{\text{CE}}(F, \mathcal{D}_{t-1}^{d*} \cup \mathcal{D}_t) + \mathcal{L}_{\text{CL}}(F, \mathcal{D}_t, \mathcal{M}) \Big], \tag{6.2}$$

where $\mathcal{L}_{\text{CL}}$ is an additional continual learning loss that counters forgetting and explicitly leverages the replay buffer $\mathcal{M}$ for rehearsal. In practice, when sampling from the dream distributions $\mathcal{D}_{t-1}^{d*}$, we employ items stored in the buffer as input conditions to the generator G, to increase variability in the dreamed images. Importantly, dreamed samples are never added to $\mathcal{M}$ (refer to Appendix 6.3.5); only their prompts are retained as part of a persistent dream inventory. After training on task τ_t and storing rehearsal samples into $\mathcal{M}$, we update the dream inventory for the next task, by optimizing a new set of prompts $\{\mathbf{p}_c \mid c \in \mathcal{C}_t\}$, corresponding to new dream class distributions. The set of n_t newly-generated dream distributions is used to replace an equal number of existing dreaming classes using again the mapping strategy in [163].

The proposed method is detailed in Algorithm 1. For clarity of presentation, the pseudo-code omits the initial and final tasks. The initial task lacks dreaming classes, reducing the training loss to Eq. 6.1, while the final task does not perform dreaming generation and optimization, since these steps are unnecessary.

6.2.1 Dreaming optimization and mapping

The dreaming optimization process for task τ_t consists of learning a proper conditioning for generator G, in order to synthesize samples of novel concepts, expanding the model's representation space while preserving feature reuse.

Algorithm 1 Dream2Learn (D2L)

Notation

T, the number of tasks
$\mathcal{C}_t$, the classes of task t
F, the continual classifier
G, the generator
$\mathcal{M}$, the memory buffer
$\mathcal{D}_t$, the real data distribution at task t
$\mathcal{D}^d_{t-1}$ the *dream* distribution used during continual training at task t
$\mathcal{D}^r_t$, the distribution of the dream classes to be *removed* at task t
$\mathcal{D}^{d*}_{t-1}$, the residual *dream* classes after mapping at task t
$\mathbf{x}$, a real image
$\mathbf{x}^d$, a generated *dream* image
$\mathbf{p}_c$, the learnable prompt associated with class c
$\mathcal{D}^d_{t,c}$, the distribution of *dreams* generated after task t from class c
$\mathcal{D}^d_{t,\mathcal{C}_t}$, the distribution of *dreams* generated after task t from classes $\mathcal{C}_t$
$\mathcal{D}^d_t$, the distribution of all *dreams* after task t

for $t = 2$ **to** $T - 1$ **do**
  $\mathcal{D}^r_t \leftarrow Mapping(F_{\boldsymbol{\theta}}, \mathcal{D}_t)$ ▷ *real classes mapping* (Sec. 6.2)
  $\mathcal{D}^{d*}_{t-1} \leftarrow \mathcal{D}^d_{t-1} \setminus \mathcal{D}^r_t$
  for all epochs **do** ▷ *CL training*
    $loss \leftarrow \mathcal{L}_{CE}(F, \mathcal{D}^{d*}_{t-1} \cup \mathcal{D}_t) + \mathcal{L}_{\text{CL}}(F, \mathcal{D}_t, \mathcal{M})$ ▷ Eq. 6.2
    $\mathcal{M} \leftarrow ReservoirSample(\mathcal{M}, \mathcal{D}_t)$
    update $\boldsymbol{\theta}$
  end for
  for all $c \in \mathcal{C}_t$ **do** ▷ *dreaming optimization* (Sec. 6.2.1)
    repeat
      $\mathbf{x}^d \leftarrow G(\mathbf{x}, \mathbf{p}_c)$
      $loss \leftarrow \mathcal{L}_{CE}(F, (\mathbf{x}^d, c))$ ▷ Eq. 6.3
      update $\mathbf{p}_c$
      $stop \leftarrow Oracle(\mathbf{x}, \mathbf{x}^d)$ ▷ Sec. 6.2.2
    until *stop*
    $\mathcal{D}^d_{t,c} \leftarrow Generate(G, \mathcal{M}, \mathbf{p}_c)$ ▷ *dreaming generation* (Sec. 6.2.1)
  end for
  $\mathcal{D}^d_{t,\mathcal{C}_t} \leftarrow \bigcup_c \mathcal{D}^d_{t,c}$
  $\mathcal{D}^r_t \leftarrow Mapping(F, \mathcal{D}^d_{t,\mathcal{C}_t})$ ▷ *dream classes mapping* (Eq. 6.4)
  $\mathcal{D}^d_t \leftarrow (\mathcal{D}^{d*}_{t-1} \setminus \mathcal{D}^r_t) \cup \mathcal{D}^d_{t,\mathcal{C}_t}$
end for

For each real task class $c \in \mathcal{C}_t$, we aim to generate a corresponding dreamed class c^d that is distinct from all real classes, while contributing to a structured representation in the latent space. To achieve this, we optimize a learnable prompt $\mathbf{p}_c$ that conditions the generator G

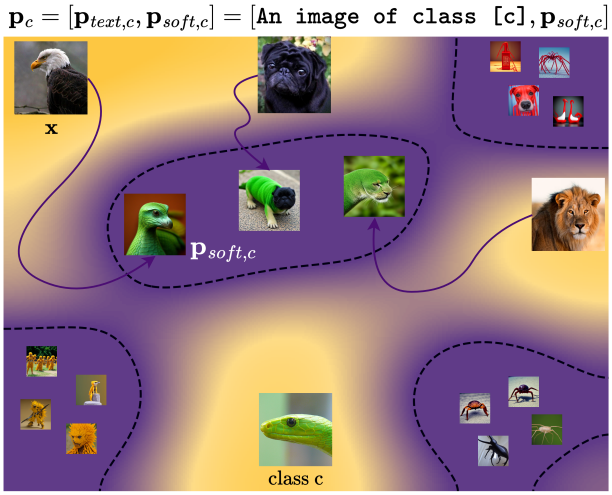

Figure 6.3: **Visualization of the dreaming optimization process in the latent space of a LDM.** Given a real sample $\mathbf{x}$, the optimization refines the soft prompt $\mathbf{p}_{\text{soft},c}$ to steer the diffusion model towards generating a dreamed counterpart that aligns with the target class c (e.g., a green mamba in this example). The dreaming process explores latent regions where images are visually similar yet distinct from target classes, forming novel intermediate classes (violet zones).

to synthesize samples of class c^d. Our objective is to identify a transformation trajectory in the LDM latent space such that, given an arbitrary real image $\mathbf{x}$ as input to G, the learned prompt $\mathbf{p}_c$ guides G to generate a dreamed version $\mathbf{x}^d$ that shares some characteristics with class c, while remaining distinct enough to not be classified as c by the model F. This ensures that the dreamed samples populate a structured latent space region that remains visually coherent but semantically separated from real classes. Formally, we structure the dream class condition $\mathbf{p}_c = [\mathbf{p}_{\text{soft},c}, \mathbf{p}_{\text{text},c}]$ with $\mathbf{p}_{\text{text},c}$ being the fixed text prompt describing the transformation for class c, as: "`An image of class` $[c]$", and $\mathbf{p}_{\text{soft},c}$ being a learnable soft prompt vector optimized to refine the conditioning for class c.

Due to the stochasticity of G, multiple dreamed samples can be generated from the same real image $\mathbf{x}$ and condition $\mathbf{p}_c$. Fig. 6.3 shows the dreaming optimization process in the LDM latent space.

Prompt optimization is performed by minimizing the cross-entropy loss:

$$\min_{\mathbf{p}_{\text{soft},c}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_i\setminus\mathcal{D}_i^c}[-\log p(c|G(\mathbf{x},\mathbf{p}_c);\boldsymbol{\theta})], \tag{6.3}$$

where $\mathcal{D}_i^c$ is the subset of real samples belonging to class c, excluded from the optimization process: this ensures that optimization is not conditioned on images already belonging to the target distribution, allowing the process to gradually converge toward it while generating

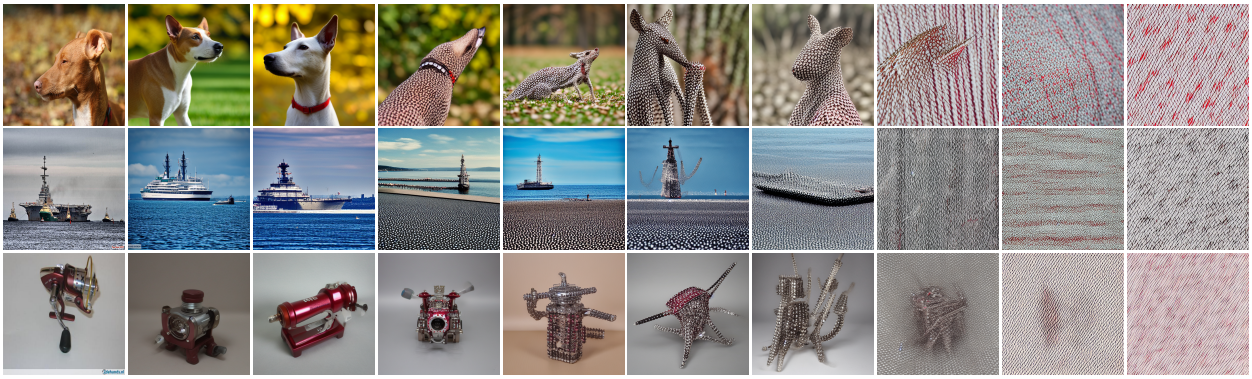

Figure 6.4: **Examples of dreaming optimization trajectories showing collapse**. From left to right, the images depict different stages of the optimization process. Each row illustrates the evolution of three example images throughout the same prompt optimization. Initially, the generated samples maintain meaningful variations. However, as optimization progresses, they become increasingly similar, reducing diversity and leading to less effective representations.

novel yet structured concepts. As optimization progresses, dreamed samples populate a distinct but structured latent region, allowing future tasks to benefit from enhanced feature reuse and transferability.

Dreaming optimization produces a set of conditioning prompts $\{\mathbf{p}_c \mid c \in \mathcal{C}_t\}$ for each class of task τ_t. Next, we determine the output neurons to which the new dream classes $\mathcal{D}^d_{t,\mathcal{C}_t} = \bigcup_c \mathcal{D}^d_{t,c}$ should be mapped, replacing a subset of dream classes from past tasks. The rationale for this step is to map new classes over "similar" past dream classes, to ensure a smooth integration and prevent high gradients during training. Thus, let $\mathcal{C}$ be the set of all output model neurons, and $\mathcal{C}_{\text{real}} = \bigcup_{i \leq t} \mathcal{C}_i$ the set of outputs assigned to real past tasks. We compute the set of possible destination neurons for the new dream classes as $\mathcal{C}_{\text{avail}} = \mathcal{C} \setminus \mathcal{C}_{\text{real}}$. Then, we obtain the output neuron c_{out} for dream class c^d as:

$$c_{\text{out}} = \underset{c \in \mathcal{C}_{\text{avail}}}{\arg\min}\, \mathbb{E}_{\mathbf{x} \sim G_{c^d}}[-\log p(c|\mathbf{x}; \boldsymbol{\theta})], \tag{6.4}$$

where G_c is the distribution associated to samples produced by G when conditioned with $\mathbf{p}_c$. In practice, c^d replaces the dream class to which it is more "aligned" in terms of classification likelihood.

6.2.2 Oracle-guided dreaming optimization

One key challenge in the dreaming process is determining when to stop optimizing the soft prompt $\mathbf{p}_{\text{soft},c}$ to avoid collapse or excessive task-specific bias, as illustrated in Fig. 6.4. If

optimization continues indefinitely toward the convergence of Eq. 6.3, the generated samples risk becoming redundant or overfitting to the current task, reducing their effectiveness for future learning. To prevent this, we introduce an oracle network that predicts the optimal stopping point by evaluating whether further refinement of $\mathbf{p}_{\text{soft},c}$ contributes to meaningful latent representation learning. The oracle is trained on a separate dataset $\mathcal{D}_O$, where stopping decisions are labeled based on the quality of generated dreams.
Formally, we define the oracle network O, which takes as input a sequence of feature vectors extracted over a temporal window of k optimization steps and provides a binary decision:

$$O(\mathbf{Z}_t) \in \{0, 1\} \tag{6.5}$$

where $\mathbf{Z}_t$ is the aggregated feature matrix over the last k generated samples:

$$\mathbf{Z}_t = [\mathbf{z}_{t-k+1}, \mathbf{z}_{t-k+2}, \ldots, \mathbf{z}_t] \tag{6.6}$$

The components of each vector $\mathbf{z}_i \in \mathbb{R}^4$ are the following quantities, computed using the generator G_c conditioned by $\mathbf{p}_{\text{soft},c}$ at the i-th optimization iteration of Eq. 6.3:

1. $\mathbb{E}_{\mathbf{x}}[\text{sim}(\mathbf{x}, G_c(\mathbf{x})]$, with $\text{sim}(\cdot)$ measuring the structural similarity between the generated image $G_c(\mathbf{x})$ and its conditioning image $\mathbf{x}$, ensuring that the generated image maintains structural coherence;

2. $\mathbb{E}_{\mathbf{x}}[f(\mathbf{x})^\intercal f(G_c(\mathbf{x}))]$, i.e., the dot product between feature embeddings f extracted by the classifier F, capturing the alignment between the representations of $\mathbf{x}$ and $G_c(\mathbf{x})$;

3. $\mathbb{E}_{\mathbf{x}}[\text{Q}(G_c(\mathbf{x}))]$, where Q computes the CLIP-based Image Quality Assessment [181], evaluating the perceptual quality of the generated image;

4. $\mathbb{E}_{\mathbf{x}}\Big[\sigma\Big(f(G_c(\mathbf{x}))\Big)\Big]$, i.e., the standard deviation of the feature embeddings, capturing the diversity within generated samples.

Once trained, the oracle network O is frozen and used across all tasks to determine when to stop the optimization of $\mathbf{p}_{\text{soft},c}$.

6.3 Experimental results

6.3.1 Datasets and metrics

We evaluate D2L on three continual learning benchmarks: Split Mini-ImageNet [59], Split ImageNet-100 [61], and Split ImageNet-R [62]. In our experimental setup, half of the classes

in each dataset are used for the first task, while the remaining classes are equally split across the subsequent tasks. In particular, excluding the first task, the Mini-ImageNet and ImageNet-100 datasets consist of 10 tasks with 5 classes each, whereas the ImageNet-R dataset consists of 5 tasks with 20 classes each. To evaluate performance, we report *final average accuracy* (FAA) and *final forward transfer* (FFWT) after training on the last task in the class-incremental setting.

6.3.2 Training procedure

In terms of training procedure, we adopt ResNet-18 [111] as the backbone and train for 10 epochs per task using SGD (learning rate 0.03, batch size 32). Given the large number of classes, we use buffer sizes of 2000 and 5000. Prompt optimization is performed in Stable Diffusion's text space via cross-entropy loss, guided by classifier predictions. We use the Adam optimizer [132] (learning rate: 0.1, batch size: 1), with the number of iterations controlled by an oracle network. The oracle is a single-hidden-layer MLP trained on a labeled dream quality dataset $\mathcal{D}_O$ (see Sec. 6.2.2) using Adam (learning rate set to 0.001, for 500 epochs). It stops optimization when a termination signal is predicted in at least $n = 2$ of the past $k = 3$ iterations. We use ImageNet-R as $\mathcal{D}_O$ when testing on Mini-ImageNet or ImageNet-100, and ImageNet-100 for ImageNet-R. Results are for class-incremental setting, reported as mean $\pm$ std over 5 runs.

6.3.3 Results

To assess the impact of our method, we evaluate its effectiveness when applied in conjunction to continual learning state-of-the-art models. Since the dream generation mechanism relies on combining learned prompts with past experiences stored in the buffer, we apply it exclusively in conjunction with rehearsal-based methods. Specifically, we consider ER [131], DER++ [96], and ER-ACE [103], comparing their performance with and without the dreaming generation. Tab. 6.1 presents results in terms of FAA in the class-incremental setting. Our approach leads to a significant improvement in performance across all benchmarks, demonstrating the importance of mimicking human dreaming for mitigating forgetting.

One of our key claims is that our dreaming mechanism enhances a model's ability to prepare for future tasks. To validate this, we report final forward transfer (FFWT) in Tab. 6.2. Results shows that dream generation improves FFWT, with D2L achieving positive forward transfer in some cases, similar to WSCL. However, WSCL achieves positive forward transfer by relying on additional real data to simulate dreams, whereas D2L internally generates these dreams by leveraging the model's own internal knowledge. Furthermore, we

	Mini-ImageNet		ImageNet-100		ImageNet-R	
	2000	**5000**	**2000**	**5000**	**2000**	**5000**
ER [103]	27.91±3.49	34.21±3.04	21.08±2.38	22.21±3.44	7.68±0.97	10.69±1.29
↪**D2L**	**31.18**±2.74	**39.75**±2.61	**23.53**±1.98	**32.73**±3.39	**8.67**±0.66	**11.84**±0.95
DER++ [96]	14.74±2.14	26.92±4.72	14.43±3.68	23.86±2.54	6.08±0.81	8.29±1.15
↪**D2L**	**21.06**±5.45	**31.91**±5.19	**18.86**±3.22	**25.38**±2.17	**8.60**±1.00	**10.89**±1.56
ER-ACE [131]	33.26±3.51	40.59±1.20	24.79±5.02	30.16±4.97	7.09±0.59	9.44±0.70
↪**D2L**	**40.90**±0.95	**47.32**±0.89	**31.57**±1.20	**38.50**±1.01	**9.54**±0.39	**12.51**±0.56

Table 6.1: **Final average accuracy of rehearsal-based methods with and without D2L in the Class-IL setting.** Results are reported for buffer sizes of 2000 and 5000. Bold values indicate the best performance in each column.

	Mini-ImageNet		ImageNet-100		ImageNet-R	
	2000	**5000**	**2000**	**5000**	**2000**	**5000**
ER [103]	-2.58	-1.62	-1.88	-1.52	-1.32	-0.64
↪**D2L**	**+0.33**	**+0.47**	**+1.79**	**+1.19**	**+0.24**	**+0.14**
DER++ [96]	-1.55	-2.00	-1.36	-2.48	-1.03	-1.83
↪**D2L**	**+0.97**	**+0.71**	**-0.13**	**+1.86**	**+0.30**	**+0.24**
ER-ACE [131]	-1.99	-2.45	-2.00	-2.16	-2.46	-1.27
↪**D2L**	**+1.05**	**-1.58**	**+1.09**	**+0.08**	**-0.25**	**+0.17**

Table 6.2: **Final forward transfer of rehearsal-based methods with and without D2L in the Class-IL setting.** Results are reported for buffer sizes of 2000 and 5000. Bold values indicate the best performance in each column.

conduct a comprehensive performance analysis by comparing the best performing version of our approach from Tab. 6.1 (i.e., ER-ACE + D2L) with state-of-the-art continual learning methods[3]: GSS [91], A-GEM [92], iCaRL [88], FDR [89], BiC [90], and RPC [98]. Results are shown in Table 6.3. To contextualize these results, we also define a lower bound as training without any countermeasure to forgetting (*Fine-tune*). ER-ACE + D2L outperforms state-of-the-art methods across all examined datasets and buffer sizes, with significant margins. Note that our method targets continual learning from scratch with a randomly initialized convolutional backbone, without external pre-training. By contrast, a separate line of work relies on pre-trained models, either via full/partial fine-tuning [182, 183, 184] or prompt tuning [185, 106] that adapts frozen ViT backbones. These approaches operate in a different

[3]Results were obtained using the original code released alongside the corresponding papers.

Method	Mini-ImageNet		ImageNet-100		ImageNet-R	
Fine-tune	6.72±1.20		6.98±0.10		4.46±0.15	
	Buffer-based methods					
	2000	**5000**	**2000**	**5000**	**2000**	**5000**
A-GEM [92]	6.78±1.13	7.45±0.76	6.20±1.11	6.11±1.13	4.69±0.03	6.29±0.84
BiC [90]	30.56±7.41	37.84±0.61	27.83±2.75	32.29±0.70	7.15±1.14	8.60±2.07
DER++ [96]	14.74±2.14	26.92±4.72	14.43±3.68	23.86±2.54	6.08±0.81	8.29±1.15
ER [103]	27.91±3.49	34.21±3.04	21.08±2.38	22.21±3.44	7.68±0.97	10.69±1.29
ER-ACE [131]	33.26±3.51	40.59±1.20	24.79±5.02	30.16±4.97	7.09±0.59	9.44±0.70
FDR [89]	15.46±1.10	11.58±0.96	9.17±2.40	12.91±0.95	5.71±0.18	5.77±0.10
GSS [91]	6.40±0.38	5.71±0.08	8.07±0.26	9.23±0.85	5.08±0.13	4.29±0.32
iCaRL [88]	16.46±0.51	16.50±0.33	8.54±0.88	8.86±0.25	1.97±0.28	1.91±0.29
RPC [98]	9.22±0.30	9.02±0.24	8.13±0.11	7.41±0.74	5.71±0.03	6.32±0.80
ER-ACE + D2L	**40.90**±0.95	**47.32**±0.89	**31.57**±1.20	**38.50**±1.01	**9.54**±0.39	**12.51**±0.56

Table 6.3: **Final average accuracy of state-of-the-art continual learning methods in the Class-IL setting.** Results are reported for buffer sizes of 2000 and 5000, where applicable. Bold values indicate the best performance in each column.

evaluation regime, adapting an already rich representation and thus conflating the effect of the CL strategy with benefits from large-scale pre-training and transformer inductive biases. In our setting, the classifier's emergent knowledge directly guides the dreaming generation process, so mixing regimes would not yield an informative head-to-head comparison. For clarity and fairness, we therefore restrict comparison to methods that, like ours, train a CNN backbone from scratch under the same protocol constraints.

6.3.4 Ablation study

The ablation study is primarily conducted to assess the contribution of the dream generation strategy. The ER-ACE model [131], identified as the top-performing method when combined with our approach (see Tab. 6.1), is used as the baseline model for this study. All experiments are performed on the Split Mini-ImageNet dataset [59].

Impact of the oracle. We ablate the oracle and consider *Fixed optimization*, where prompt updates stop once the classifier predicts the target class for four consecutive steps. This criterion assumes the trajectory has reached and stabilized in the target representation space, but it limits adaptability. As shown in Tab. 6.4, our full method avoids this issue and

Method	Buffer size	
	2000	**5000**
ER-ACE baseline	33.26±3.51	40.59±1.20
+ Fixed optimization	38.94±0.97	46.41±0.55
+ **Oracle (D2L)**	40.90±0.95	47.32±0.89

Table 6.4: **Ablation on the oracle.** Results on Mini-ImageNet comparing our method with Fixed optimization.

Dreaming	Buffer size	
	2000	**5000**
No dreams	33.36±3.51	40.59±1.20
At beginning	36.93±2.09	41.59±2.54
Incremental	36.85±1.16	43.87±2.90
D2L	**40.90**±0.95	**47.32**±0.89

Table 6.5: **Impact of dream class updates.** Comparison on Mini-ImageNet of different dreaming strategies.

achieves superior performance.

Impact of dream class updates. We assess how updating dreams during the sequential learning of tasks affects the overall performance. Our default strategy creates new dream classes at the end of each task—equal in number to that task's classes—and replaces an equal number of old ones, keeping the classifier output size fixed. We compare this with three variants: (1) *No dreams*, a baseline without generated classes; (2) *At beginning only*, where dream classes are generated once during the first task and reused throughout; and (3) *Incremental*, which accumulates dream classes across tasks, expanding the output layer with weights sampled from a Gaussian distribution estimated from existing neurons. Results in Tab. 6.5 demonstrate that our replacement strategy yields the best results. Fixed dreams limit forward transfer by excluding later tasks, while incremental expansion likely degrades performance due to increased task complexity.

Dreaming generation mechanism. To further validate our method, we also replaced the dreaming mechanism with alternative strategies for creating class blends. In principle, the dreaming process could be replaced by surrogate samples generated from past knowledge using interpolation-based techniques. To test this hypothesis, we substituted our dreamed

Method	Buffer size	
	2000	**5000**
ER-ACE baseline	33.26±3.51	40.59±1.20
+ Mixup	36.84±0.94	44.82±1.27
+ Continual Mixup	36.05±1.22	43.45±0.83
+ **Oracle (D2L)**	**40.90**±0.95	**47.32**±0.89

Table 6.6: **Ablation on dream generation strategies.** Evaluation on Mini-ImageNet comparing interpolation-based baselines with our proposed D2L.

classes with synthetic ones obtained through different strategies: (1) *Mixup* [186]: combines images from the current data stream with samples from the replay buffer to form auxiliary synthetic classes; (2) *Continual Mixup*: applies the same interpolation scheme, but only between images sampled from the replay buffer. The corresponding results are reported in Table 6.6. In contrast to Mixup variants, which rely on static blending of existing representations, D2L produces distinct and task-aware latent clusters. Importantly, our generation process is explicitly guided by the knowledge encoded in the classifier.

6.3.5 Comparison with Generative Replay

Finally, we clarify the conceptual differences between *generative replay* (GR) methods and D2L, and we provide additional experimental results for completeness.

GR methods such as DGR [81] or DDGR [85] discard the memory buffer and rely entirely on a generative model to reconstruct past data, with the objective of preserving knowledge of previous tasks through explicit replay. In contrast, D2L differs along two fundamental dimensions:

- Unlike GR, D2L retains a fixed-size buffer (e.g., 2000 samples), ensuring direct access to real exemplars throughout training. Nonetheless, buffer never contains generated images.
- While GR employs generation to reproduce past samples that directly replace the buffer, D2L leverages generation in a profoundly different way: generated images in D2L do not serve as memory replacements, but as additional data stream (additional classes) that pre-activate future-class representations.

This conceptual divergence means that direct comparisons should be interpreted carefully, as the underlying objectives and mechanisms are not the same. Nevertheless, since

Method	Mini-ImageNet	ImageNet-100	ImageNet-R
DGR [81]	23.33 ± 0.32	26.17 ± 0.21	7.00 ± 0.26
DDGR [85]	37.48 ± 0.98	30.81 ± 0.54	9.21 ± 0.17
ER-ACE + D2L (ours)	**40.90** ± 0.95	**31.57** ± 1.20	**9.54** ± 0.39

Table 6.7: **Comparison with generative replay methods under the same buffer constraint (2000 samples).** While D2L is not a generative replay method, its use of generation for anticipatory transfer leads to superior performance compared to GR approaches.

both approaches involve generative components during training and thus incur comparable overheads, we report results against representative GR methods under the same buffer constraint, as shown in Tab. 6.7.

These results confirm that D2L achieves higher accuracy than GR approaches, despite pursuing a different goal. While GR attempts to reconstruct and replay the past, D2L leverages anticipatory generation to expand and stabilize the representation space, proving more effective across all benchmarks.

6.4 Discussion

In this chapter, we introduced Dream2Learn, an approach inspired by the ability of the human brain to consolidate past experiences and anticipate future ones through dreaming. Our method pairs a classification network with a generative model to synthesize structured training signals, reinforcing past knowledge and enhancing forward transfer. Experiments on standard continual learning benchmarks show that dreaming helps mitigate forgetting and can support feature learning by expanding the classifier's representation space, turning negative forward transfer into positive. Using soft prompt optimization within a latent diffusion model, D2L generates novel yet coherent classes with the oracle model helping to maintain sample quality by preventing collapse.

While D2L-generated dreams are qualitatively distant from the target dataset, a limitation is the potential categorical leakage of future-class information, due to the possibility that such knowledge is embedded in the generator. In a preliminary study that investigates this risk, we found that these events are not only rare but also contingent on the premise that Stable Diffusion can faithfully synthesize future classes. Nonetheless, mitigation of this theoretical leakage could involve unlearning future-class concepts from the diffusion model, e.g., through concept-erasure or negative-gradient editing—prior to prompt optimization. An additional limitation of the proposed approach, compared to simpler rehearsal methods,

is the computational overhead that D2L incurs during training due to the optimization of dream prompts.

6.5 Publications

Bellitto, G., Calcagno, S., Pennisi, M., Salanitri, F. P., Sorrenti, A., Palazzo, S., & Spampinato, C. "Dream2Learn: Structured Generative Dreaming for Continual Learning". Submitted to *International Journal of Computer Vision*.

Chapter 7

Continual Object 6D Pose Estimation

6D pose estimation, i.e., the prediction of the 3D position and 3D orientation of a target object from images, is a fundamental problem in computer vision, with wide applications ranging from autonomous driving to virtual/augmented reality, robotic grasping, and robot-assisted surgery [187, 188, 46].

Supervised methods for 6D pose estimation typically rely on establishing keypoint correspondences between CAD models and input RGB images [189], RGB-D images [190, 191], or directly regress pose using input RGB-D images [192, 193]. Depending on the underlying assumptions, 6D pose estimation can be performed at either the instance level [189, 190] or the category level [191, 193]. However, the majority of these approaches are trained on large datasets in an offline setting, assuming that the training and test sets are independently and identically distributed (i.i.d.) and that all target objects are available for training at the same time. This assumption is mostly impractical in real-world scenarios where a deployed model must incrementally adapt to new data. In such contexts, existing methods struggle to accurately estimate the 6D pose for unseen objects, whose feature distribution differs from those encountered in the training set. Moreover, the conventional practice of retraining the model with the entire dataset, whenever a new object is introduced, is unfeasible and not efficient: retraining the model demands significant computational resources and time, and storing and processing large datasets may overwhelm robots with limited memory capacities.

To address the limitation of offline retraining, one-shot [194, 195] and few-shot [196] methods leverage annotated support views of new objects, establishing correspondences between these views and the query view for pose estimation. These methods relax the constraint for high-fidelity object models, but necessitate training on specific instances or categories. Test-time adaptation methods [197, 198] attempt to bridge the gap between the training and

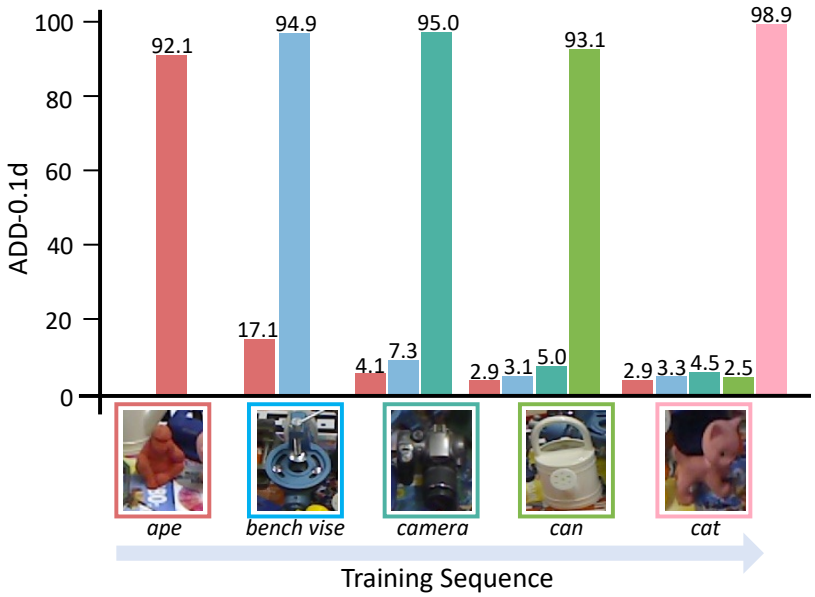

Figure 7.1: **Performance degradation during sequential training on five objects of the Linemod dataset.** Each color bar corresponds to a specific object. For example, pose estimation accuracy for *ape* (in red) drops from 92.1% at task 1 to 2.9% at task 5.

test sets by adapting a pre-trained model to new objects encountered during testing, either through supervised or unsupervised training.

However, adapting the model only using data from new objects leads to catastrophic forgetting, as illustrated in Figure 7.1. Existing applications of continual learning methodologies predominantly deal with image classification, with relatively little attention devoted to other domains such as robotic perception [199], reinforcement learning [200, 201], and natural language processing [202]. To our knowledge, the exploration of continual learning for 6D pose estimation has not been investigated. In this chapter, we introduce a novel framework for learning the 6D pose of objects, specifically designed for scenarios where a model can adapt incrementally to new objects without losing knowledge of previously encountered ones. Our approach trains a model to estimate the 6D pose on a subset of objects (a *task*) at a time, selectively storing keyframes of objects from the current task in a memory buffer. At each new task, the model keeps learning by combining data from new objects and the memory buffer, which is simultaneously updated to accommodate keyframes of the new task objects. Throughout this continual learning process, we introduce a parameter regularization method aimed at adaptively adjusting the model parameters to retain those essential for accurately estimating the pose of objects stored in the memory buffer. Furthermore, we propose a strategy to alleviate overfitting on the objects contained within the memory buffer.

Our main contributions are the following:

- We present a novel setting for 6D object pose estimation, wherein the model progressively learns to estimate the 6D pose of new objects.
- We introduce a replay-based continual learning method, called ILPose, that prioritizes adaptability to new objects while ensuring retention of knowledge about previously seen ones.
- We validate our method by comparing with existing continual learning baselines on the Linemod [203] and YCB-Video [204] datasets.

7.1 Related work

Methods for 6D pose estimation can be classified as either *instance-level* or *category-level*, depending on their generalization capabilities. *Instance-level* methods [192, 189, 205] assume that target object CAD models are available, and focus on estimating the 6D pose of specific objects. *Category-level* methods [206, 207, 193] assume that category information is available, and build models to learn the category-specific representation of object appearance and shapes, enabling 6D pose estimation within the same category. However, since instance-level methods require high-fidelity CAD models for each object of interest and category-level methods require prior knowledge of the target category, neither can adequately handle new objects that are not shown in the training set.

To address the challenge of achieving 6D pose estimation for new objects, one-shot [194, 195] and few-shot [196] 6D pose estimation methods have been proposed. These methods annotate one or several views of the new object as support views and then match keypoints between the support views and the query scene for 6D pose estimation [208, 209]. These methods avoid using CAD models but require training a separate model for each object. Test-time adaptation is another approach to estimate the 6D pose of new objects [210], which is used to enhance the performance of the model when there is a distribution shift between known and new objects. These approaches [198, 211] pre-train a model on known objects in a supervised manner, and then adapt the pre-trained model to new objects in a supervised or unsupervised manner. However, the model would forget previous knowledge after adaptation.

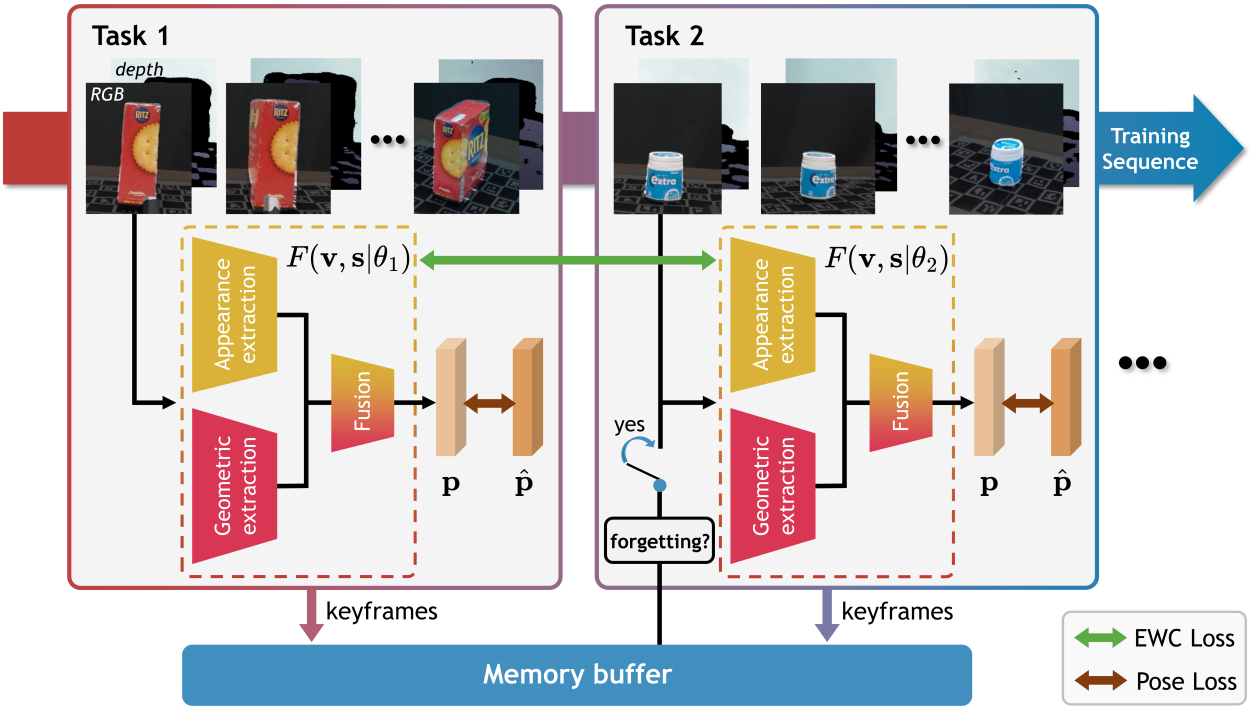

Figure 7.2: **Continual object 6D pose estimation framework.** Our model takes an RGB-D frame and its target object mask as input for 6D pose estimation. The trained model is sequentially adapted to new objects by using data from both the new object and the memory buffer, where keyframes of previously seen objects are stored.

7.2 Method

We define the proposed continual object 6D pose estimation as a task-incremental learning problem, where a network F undergoes training on a sequence of T tasks $\{\tau_1, \ldots, \tau_T\}$ to estimate the 6D pose of multiple objects. Each task involves learning the 6D pose of a specific subset of objects from a set $O = \{o_1, \ldots, o_n\}$ with annotated poses. Formally, τ_i represents the subset of objects for the i-th task, such that $\tau_i \subset O$, $\tau_i \cap \tau_j = \emptyset$, and $\bigcup \tau_i = O$.

We design a network F that takes as input a pair $\mathbf{d} = (\mathbf{v}, \mathbf{s})$, where $\mathbf{v}$ is the RGB-D image and $\mathbf{s}$ is the segmentation mask for the target object. The network F directly regresses the object 6D pose between the camera and object coordinate system, represented as $\mathbf{p} = F(\mathbf{v}, \mathbf{s}|\boldsymbol{\theta})$, with $\boldsymbol{\theta}$ being the trainable parameters. We represent 6D pose as a homogeneous transformation matrix $\mathbf{p}$ that consists of a 3D rotation $\mathbf{R} \in SO(3)$ and a 3D translation $\mathbf{t} \in \mathbb{R}^3$.

Figure 7.2 shows the overview of our method. During each task τ_i, we employ a selection process to identify k keyframes, maximizing multi-view diversity. These selected keyframes, along with their corresponding ground truth poses, are stored in a memory buffer $\mathcal{M}$. At

each task, the model F is trained to estimate the pose of the task's objects; additionally, starting from task τ_2, we include data from the memory buffer $\mathcal{M}$ in the training process.

7.2.1 Object pose estimation

We employ a model that directly estimates the 6D pose information from input images, leveraging both color and geometric features to address pose estimation challenges for objects with low texture or similar colors. Given an RGB-D image as input, we crop the RGB image and its corresponding depth image using the segmentation mask of the target object. This ensures that the model focuses solely on the region containing the target object.

For RGB feature extraction, we utilize a ResNet-18 [111] as an encoder to extract appearance features from the cropped image. Simultaneously, we extract geometric features by lifting the target object's point cloud from the cropped depth images using camera intrinsic parameters. We use PointNet [212] to process the point cloud and obtain geometric features for each 3D point. We randomly select 500 points with the corresponding pixels in RGB image, fuse their extracted appearance and geometric features, and obtain a pixel-wise feature representation using DenseFusion [192]. Finally, we regress the 6D pose of the target object by leveraging the pixel-wise feature representation.

The estimation model is trained by minimizing a pose loss $\mathcal{L}_{pose}$ that encourages the estimation of accurate pose information. We use the Average Distance of Model (ADD) scores [203] as $\mathcal{L}_{pose}$. It measures the distance between points transferred using the predicted 6D pose and points transferred through the ground truth pose, expressed as:

$$\mathcal{L}_{pose} = \frac{1}{T} \sum_{i} \left\| (\mathbf{R}\mathbf{x}_i + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_i + \hat{\mathbf{t}}) \right\|^2, \tag{7.1}$$

where $\mathbf{x}_i$ represents the i-th 3D point from the object CAD model (with T overall number of points), $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ denote the rotation and translation annotations and $\mathbf{R}$ and $\mathbf{t}$ the estimated rotation and translation values.

7.2.2 Memory-based continual learning

We design a replay-based strategy for continual object 6D pose estimation. This strategy consists of two key components: *keyframe selection* and *adaptive parameter regularization*.

Keyframe selection. The selection of samples from past tasks is a crucial step, as these samples directly influence the model's robustness on previous tasks following adaptation to new ones. The memory buffer contains frames that must effectively provide exhaustive visual

Algorithm 2 Incremental object 6D pose estimation for each task

Input: data from both the task data distribution $\mathcal{D}_n$ and the memory buffer $\mathcal{M}$
for all *epochs* **do**
  for $\mathbf{v}_i, \mathbf{s}_i \in \mathcal{D}_n$ **do** ▷ training of the model on $\mathcal{D}_n$
    $\mathbf{p}_i \leftarrow F(\boldsymbol{\theta}|\mathbf{v}_i, \mathbf{s}_i)$
    $Loss \leftarrow Loss + \mathcal{L}_{pose}(\mathbf{p}_i) + \lambda \mathcal{L}_{EWC}(\mathbf{p}_i)$
  end for
  for $\mathbf{v}_j, \mathbf{s}_j \in \mathcal{M}$ **do** ▷ evaluation of the model on $\mathcal{M}$
    $\mathbf{p}_j \leftarrow F(\boldsymbol{\theta}|\mathbf{v}_j, \mathbf{s}_j)$
    $Loss_{eval} \leftarrow Loss_{eval} + \mathcal{L}_{pose}(\mathbf{p}_j)$
  end for
  if $(Loss_{eval} \,/\, |\mathcal{M}|) > \delta$ **then**
    for $\mathbf{v}_j, \mathbf{s}_j \in \mathcal{M}$ **do** ▷ training of the model on $\mathcal{M}$
      $\mathbf{p}_j \leftarrow F(\boldsymbol{\theta}|\mathbf{v}_j, \mathbf{s}_j)$
      $Loss \leftarrow Loss + \mathcal{L}_{pose}(\mathbf{p}_j) + \lambda \mathcal{L}_{EWC}(\mathbf{p}_j)$
    end for
  end if
  update Fisher information matrix $\mathcal{G}$ ▷ using data from $\mathcal{M}$
  update $\boldsymbol{\theta}$ based on $\frac{\partial Loss}{\partial \boldsymbol{\theta}}$
end for
update $\mathcal{M}$ with data from the current task data distribution $\mathcal{D}_n$

appearance information about seen objects. To achieve this, we propose a selection strategy based on the Scale-Invariant Feature Transform (SIFT) algorithm [213].

SIFT is primarily employed to detect and describe local features in images by identifying distinctive matching keypoints in two images and estimating the transformation between them. We utilize this transformation as a measure of the view changes between sequential frames. Hence, we leverage SIFT to quantify the changes in view across consecutive frames and select *keyframes* that maximize multi-view diversity while minimizing redundancy.

Formally, given a sequence of frames $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ depicting an arbitrary object $o \in O$, we select the initial frame $\mathbf{v}_1$. Subsequently, we compare the other $n-1$ frames with $\mathbf{v}_1$ using SIFT to find matched keypoints for transformation calculation, yielding a set of scores $\mathbf{S} = \{s_{1,j}, \text{ with } j = 2 \ldots n\}$ with $s_{1,j}$ measuring the Euclidean norm of translation and rotation angle, indicating the extent of view changes. The k frames with the highest scores are identified as *keyframes* and added to the memory buffer $\mathcal{M}$. The number of keyframes in the buffer is equal for all objects.

Adaptive parameter regularization. While adapting the current model to a new task, knowledge of past tasks may be overwritten or altered, leading to catastrophic forgetting. We address the forgetting problem in our continual 6D pose estimation by adaptively regularizing

the parameters using a variant of Elastic Weight Consolidation (EWC) [72]. In particular, while the common online EWC implementation computes the Fisher information matrix $\mathcal{G}$ at the end of each task, on the task's data only, and updates it through exponential moving average, we compute it using all samples from the buffer $\mathcal{M}$, through the second-order derivative:

$$\mathcal{G} = \mathbb{E}[-\frac{\partial^2}{\partial\theta_i^2}\ell(\mathbf{p}_m|\boldsymbol{\theta}_i)], \tag{7.2}$$

where $\mathbf{p}_m$ is the ground truth for the m-th sample in the buffer and $\ell(\mathbf{p}_m|\boldsymbol{\theta}_i)$ is the corresponding log-likelihood [214]. This choice is motivated by the replay-based nature of our framework, as the buffer $\mathcal{M}$ stores selected keyframes of previously seen objects and thus provides meaningful information for estimating parameter importance. As the Fisher information matrix is related to the importance of each parameter, we add a regularization loss to penalize the change of each parameter according to its importance:

$$\mathcal{L}_{EWC} = \sum_j \mathcal{G}(\boldsymbol{\theta}_{i-1}^{(j)} - \boldsymbol{\theta}_i^{(j)})^2, \tag{7.3}$$

where $\boldsymbol{\theta}_{i-1}^{(j)}$ represents the j-th parameter of the weights at task τ_{i-1} and $\boldsymbol{\theta}_i^{(j)}$ is the same parameter of the model at task τ_i.

7.2.3 Optimization

Our learning strategy foresees that the pose estimation model is trained on the first task in a supervised manner, using only Eq. 7.1. For subsequent tasks, the model is trained by minimizing the overall loss $\mathcal{L}_{adp}$:

$$\mathcal{L}_{adp} = \mathcal{L}_{pose} + \lambda\mathcal{L}_{EWC}, \tag{7.4}$$

where $\mathcal{L}_{pose}$ is computed for all samples of the new task, while $\mathcal{L}_{EWC}$ only on the samples present in the replay buffer $\mathcal{M}$. However, a common challenge in replay-based continual learning methods is the tendency to overfit the buffer data [215]. This occurs as the model repeatedly learns from the buffer, potentially hindering generalization to previous tasks and causing loss of previously learned knowledge. Additionally, using a buffer directly without additional training strategies, often results in a polarization towards the new object data, whose effect is controlled by the hyperparameter λ.

To mitigate this issue, we propose a novel adaptation strategy, adding a loss-gating mechanism. We exclusively use data from the new object to enable rapid adjustment of the network's trainable parameters to suit the characteristics of the new object. During

training, after each epoch, we evaluate network performance on memory buffer data to detect any significant drop in performance on previously seen objects. If the loss value (i.e., the ADD value obtained by Eq. 7.1) exceeds a predefined threshold, δ, indicating a decline in performance, we incorporate memory buffer data for training using Eq. 7.4. Conversely, if the loss value remains below the threshold, signifying retention of previously encountered objects, we refrain from adding memory buffer data to training to prevent overfitting. The pseudocode of the procedure performed for each task is shown in Algorithm 2.

7.3 Experimental results

7.3.1 Datasets and metrics

We evaluate our approach on two widely adopted 6D pose estimation benchmarks and on a proprietary dataset collected with a robotic arm to assess real-world applicability:

- **Linemod** [203]: The Linemod dataset contains 13 low-textured objects in 13 videos, with different backgrounds. Linemod provides annotated 6D pose information and semantic segmentation information. It is widely used by recent deep learning-based methods [189, 190, 216]. We mainly use this dataset to train our model incrementally, one object at a time.

- **YCB-Video** [217]: The YCB-Video dataset contains 21 YCB objects of varying shape and texture. The dataset contains 92 RGB-D videos, where each video shows a subset of the 21 objects in different indoor scenes. The videos are annotated with 6D poses and segmentation masks.

- **Ours**: We use a robotic arm to collect the dataset including four objects, *cracker box*, *gum*, *stapler*, *tea box*, and two low-texture objects, *red cup*, *paper cup*. A camera is fixed on the tip of the robotic arm and captures RGB-D images with a resolution of 1280×720 pixels. We use this dataset to validate our approach in a real-world setting by performing 6D object pose-based robotic grasping.

To quantify pose accuracy, we consider three metrics widely used in 6D pose estimation problems:

- **ADD-0.1d** ($\uparrow$): percentage of correct poses. The estimated pose is considered to be correct if the ADD distance is less than 10% of the object diameter.

- $\boldsymbol{R_{err}}$ ($\downarrow$): mean rotation error in degrees, which measures the average angle between the predicted rotation and the ground truth rotation.

- $\boldsymbol{T_{err}}$ (↓): mean translation error in centimeters, which is used to measure the average Euclidean distance between the predicted translation and the ground truth.

We assess the effectiveness of our method by evaluating its overall performance on both past and present tasks, encompassing all objects encountered so far. To accomplish this, we report the performance of the model in terms of *Final Average* metrics [103]: ADD-0.1d^{FA}(↑), R_{err}^{FA}(↓) and T_{err}^{FA}(↓). These metrics represent the average performance measures over all objects encountered after the last task of the sequence. Let ψ_i^j denote the value of an arbitrary metric at the end of task j computed on the test set of task τ_i (with $i \leq j$), the *Final Average* is defined as:

$$\Psi^{FA} = \frac{1}{T} \sum_{i=1}^{T} \psi_i^T, \tag{7.5}$$

with T representing the total number of tasks.

7.3.2 Baselines

We compare our method, *ILPose*, with a selection of other approaches broadly inspired by existing state-of-the-art continual learning methods, introduced in Sec. 2.5. Most of these methods are originally designed for the classification task, so some changes have been made to adapt them to the 6D pose estimation task.

- **Multi-Encoder**: This strategy draws inspiration from the Progressive Neural Networks (PNN) architectural approach [76], in which a distinct replica of the entire backbone is dedicated to each task. While this strategy inherently prevents forgetting, its primary drawback lies in the linear increase of memory requirements with the number of tasks. To balance efficiency and memory footprint, our implementation allocates a separate encoder module for each task, while the rest of the network remains shared across all tasks. When a new task begins, a new encoder replica is instantiated, initialized with the same weights as its predecessor, and adopted during the subsequent training session, while all the other encoders are inactive. During inference, an object identifier is used to select the appropriate encoder.

- **Self-Distillation**: Inspired by [70], this method applies functional regularization via self-distillation between the in-training model and a previous snapshot stored in the buffer. More specifically, at the end of task τ_{i-1}, we store the pixel-wise features $\mathbf{h}_{i-1}$ for each buffer sample $m \in \mathcal{M}$. In the next task τ_i, we incorporate an additional loss $\mathcal{L}_{SD}$ to mitigate potential degradation of the learned representations up to task τ_{i-1}. At each training step of the current task τ_i, we sample an image v from the

training stream, and an image m randomly selected from the buffer, and we optimize the network by minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_{pose}^{(v)} + \mathcal{L}_{pose}^{(m)} + \alpha \mathcal{L}_{SD}(\mathbf{h}_{i-1}^{(m)}, \mathbf{h}_{i}^{(m)}), \tag{7.6}$$

where α is a weighting factor between the three loss terms, and $\mathcal{L}_{SD}$ is the Mean Squared Error Loss.

- **Hybrid**: This approach combines the strengths of the first two methods, leveraging the advantages of both techniques. It employs a separate encoder for each new object, coupled with the additional loss $\mathcal{L}_{SD}$ to preserve the knowledge associated with the previous state of the model.

- **vanilla-EWC** [72]: In this buffer-free method, the model continually adapts to new objects by updating trainable parameters under the penalty of the EWC term applied to important parameter changes. In this case, the loss-gating mechanism is not applied.

- **Joint** & **Fine-tune**: To provide a more exhaustive understanding of our findings, we also include the scenario where a model is trained jointly on all objects together (referred to as *Joint*) in a conventional, non-incremental fashion. In addition, we present the results by training the model sequentially on each task without implementing any measures to mitigate forgetting (referred to as *Fine-tune*). These two results can be viewed as upper and lower bounds, respectively.

7.3.3 Training procedure

All models are trained according to a standard continual learning protocol [146]. When training on a given task, only images corresponding to that task are used, with the exception of k samples for each previous object stored in the buffer (if the method permits). To ensure a fair comparison between different methods, all the networks are trained using the Adam [132] optimizer for 300 epochs per task. We select $\lambda = 2$ to balance the performance on a new task and retention of previous knowledge, while the threshold δ is determined based on the object diameter d, defined as $\delta = 0.1d$.

For each method, the model is trained on a sequence of 5 objects from either the Linemod dataset [203] (*ape*, *bench vise*, *camera*, *can* and *cat*) or the YCB-Video dataset [217] (*master chef can*, *cracker box*, *sugar box*, *tomato soup can* and *mustard bottle*); for both datasets, each task is associated to a single object, i.e., $\tau_i = o_i$. The evaluation of the methods equipped with a buffer was performed with two different buffer size settings, i.e. storing $k = 30$ and

Figure 7.3: **Visualization of 6D pose.** The results on (top) Linemod and (bottom) YCB-Video. The green 3D bounding boxes are obtained based on the ground truth 6D pose, while the red ones are generated from the model prediction after the incremental adaptation to the final object.

$k = 50$ images per object. Results are presented as the mean and standard deviation over three different runs.

7.3.4 Results

Table 7.1 and Table 7.2 show that ILPose outperforms all baselines on the Linemod and YCB-Video datasets, providing a better trade-off between forgetting and final performance. In the tables, we also compare the impact of the number of keyframes on final results. Qualitative results are shown in Figure 7.3.

Although vanilla-EWC does not leverage frames from previous tasks, it yields results comparable to buffer-based methods, particularly in the case of $k = 30$. This serves as a direct baseline for our method, suggesting that regularization based on the Fisher information matrix effectively addresses our task.

Among buffer-based methods, the architectural approach utilizing multiple encoders performs poorly across all cases. Employing separate encoders for each task seems ineffective, as the encoder's internal representation remains closely tied to the inherent features of the object in question. Consequently, the shared part of the network struggles to exploit features extracted by different encoders. This phenomenon may also explain the results obtained with the Self-Distillation method, which, despite using a single encoder, better preserves the model's internal representation due to the additional loss. Clearly, when the network retains its weights to ensure that current pixel-wise features resemble those extracted previously, it suffers less from forgetting.

We evaluate the impact of catastrophic forgetting on the 6D pose estimation task by

Method	ADD-0.1d$^{FA}(\uparrow)$	$R_{err}^{FA}(\downarrow)$	$T_{err}^{FA}(\downarrow)$
Joint	90.3±0.7	7.7±1.1	0.7±0.2
Fine-tune	18.1±1.9	50.4±6.4	3.1±0.3
vanilla-EWC	36.6±2.5	27.7±1.7	1.7±0.1

	30 keyframes			**50 keyframes**		
	ADD-0.1d$^{FA}(\uparrow)$	$R_{err}^{FA}(\downarrow)$	$T_{err}^{FA}(\downarrow)$	ADD-0.1d$^{FA}(\uparrow)$	$R_{err}^{FA}(\downarrow)$	$T_{err}^{FA}(\downarrow)$
Multi-Encoder	29.6±3.1	30.3±2.0	9.2±1.2	39.9±4.8	23.8±2.9	8.4±1.0
Self-Distillation	39.3±3.5	31.9±3.3	6.4±0.9	44.6±4.4	24.3±2.1	8.7±0.9
Hybrid	34.4±4.7	29.1±2.8	8.7±1.3	50.9±6.7	21.7±3.3	7.7±1.1
ILPose	**58.4**±3.0	**14.4**±1.1	**1.6**±0.1	**67.5**±1.1	**10.8**±1.8	**1.3**±0.1

Table 7.1: **6D pose estimation results on Linemod.** We report final average metrics for different numbers of keyframes per object stored in the buffer. We compare ILPose against adapted incremental learning baselines. Bold values indicate the best performance in each column.

Method	ADD-0.1d$^{FA}(\uparrow)$	$R_{err}^{FA}(\downarrow)$	$T_{err}^{FA}(\downarrow)$
Joint	96.2±0.9	5.7±0.5	0.5±0.1
Fine-tune	27.8±2.1	52.4±2.7	3.9±0.5
vanilla-EWC	46.8±1.4	31.2±1.6	2.5±0.3

	30 keyframes			**50 keyframes**		
	ADD-0.1d$^{FA}(\uparrow)$	$R_{err}^{FA}(\downarrow)$	$T_{err}^{FA}(\downarrow)$	ADD-0.1d$^{FA}(\uparrow)$	$R_{err}^{FA}(\downarrow)$	$T_{err}^{FA}(\downarrow)$
Multi-Encoder	35.6±1.1	26.1±0.8	8.2±1.4	38.9±1.4	23.9±2.2	6.3±0.8
Self-Distillation	39.4±2.5	29.8±1.3	3.2±2.1	45.1±2.7	25.1±1.0	4.4±1.5
Hybrid	36.8±2.9	25.2±2.7	5.9±2.3	49.2±3.9	21.7±3.1	6.7±1.4
ILPose	**63.5**±2.8	**15.2**±0.9	**1.6**±0.2	**79.2**±1.8	**10.2**±0.7	**1.2**±0.1

Table 7.2: **6D pose estimation results on YCB-Video.** We report final average metrics for different numbers of keyframes per object stored in the buffer. We compare ILPose against adapted incremental learning baselines. Bold values indicate the best performance in each column.

comparing the performance metrics at the end of each task between two scenarios: one with Fine-tune (lacking countermeasures to reduce forgetting) and the other with our proposed solution. Figure 7.4 illustrates this comparison, demonstrating a stark difference in performance. While the former shows a significant drop, our approach notably mitigates this decline.

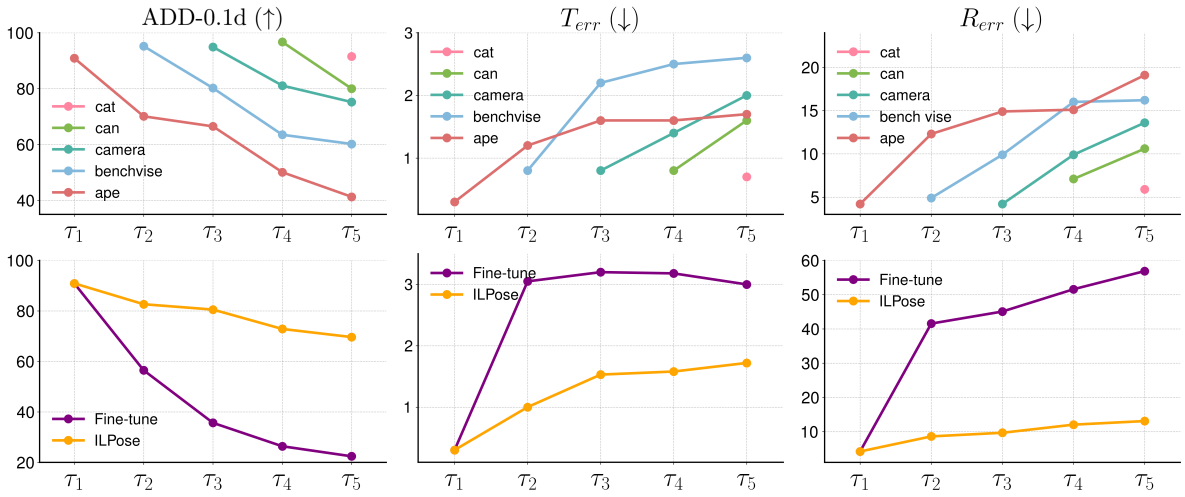

Figure 7.4: **Evaluation metrics for ILPose on Linemod.** First row: per-task ADD-0.1d(↑), R_{err}(↓) and T_{err}(↓) computed after training on each task. Second row: comparison between average metrics of our method and the Fine-tune model.

7.3.5 Ablation study

Impact of EWC term. We assess the effect of the EWC regularizer, by replacing it with a standard L_2 regularization term, keeping other settings the same as ILPose. Unlike EWC, L_2 regularization penalizes all trainable parameter changes. Consequently, the model using L_2 regularization tends to retain the first encountered object as fewer parameters can be updated to fit newly encountered objects. The results are displayed in Table 7.3, and reflect EWC's better capability in handling new objects by penalizing only significant changes.

Impact of keyframes selection. To assess the impact of keyframes selection on the results, we compared the proposed SIFT-based strategy for filling the memory buffer with *random selection*, i.e., keyframes are selected from the training set without taking into account any discriminative criteria. As detailed in Section 7.2.2, the SIFT-based selection aims at capturing diverse and informative keyframes. For each approach, we conduct experiments using the same task order and training configuration. Figure 7.5 shows a comparison of the results in terms of ADD-0.1d^{FA}(↑). These results demonstrate the limitations of random selection in losing crucial features related to different poses, highlighting the effectiveness of SIFT-based selection in preserving such information.

Method	30 keyframes			50 keyframes		
	ADD-0.1d^{FA}(↑)	R_{err}^{FA}(↓)	T_{err}^{FA}(↓)	ADD-0.1d^{FA}(↑)	R_{err}^{FA}(↓)	T_{err}^{FA}(↓)
L_2-based	42.7±1.4	24.7±2.1	1.5±0.1	52.4±2.1	21.1±1.4	1.3±0.2
ILPose	**58.4**±3.0	**14.4**±1.1	**1.6**±0.1	**67.5**±1.1	**10.8**±1.8	**1.3**±0.1

Table 7.3: **Effect of regularization.** We compare our original model, which employs EWC, with the L_2-based model, which uses L_2 regularization, while keeping all other settings unchanged.

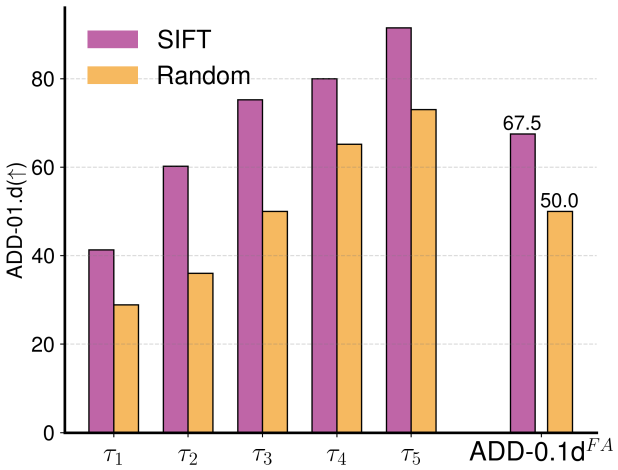

Figure 7.5: **Comparison of the impact of different keyframes selection strategies.** Per-task ADD-0.1d computed at the end of the training on the Linemod dataset. On the right, we report the ADD-0.1d^{FA}(↑).

7.3.6 Object grasping

In the absence of CAD models, $\mathcal{L}_{pose}$ is obtained by directly matching the predicted pose to the ground truth in terms of rotation and translation:

$$\mathcal{L}_{pose} = ||\mathbf{R} - \hat{\mathbf{R}}||^2 + ||\mathbf{t} - \hat{\mathbf{t}}||^2. \tag{7.7}$$

We leverage our collected dataset to train the model and assess robotic grasping from six different views. The model is optimized using the pose loss in Eq. 7.7 over the following sequence: *cracker box*, *gum*, *paper cup*, *red cup*, *tea box*, and *stapler*. Once training is completed, we estimate the 6D pose of each object and use the predicted poses to perform grasping.

Since our model predicts the object pose p, in the camera coordinate system, we first need to transform the object pose to the robot coordinate system before executing the

Method	*cracker box*	*gum*	*paper cup*	*red cup*	*tea box*	*stapler*
Joint	83.3%	100%	83.3%	50%	100%	66.7%
ILPose	83.3%	83.3%	66.7%	50%	100%	50%

Table 7.4: **Grasping success rate.** The robot attempts to grasp the object from 6 different views. A grasp is counted to be successful when the robot gripper holds for at least 5 seconds.

grasping. To achieve so, we need transformation $\mathcal{T}_{cg}$ from the camera to the gripper and the transformation $\mathcal{T}_{gr}$ from the gripper to the robot base. We then obtain the object pose in the robot coordinate system by:

$$\mathcal{T}_{or} = \mathcal{T}_{gr} * \mathcal{T}_{cg} * p, \tag{7.8}$$

where $\mathcal{T}_{cg}$ is estimated by the camera calibration process and $\mathcal{T}_{gr}$ is computed from the robot's forward kinematics.

In practice, our objective is to grasp the centroid of the object's 3D bounding box, as illustrated in Figure 7.3. The robot will move to each of the 6 different views and retrieve the 6D pose of the object estimated by our model. A successful grasp is defined as holding the object in the gripper for 5 seconds. The success rates are presented in Table 7.4. Our model's performance closely approaches that of the Joint (upper-bound) method, although the Joint method requires all available data for training, whereas ours is sequentially adapted to new objects. However, both the Joint method and our method yield unsatisfactory grasping results for *red cup*, which has a wider width and closely matches the maximum width of the gripper, resulting in less tolerance to translation errors. Additionally, the *red cup* and *paper cup* are textureless and symmetric objects, making it challenging for our model to obtain distinguishing features for 6D pose recovery. Furthermore, the lower accuracy observed on the *stapler* may be attributed to its limited height, which increases the likelihood of the gripper colliding with the table during grasping. For safety reasons, grasp execution is interrupted when the gripper gets too close to the table, which may lead to unsuccessful grasp attempts.

7.4 Discussion

In this chapter, we introduced a novel setting for 6D object pose estimation in which the model sequentially learns to recognize and estimate the pose of newly encountered objects without the impractical requirement of being retrained from scratch.

Within this formulation, we presented ILPose, a replay-based continual learning approach for 6D pose regression that maintains a memory buffer of representative keyframes from past

objects, selected to maximize viewpoint diversity. Beyond replay, our method integrates an adaptive regularization scheme inspired by EWC, which selectively constrains parameter updates based on their relevance to buffered objects. Since excessive reliance on buffer samples may bias optimization and hinder adaptation, we employ a loss-gated replay mechanism that is activated only when performance on the memory exhibits a clear degradation.

Experiments on Linemod and YCB-Video show that ILPose outperforms existing baselines for continual 6D pose estimation, improving the balance between final accuracy and forgetting. The robotic grasping experiments then suggest that these gains translate into more reliable performance in downstream manipulation. Nevertheless, failures on symmetric or textureless objects and in low-tolerance grasping scenarios indicate that errors that appear small under standard pose metrics may still be destructive for control. For these reasons, future work should focus on symmetry-aware modeling, uncertainty estimation, and tighter perception-manipulation integration.

7.5 Publications

Tian, L., Sorrenti, A., Pang, Y. L., Bellitto, G., Palazzo, S., Spampinato, C., & Oh, C. (2024, December). "Incremental Object 6D Pose Estimation". In *International Conference on Pattern Recognition* (pp. 331-346). Cham: Springer Nature Switzerland.

Chapter 8

Personalized Mental Well-being Monitoring

Psychological stress is increasingly recognized as a major health concern, particularly among university students, whose daily lives are characterized by intense academic demands, complex social interactions, and significant lifestyle changes [48, 49, 50]. Chronic exposure to high levels of stress has been linked to a broad spectrum of adverse health outcomes, including anxiety disorders, depression, impaired cognitive performance, sleep disturbances, and reduced physical health [218, 219, 220]. Consequently, there is a growing interest in the scientific community in developing automated, reliable methods to continuously monitor and predict stress levels in daily life.

Recent advances in wearable and smartphone technology have provided unprecedented opportunities for real-time and unobtrusive monitoring of individuals' psychological and behavioral states [221, 222]. These devices collect a wealth of multimodal data, including physiological signals (such as heart rate variability and galvanic skin response), behavioral patterns (such as physical activity and smartphone usage), and contextual information (such as location, ambient noise, and social interactions) [223, 224]. Leveraging this data, affective computing and mobile sensing research have made significant strides in creating predictive models of psychological stress that are applicable to everyday settings. Despite the progress achieved thus far, existing approaches primarily rely on static machine learning models trained offline using aggregated data from multiple individuals [221, 225, 226]. However, these methods struggle to generalize in real-world scenarios, as they are unable to adapt to temporal shifts in user behavior and require full retraining when exposed to new users or changing data streams.

In response to these challenges, this chapter introduces a novel paradigm by formulating the stress prediction problem within a continual learning framework. By framing each user

as a distinct incremental learning task, we create a model capable of dynamically adapting to new individuals while maintaining performance on previously learned users, thus inherently supporting personalized adaptation and sustained learning over extended periods.

Within this continual learning setting, we explore and compare two major approaches. The first is a memory replay-based approach, leveraging recurrent neural network architectures, such as standard RNNs [227] and bidirectional long short-term memory (BiLSTM) [228], combined with episodic memory buffers to mitigate catastrophic forgetting of previously learned tasks. The second is an extension of the former, where prompt-based mechanisms are integrated into the memory replay framework to further enhance knowledge retention across tasks. In this scenario, both the transformer backbone [229] and the task-specific prompts are continuously updated, with the prompts being refined using replayed data from previous tasks to support continual adaptation.

To rigorously evaluate the proposed approaches, we utilize a carefully curated subset of the StudentLife dataset [223]. This dataset comprises comprehensive smartphone-sensed behavioral data and detailed self-reported stress annotations from a population of university students collected over an extended observation period. The dataset provides rich multimodal signals, enabling a robust evaluation of stress prediction in realistic conditions.

Our experimental results demonstrate that the prompt-based approach achieves comparable or superior performance compared to standard memory replay methods, highlighting the benefit of incorporating task-aware adaptation.

Our main contributions are as follows:

- We propose a novel continual learning formulation of the stress prediction problem, tailored for dynamic real-world scenarios.
- We compare two continual learning strategies: memory replay using RNN-based models and prompt-based adaptation using transformer models.
- We empirically validate the effectiveness of behavioral and contextual features extracted from smartphone data for psychological stress prediction in both task- and domain-incremental learning settings.

8.1 Related work

The automatic estimation of psychological stress has been widely investigated in the field of affective computing, leveraging signals from mobile and wearable devices to infer mental states in everyday life [230, 222].

Designed to study stress in daily life, the StudentLife dataset [223] consists of passive sensing and self-report data collected from 48 university students over a 10-week period. The data revealed that increased academic demands were associated with reduced physical activity, shorter sleep, and lower positive affect. Motivated by these findings, numerous studies have proposed predictive models using both traditional and deep learning approaches. Classical methods such as logistic regression [221, 223] and decision trees [225] remain popular due to their interpretability and robustness in data-scarce conditions. Muaremi et al. [221] classified workplace stress using smartphone-derived features such as communication logs and heart rate variability (HRV), while Wang et al. [223] demonstrated associations between smartphone behavior, self-reported stress, and academic performance.

In parallel, machine learning models have sought to identify the most informative behavioral signals for stress prediction. DaSilva et al. [226] used penalized regression identifying social activity, location variance, and ambient noise as significant stress correlates. Bonafonte et al. [231] systematically compared feature subsets (e.g., WiFi, motion, phone usage), and found that mobility and device-logging patterns (e.g., night-time movement, battery events) were more predictive than communication metrics such as call or SMS frequency. These findings suggest that physical activity and rest patterns may serve as more reliable indicators of stress than communication behavior alone.

To support these modeling efforts, a variety of datasets have been developed across both controlled and real-world environments. In laboratory settings, the WESAD dataset [232] includes physiological and motion signals collected from 15 participants under three affective conditions: neutral, stress, and amusement. In contrast, real-world studies such as those by Muaremi et al. [221] and K-EmoPhone [224] provide multimodal smartphone and wearable recordings over multiple days, annotated with self-reported stress and affective states. Larger-scale longitudinal studies such as TILES [233] and Tesserae [234] further extend this direction by integrating wearable, mobile, and self-report data in workplace environments, enabling the study of stress over extended periods.

Despite these advances, most existing models rely on static training-test splits and assume a fixed population, limiting their ability to adapt to new users or evolving behavioral patterns. As Lazarou et al. [235] emphasize, the widespread adoption of wearable technologies enables continuous stress monitoring, yet current models often lack mechanisms for incremental adaptation and long-term personalization.

A central challenge in this context is the heterogeneity of behavioral and physiological signals across users, which has led to increasing interest in personalized or user-aware models. Shaw et al. [236] and Luo et al. [237] introduced deep multitask architectures with shared and user-specific components, allowing better adaptation to individual variability. In particular,

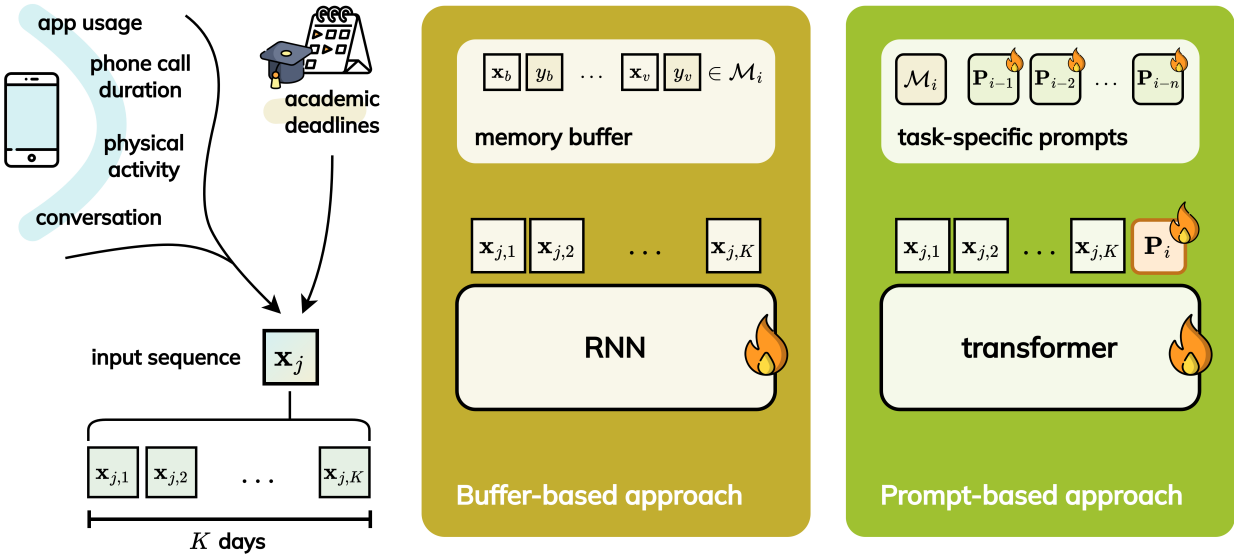

Figure 8.1: **Overview of the proposed method.** On the left, the input sequence $\mathbf{x}_j$, collected over the K days preceding a given stress report y_i, consists of behavioral and contextual features such as app usage, phone call duration, physical activity, conversation, and academic deadlines. In the middle, the replay-based approach combines a sequence model, such as an RNN, with a memory buffer of past task examples. On the right, the prompt-based method leverages a transformer encoder and a memory buffer, using learnable task-specific prompts P_i prepended to the input sequence to enable task adaptation.

their designs enable dynamic grouping of users based on latent similarity, which facilitates parameter sharing and mitigates the cold-start problem. Although not explicitly framed as continual learning, these approaches align with task-based continual learning paradigms, in which each user or user cluster can be considered a distinct task introduced sequentially.

This perspective resonates with the principles of personalized continual learning, in which models are updated incrementally as data from new users becomes available, without retraining from scratch.

Continual learning in health and affective computing. Acquire new information over time while retaining previously learned knowledge is especially critical in health-related and affective computing domains characterized by longitudinal data, inter-subject variability, and the need for sustained personalization [238].

In the medical domain, continual learning has been applied to diagnostic prediction, physiological signal interpretation, and medical image segmentation. Gonzalez et al. [239] proposed Lifelong nnU-Net, a continual learning variant of the widely-used nnU-Net, enabling models to adapt to novel anatomical structures and imaging protocols without catastrophic forgetting. Kiyasseh et al. [240] introduced CLOPS, a framework for biosignal analysis that

leverages experience replay and task-specific memory buffers to retain key temporal patterns from streaming physiological data such as ECG and PPG. These methods often integrate regularization strategies (e.g., EWC [72], LwF [70]) or modular architectures to support scalable and stable learning over time [215].

In affective computing, the need for personalization is especially acute due to high inter-subject variability and the dynamic nature of emotional states. Ahmad et al. [241] recently introduced SSOCL, a bi-level self-supervised framework for EEG-based emotion recognition that uses a dynamic buffer and pseudo-labeling to handle continuously streaming and unlabeled physiological data. In parallel, few-shot learning techniques have been effectively employed to address cross-subject variability in EEG emotion recognition, where models pre-trained offline can be rapidly fine-tuned to new users with minimal data, as demonstrated in several studies of prototype or metric-learning architectures [242].

Stress prediction is herein framed as a continual learning problem, where each user is treated as an independent task, introducing temporal continuity and lifelong adaptation and aligning more closely with the dynamic, evolving nature of real-world deployment scenarios.

8.2 Method

We pose our investigation as a continual learning regression problem. Given a sequence of T distinct tasks, $\mathcal{T} = \{\tau_1, \ldots, \tau_T\}$, each task τ_i is associated with a dataset $\mathcal{D}_i = \{(\mathbf{x}_j, y_j)\}$, where $\mathbf{x}_j$ is an input sample and y_j is its corresponding target value. Each label y_j represents the stress level self-reported by the student on a given day, and the input sample $\mathbf{x}_j$ is the sequence of sensing data $\{\mathbf{x}_{j,1}, \mathbf{x}_{j,2}, \ldots, \mathbf{x}_{j,K}\}$ collected over the K days preceding that report. We operate in both a domain-incremental learning setting, where each task τ_i corresponds to a specific subset of students. While the input distribution changes across tasks, the output space remains fixed, with stress levels ranging in a shared label space $\mathcal{Y} = \{1, 2, 3, 4, 5\}$, where higher values indicate higher perceived stress.

The objective is to learn a function $F : \mathcal{X} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta}$, that generalizes across tasks, while at each task τ_i the model is exposed only to data from the corresponding distribution $\mathcal{D}_i$, i.e., $(\mathbf{x}, y) \sim \mathcal{D}_i$.

Our methodology draws inspiration from two main paradigms in continual learning: memory replay (*rehearsal*) with task-agnostic models [103, 131, 148], and recent developments in prompt-based adaptation for transformers [104, 105]. An overview of the proposed method is presented in Fig. 8.1.

8.2.1 Buffer-based approach

To reduce forgetting from previous tasks, buffer-based methods store a limited number of samples from previous tasks in a *memory buffer* $\mathcal{M}$. The model update can be summarized as:

$$\langle F, \boldsymbol{\theta}_{i-1}, \mathcal{M}_{i-1} \rangle \xrightarrow{\mathcal{D}_i} \langle F, \boldsymbol{\theta}_i, \mathcal{M}_i \rangle, \tag{8.1}$$

where $\boldsymbol{\theta}_i$ and $\mathcal{M}_i$ denote the model parameters and the memory buffer at the end of task τ_i, respectively.

The training objective is to minimize a regression loss over the sequence of tasks, aiming to preserve performance on both current and previous tasks:

$$\begin{aligned} \arg\min_{\boldsymbol{\theta}_T} \alpha \sum_{i=1}^{T} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_i} \Big[\mathcal{L}\Big(F\left(\mathbf{x}; \boldsymbol{\theta}_T\right), y \Big) \Big] \\ + \beta \sum_{i=1}^{T} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{M}_i} \Big[\mathcal{L}\Big(F\left(\mathbf{x}; \boldsymbol{\theta}_T\right), y \Big) \Big] \end{aligned} \tag{8.2}$$

where $\mathcal{L}$ is a regression loss function (e.g., Mean Squared Error), and α and β are weighting coefficients that balance the contributions of the two loss components.

8.2.2 Prompt-based approach

As introduced in Sec. 2.5.4, transformers have recently been increasingly adopted in continual learning research, thanks to the capability to condition their behavior through learnable prompt tuning: in this setting, task adaptation is achieved by tuning small *task-specific prompt vectors* while keeping a pre-trained backbone frozen. Since pre-training is not applicable in our setting due to the nature of the data, we instead jointly train the backbone and the prompts across tasks, with a supporting memory buffer to ensure alignment between prompts learned in older tasks and the continually-updated backbone. In this way, we are able to extend the replay-based approach defined in the previous section with a strong and standardized conditioning mechanism.

In this formulation, a transformer $\mathcal{T}$ is employed as the backbone encoder, projecting the sequence of sensing features into a d-dimensional representation. For each task τ_i, the input sequence $\mathbf{x}_j$ is augmented by prepending a set of N learnable task-specific prompts, $P_i = \{\mathbf{p}_1^i, \ldots, \mathbf{p}_N^i\}$. The resulting input to the transformer is defined as:

$$\mathcal{T}(P_i \,\|\, \mathbf{x}_j), \tag{8.3}$$

where $\|$ denotes concatenation along the temporal axis.

During training, both the backbone transformer $\mathcal{T}$ and the parameters of each prompt P_i are updated. As mentioned above, to ensure that the prompts $\{P_1, \ldots, P_{i-1}\}$ associated with previous tasks $\{\tau_1, \ldots, \tau_{i-1}\}$ remain well-adapted, a memory buffer $\mathcal{M}_i$ is employed to store samples from past tasks, which are replayed during training on task τ_i to refine the corresponding prompts. At inference time, the task identity is assumed to be known a priori, allowing the model to use the corresponding learned prompt set.

The learning objective remains the same as in Equation 8.2, with the optimization extended to include the prompt parameters.

8.3 Experimental results

8.3.1 Datasets and metrics

We used the StudentLife dataset [223], which includes multimodal data from 48 undergraduate and graduate students at Dartmouth College over a 10-week spring term. The dataset contains over 53 GB of continuous sensor data, 32,000 self-reports, and several pre/post-study surveys covering mental health, lifestyle factors, and academic-related attitudes.

Given that self-reported stress is the target variable in our study, we focused on the corresponding questionnaire data provided by each participant. However, as not all participants contributed responses evenly across the 10-week period, we applied a filtering step to ensure both temporal consistency and sufficient response density. We retained only those who completed at least 30 stress questionnaires within a time window of 60 to 75 days, resulting in a final subset of 16 students. Additionally, we excluded one outlier who submitted an high number of responses, many of which extended beyond the expected 10-week timeframe. Figure 8.2 reports the number of days with available stress self-report data for each participant, while Fig. 8.3 reports the number of stress responses for the selected subset.

Data processing. Among the various data types available in the StudentLife dataset, we selected a limited subset of features consisting of *phone call duration*, *conversation*, *physical activity*, *app usage* and *academic deadlines*.

To streamline the analysis and mitigate the risk of bias or information leakage, we excluded features deemed redundant or less informative (e.g., additional phone data, location information, and seating logs). Instead, we focused on behavioral and contextual signals while avoiding variables that may directly reveal or strongly correlate with the target outcome.

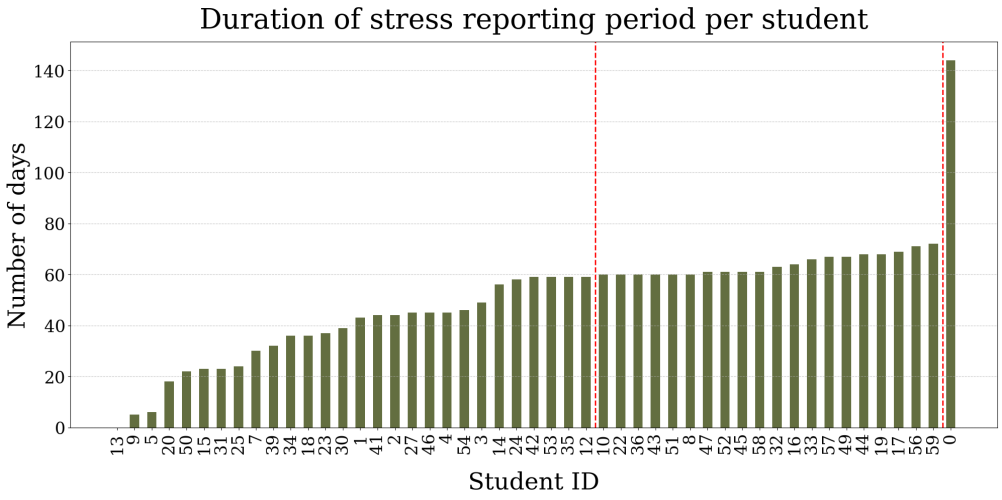

Figure 8.2: **Length of the stress self-reporting period across students.** Students to the right of the red dashed line are retained in the final subset of 16 students, while those to the left are excluded.

The selected features were extracted and aggregated per day, as described below:

- **Phone call duration** includes both the total time spent on completed incoming and outgoing calls, expressed in minutes, and the number of such interactions. Missed or rejected calls are excluded, as they do not reflect active engagement.

- **Conversation** consists of time periods inferred from microphone data via acoustic analysis as indicative of nearby spoken interactions. Both the number of detected periods and their total duration in minutes were extracted.

- **Physical activity** is derived from smartphone sensor data using a classifier that operates continuously with duty cycling to optimize energy consumption. The resulting activity is classified into three categories: stationary, walking, and running. The total daily duration in minutes was computed for each activity state.

- **App usage** captures the set of mobile applications accessed by the participant. To organize this data meaningfully, a clustering approach was applied using BART [243], a transformer-based model capable of generating semantically rich representations. Each app was assigned to one of ten predefined semantic categories: “social media and communication”, “productivity and business”, “entertainment and media”, “health and fitness”, “finance and shopping”, “utilities and tools”, “games”, “education”, “travel and navigation”, and “photography”. Based on this categorization, two features were computed for each day: the overall time spent using the phone, and the usage time in minutes for each category.

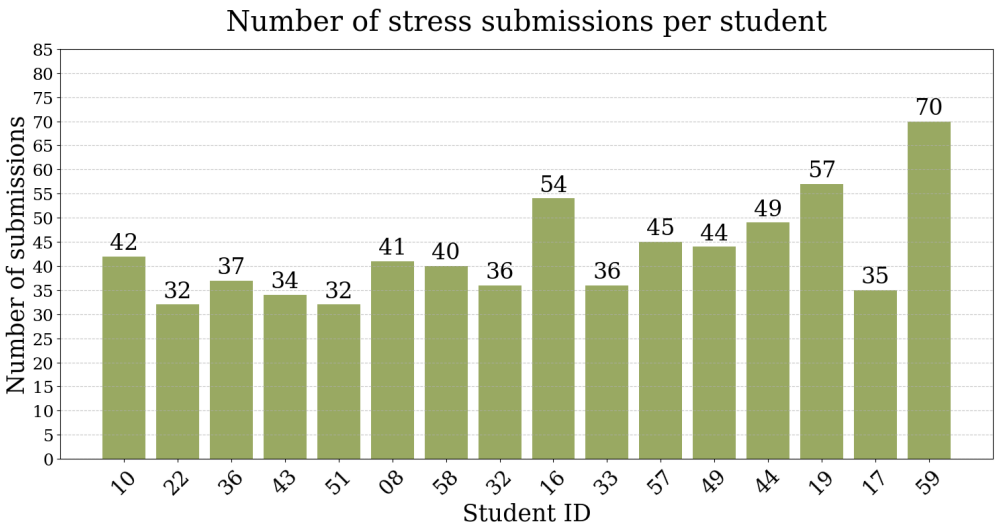

Figure 8.3: **Number of stress self-report submissions in the filtered subset.** For each of the 16 selected students, at least 30 submissions are available over the 10-week period. Values above the bars indicate exact counts.

- **Academic deadlines** represent the daily number of academic events scheduled for each participant, including assignments, quizzes, projects, and exams. These data were obtained from a structured log of time-stamped academic events.

Sensor-based features, such as *conversation*, may suffer from occasional missing data due to sensor unavailability, battery optimization mechanisms, or interruptions in background processes. To ensure a consistent temporal resolution, we applied linear interpolation to estimate missing values on a per-day basis.

Stress levels, ranging from 1 to 5, were obtained from participants' self-reports, with higher values indicating greater perceived stress. To predict the stress level on a given day, the input consists of a sequence of K feature vectors, each representing the full set of the aforementioned daily variables over the K preceding days.

Metrics. To evaluate performance, we use two standard regression metrics: Mean Squared Error (MSE) and Mean Absolute Error (MAE). For each metric, we provide the mean and standard deviation on the test split defined by the employed datasets, over three runs for all experiments.

8.3.2 Architectures

To investigate the effectiveness of the proposed approaches, we employed the following sequence modeling architectures:

- **Recurrent Neural Network (RNN)** [227]: a two-layer unidirectional vanilla RNN with a hidden size of 64 and a dropout rate of 0.3. The model processes input sequences one timestep at a time, maintaining a hidden state to capture temporal dependencies. Due to its simple recurrent structure, it is susceptible to vanishing gradients when modeling long-range dependencies. We train this model following the buffer-based approach described in Sect. 8.2.1.

- **Bidirectional Long Short-Term Memory (BiLSTM)** [228]: a two-layer bidirectional LSTM with a hidden size of 96 per direction and a dropout rate of 0.3. Unlike the vanilla RNN, the LSTM incorporates gating mechanisms and a dedicated cell state to preserve information over longer temporal windows. The bidirectional configuration enables the model to leverage both past and future context at each timestep, enhancing its ability to encode structured sequential data. We train this model following the buffer-based approach described in Sect. 8.2.1.

- **Transformer encoder** [229]: a two-layer transformer encoder, with each layer comprising four attention heads and an embedding dimension of 256. A final linear layer follows the encoder, projecting the internal representations into a single output dimension. To preserve the temporal structure of the input, positional encodings are added to the input embeddings. We train this model following the prompt-based approach described in Sect. 8.2.2.

Note that, while for the recurrent architectures this evaluation setting is consistent with a pure domain-incremental scenario, the transformer encoder operates in a joint domain- and task-incremental setting, as prompt selection at inference time depends on task identity.

8.3.3 Training procedure

We split the available data into training, validation, and test sets using a 70%-10%-20% ratio, ensuring that each student's data is evenly distributed across the splits.

Training is carried out for 50 epochs per task using the Adam optimizer [132], with a learning rate of 0.001 and a batch size of 32, setting $\alpha = 1$ and $\beta = 0.5$ for the training objective defined in Eq. 8.2. All hyperparameters were selected based on performance on the validation set.

Given the limited number of samples per task, we employ relatively small buffer sizes $|\mathcal{M}|$, experimenting with buffers of 20, 50, and 100 samples.

Joint training		
	MSE	**MAE**
RNN [227]	1.905±0.081	1.131±0.040
BiLSTM [228]	1.834±0.055	1.097±0.021
Transformer w/ $\|P\|= 4$	1.866±0.030	1.021±0.007
Transformer w/ $\|P\|= 8$	1.077±0.032	0.844±0.012
Transformer w/ $\|P\|= 16$	1.161±0.035	0.778±0.014
Transformer w/ $\|P\|= 32$	1.159±0.017	0.827±0.013
Transformer w/ $\|P\|= 64$	1.235±0.085	0.919±0.040

Table 8.1: **Performance of joint training across different model architectures.** Comparison of RNN, BiLSTM, and transformer with varying prompt set size $|P|$. Performance is reported in terms of MSE (↓) and MAE (↓), including standard deviations.

8.3.4 Results

In this section, we present the results of our stress prediction experiments, comparing buffer-based methods with a prompt-augmented variant. All models are evaluated using the performance metrics defined in Sec. 2.4.

To establish meaningful baselines for our continual learning methods, Tab. 8.1 reports results obtained by training each model on the full dataset, using the architectures described in Sec. 8.3.2. As a lower bound, Tab. 8.2 shows results from continual learning experiments without any replay buffer. The baseline results indicate that recurrent architectures yield the highest prediction errors, reflecting their limited ability to model long-range task dependencies. Transformer-based encoders consistently outperform recurrent models, achieving lower error rates in both joint and continual settings. In the joint training setup, the best performance is achieved with a prompt set size of $|P|= 8$, while larger sizes provide no further benefit and may even degrade performance, suggesting over-parameterization.

We evaluated our approaches in a continual learning setting by splitting the 16 students into different numbers of tasks. The results for the 16-task, 8-task, and 4-task configurations are presented in Tab. 8.3, Tab. 8.4, and Tab. 8.5, respectively.

Across all tested configurations, transformer-based architectures consistently outperform recurrent models, achieving significantly lower error metrics. Minimal prompt configurations ($|P_i|= 1$ or $|P_i|= 2$) yield the best results, whereas larger prompt sets often degrade performance, indicating potential overfitting or over-parameterization. Introducing even a single prompt per task improves transformer performance across all scenarios compared to their non-prompted counterparts, although this comes at the cost of requiring the task identifier

No rehearsal training		
	16 tasks: one student per task	
	MSE	**MAE**
RNN [227]	3.335±0.118	1.251±0.038
BiLSTM [228]	3.045±0.098	1.181±0.013
Transformer w/o prompts	2.185±0.009	1.102±0.004
	8 tasks: two students per task	
	MSE	**MAE**
RNN [227]	2.941±0.100	1.108±0.036
BiLSTM [228]	2.684±0.071	1.086±0.023
Transformer w/o prompts	1.832±0.008	1.021±0.004
	4 tasks: four students per task	
	MSE	**MAE**
RNN [227]	2.639±0.081	1.184±0.026
BiLSTM [228]	2.415±0.074	1.105±0.017
Transformer w/o prompts	1.704±0.010	0.985±0.006

Table 8.2: **Performance of continual training without rehearsal**: a comparison of RNN, BiLSTM, and transformer architectures. Performance is reported in terms of MSE (↓) and MAE (↓), including standard deviations.

	16 tasks: one student per task					
	$\lvert\mathcal{M}\rvert$= **20**		$\lvert\mathcal{M}\rvert$= **50**		$\lvert\mathcal{M}\rvert$= **100**	
	MSE	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
RNN [227]	2.644±0.310	1.109±0.015	2.543±0.383	1.096±0.016	2.703±0.169	1.107±0.008
BiLSTM [228]	2.657±0.103	1.116±0.007	2.705±0.012	1.109±0.005	2.696±0.038	1.110±0.005
Tx w/ $\lvert P_i\rvert$= 1	2.054±0.003	1.083±0.001	2.046±0.006	1.051±0.006	2.045±0.003	1.014±0.001
Tx w/ $\lvert P_i\rvert$= 2	2.157±0.171	1.086±0.001	2.052±0.010	1.084±0.001	2.029±0.011	1.069±0.012
Tx w/ $\lvert P_i\rvert$= 4	2.938±0.051	1.269±0.031	2.596±0.461	1.158±0.001	2.318±0.084	1.103±0.022

Table 8.3: **Comparison of buffer-only and prompt-augmented approaches for stress prediction across 16 tasks, with one student per task.** Results are reported by varying the memory buffer size ($|\mathcal{M}|$). In the prompt-based setting, "Tx" denotes the transformer encoder and $|P_i|$ the number of learned prompts per task. Performance is reported in terms of MSE (↓) and MAE (↓), with standard deviation.

	8 tasks: two students per task					
	$\|\mathcal{M}\|$= **20**		$\|\mathcal{M}\|$= **50**		$\|\mathcal{M}\|$= **100**	
	MSE	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
RNN [227]	2.202±0.244	0.963±0.026	2.093±0.209	0.987±0.045	2.381±0.054	0.976±0.018
BiLSTM [228]	2.298±0.078	1.004±0.029	2.289±0.022	0.985±0.004	2.245±0.017	0.978±0.003
Tx w/ $\|P_i\|$= 1	1.707±0.018	0.980±0.001	1.762±0.106	0.976±0.008	1.688±0.021	0.934±0.032
Tx w/ $\|P_i\|$= 2	1.730±0.050	0.976±0.009	1.719±0.039	0.952±0.039	1.700±0.004	0.932±0.013
Tx w/ $\|P_i\|$= 4	1.948±0.050	0.964±0.010	1.929±0.120	0.963±0.006	1.915±0.032	0.948±0.016

Table 8.4: **Comparison of buffer-only and prompt-augmented approaches for stress prediction across 8 tasks, with two students per task.** Results are reported by varying the memory buffer size ($|\mathcal{M}|$). In the prompt-based setting, "Tx" denotes the transformer encoder and $|P_i|$ the number of learned prompts per task. Performance is reported in terms of MSE (↓) and MAE (↓), with standard deviation.

	4 tasks: four students per task					
	$\|\mathcal{M}\|$= **20**		$\|\mathcal{M}\|$= **50**		$\|\mathcal{M}\|$= **100**	
	MSE	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
RNN [227]	1.462±0.196	0.968±0.039	2.236±0.269	1.097±0.061	2.431±0.095	1.091±0.038
BiLSTM [228]	2.048±0.031	1.040±0.005	2.042±0.058	1.042±0.017	2.047±0.045	1.044±0.014
Tx w/ $\|P_i\|$= 1	1.565±0.069	0.948±0.001	1.529±0.015	0.945±0.004	1.354±0.052	0.930±0.015
Tx w/ $\|P_i\|$= 2	1.540±0.023	0.943±0.004	1.512±0.001	0.933±0.006	1.499±0.055	0.927±0.019
Tx w/ $\|P_i\|$= 4	1.634±0.103	0.941±0.003	1.581±0.025	0.920±0.007	1.513±0.031	0.900±0.017

Table 8.5: **Comparison of buffer-only and prompt-augmented approaches for stress prediction across 4 tasks, with four students per task.** Results are reported by varying the memory buffer size ($|\mathcal{M}|$). In the prompt-based setting, "Tx" denotes the transformer encoder and $|P_i|$ the number of learned prompts per task. Performance is reported in terms of MSE (↓) and MAE (↓), with standard deviation.

at inference time.

Performance generally benefits from increasing the memory buffer size $|\mathcal{M}|$, yet the improvement becomes marginal for very large buffers ($|\mathcal{M}|$= 100), suggesting that the proposed methods can achieve competitive results even with reduced memory capacity. This advantage is particularly relevant in real-world stress monitoring applications where storage resources are limited.

Overall, these findings highlight the benefits of combining transformer architectures with

prompt-based replay and underscore the importance of carefully tuning the number of learned prompts per task to avoid performance degradation.

8.4 Discussion

In this chapter, we addressed the challenge of stress prediction in real-world scenarios, aiming to overcome the limitations of static and traditional machine learning models. To mimic realistic distribution shifts, we organized the data into tasks comprising one, two, or four students each. Within this framework, we investigated two adaptation strategies: a memory-based replay method employing recurrent neural networks, and a prompt-based approach leveraging a transformer backbone with task-specific tunable prompts.

Experiments on a subset of the StudentLife dataset demonstrate that, despite the need to store and retrieve task-specific prompts at inference time, prompt-based adaptation achieves lower prediction error than buffer-only replay methods. Optimal performance is obtained with minimal prompt sets, whereas larger configurations may introduce unnecessary complexity. Moreover, performance remains competitive even with reduced memory, underscoring the practicality of the approach for resource-constrained stress monitoring scenarios.

A key limitation identified in our analysis is the tendency of the model to underfit in certain tasks. This limitation primarily arises from the low reliability of the multimodal data collected via smartphones. The stress target exhibits high short-term variability, which constrains the model's ability to learn stable and meaningful patterns. Data quality is further compromised by artifacts such as environmental noise, intermittent signal loss, and inconsistent device usage by participants. As a result, the model struggles to learn robust representations and to generalize beyond the training data. To address these limitations, possible key research directions include designing more resilient multimodal fusion architectures and collecting higher-quality datasets.

Moreover, future work may explore more advanced prompting strategies, methods for reducing or removing the need for explicit task identifiers, and broader applications of the proposed framework across different domains of mental health and human behavior modeling.

8.5 Publications

Patanè, G., Sorrenti, A., Bellitto, G., & Palazzo, S. (2025, October). "Continual Learning Strategies for Personalized Mental Well-being Monitoring from Mobile Sensing Data". In *Proceedings of the International Workshop on Personalized Incremental Learning in Medicine* (pp. 9-17).

Conclusions

This thesis addressed continual learning from a different perspective, investigating how brain-inspired principles could be translated into practical mechanisms to mitigate catastrophic forgetting. In spite of a wide range of existing methods, a marked gap persists between artificial neural networks and the human ability to acquire, consolidate, and reuse knowledge over time. Motivated by this gap, this dissertation drew inspiration from brain mechanisms that reduce representational interference and support memory consolidation.

Consistent with selective neural coding and pattern separation, Chapter 3 demonstrated that shaping the geometry of learned representations via an architectural inductive bias can foster feature selectivity. Moreover, the study of model-inherent explanations revealed a systematic trade-off between predictive performance and the stability of explanation maps, and suggested that interpretability can serve as a complementary tool to study when and where forgetting emerges.

Inspired by multi-timescale synaptic consolidation, Chapter 4 related efficiency to the duration and location of plasticity within the network, showing that many learned representations remain reusable over time. To this end, it proposed a selective freezing strategy that adaptively stabilizes subsets of the backbone while retaining sufficient flexibility to learn new tasks. Chapter 5 extended this strategy by introducing memory consolidation mechanisms based on the Complementary Learning Systems theory and the role of the wake-sleep cycle. By emulating these processes, the proposed framework showed that offline consolidation can effectively prepare the system to learn subsequent tasks by leveraging previously acquired knowledge, as reflected by positive forward transfer. To overcome the reliance on external data sources to simulate "dream-like" experiences, Chapter 6 introduced a structured generative dreaming mechanism that synthesizes training signals from the network's own internal knowledge and latent structure. The obtained results suggested that mechanisms often neglected in standard continual learning pipelines, such as offline consolidation, can be effective not only in biological systems but also in artificial neural networks when incorporated into a bio-inspired formulation.

The last part of the thesis examined continual learning in two applied scenarios. Chap-

ter 7 investigated incremental object 6D pose estimation for robotic grasping, while Chapter 8 studied personalized mental well-being monitoring based on mobile sensing data. Across both applications, the results demonstrated that methods originally developed for image classification could be transferred to regression case studies via lightweight adaptations, highlighting their versatility in real-world scenarios.

The limitations discussed throughout the thesis point to several directions for future research. A deeper understanding of the relationship between angular selectivity and feature sparsity could support a more principled design of architectural inductive biases for continual learning, while also guiding the choice of the hyperparameter B. Despite its current effectiveness and efficiency, selective freezing could be improved by introducing finer-grained freezing strategies at the parameter or sub-module level, and by assessing its generality across a wider range of architectures and datasets. As for CLS-inspired approaches, although WSCL and Dream2Learn brought offline consolidation into continual learning, they still rely on simplified computational abstractions of biological processes. In WSCL, memory modeling could move beyond conventional rehearsal mechanisms toward formulations that better capture memory dynamics, while in Dream2Learn the generative process could be further refined to make dream generation more efficient and to mitigate the risk of future-class leakage through the pretrained generator. Future work on real-world applications should extend the present findings by tackling domain-specific challenges that remain open, including symmetry-aware and uncertainty-aware modeling for robotic perception and manipulation, as well as more robust multimodal fusion and higher-quality longitudinal datasets for personalized health monitoring.

This dissertation provided evidence that the brain is not only a source of metaphorical inspiration for continual learning, but also a guide for designing mechanisms that shape representations, regulate plasticity, and leverage offline consolidation to support retention and future learning. By combining architectural inductive bias, efficient plasticity control, and structured generative dreaming, this work established a solid foundation for future brain-inspired continual learning, encouraging the development of methods that increasingly build on mechanisms underlying human learning.

List of publications

[1] Amelia Sorrenti, Giovanni Bellitto, Federica Proietto Salanitri, Matteo Pennisi, Concetto Spampinato, and Simone Palazzo. Selective freezing for efficient continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3550–3559, 2023.

[2] Amelia Sorrenti, Giovanni Bellitto, Federica Proietto Salanitri, Matteo Pennisi, Simone Palazzo, and Concetto Spampinato. Wake-sleep consolidated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[3] Amelia Sorrenti, Giovanni Bellitto, Salvatore Calcagno, Rutger Hendrix, Concetto Spampinato, and Simone Palazzo. B-cos networks as an architectural inductive bias for mitigating catastrophic forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5205–5213, 2025.

[4] Long Tian, Amelia Sorrenti, Yik Lung Pang, Giovanni Bellitto, Simone Palazzo, Concetto Spampinato, and Changjae Oh. Incremental object 6d pose estimation. In *International Conference on Pattern Recognition*, pages 331–346. Springer, 2024.

[5] Giovanni Patanè, Amelia Sorrenti, Giovanni Bellitto, and Simone Palazzo. Continual learning strategies for personalized mental well-being monitoring from mobile sensing data. In *Proceedings of the International Workshop on Personalized Incremental Learning in Medicine*, pages 9–17, 2025.

[6] Federica Proietto Salanitri, Giovanni Bellitto, Raffaele Mineo, Matteo Pennisi, Amelia Sorrenti, Salvatore Calcagno, Daniela Giordano, Simone Palazzo, and Concetto Spampinato. Dynamic graph attention: Unraveling spatio-temporal synchrony in eeg data. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2201–2206. IEEE, 2023.

[7] Raffaele Mineo, Amelia Sorrenti, Gaia Caligiore, Federica Proietto Salanitri, Giovanni Bellitto, Senya Polikovsky, Sabina Fontana, Egidio Ragonese, Concetto Spampinato,

and Simone Palazzo. Text-aligned radar-based sign language recognition for healthcare communication. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4894–4902, 2025.

[8] Raffaele Mineo, Amelia Sorrenti, and Federica Proietto Salanitri. Fedetr: A federated approach for stenosis detection in coronary angiography. In *International Conference on Image Analysis and Processing*, pages 189–200. Springer, 2023.

[9] Marco Finocchiaro, Salvatore Calcagno, Isaak Kavasidis, Amelia Sorrenti, Silvia Suchalova, Robert Hudec, Manuela Pennisi, Federica Proietto Salanitri, and Concetto Spampinato. From questions to neural insights: Towards query-based fmri decoding. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference*, volume 2025, pages 1–7, 2025.

[10] Amelia Sorrenti, Leonardo Giuseppe Russo, Sarinda Samarasinghe, Simone Palazzo, and Concetto Spampinato. Weakly supervised maxn estimation in baited remote underwater video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2123–2131, 2025.

[11] Isaak Kavasidis, Amelia Sorrenti, Orazio Tomarchio, Daniela Giordano, Marco Milazzo, Gabriele Turco, Carlo Cattano, and Concetto Spampinato. Training-free fish species population monitoring in unconstrained underwater videos. *Procedia Computer Science*, 257:809–816, 2025.

Bibliography

[12] G.M.A. Grube. *Meno*. Hackett Classics. Hackett Publishing Company, Incorporated, 1980.

[13] Plato. *Plato's Phaedo*. Clarendon Press, Oxford, 1911.

[14] René Descartes. *Meditationes de prima philosophia*. Apud Danielem Elsevirum, 1937.

[15] Jonathan Barnes. *Posterior analytics*, volume 1. Clarendon Press Oxford, 1994.

[16] David W Hamlyn. Aristotle's account of aesthesis in the de anima. *The Classical Quarterly*, 9(1):6–16, 1959.

[17] John Locke. *An essay concerning human understanding*. Kay & Troutman, 1847.

[18] Immanuel Kant. *Critique of pure reason*. Cambridge university press, 1999.

[19] G Carpenter and Stephen Grossberg. Adaptive resonance theory: Stable self-organization of neural recognition codes in response to arbitrary lists of input patterns. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 45–62. Erlbaum, 1986.

[20] Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186, 1987.

[21] Richard S Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. of Eightth Annual Conference of the Cognitive Science Society*, pages 823–831, 1986.

[22] James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel distributed processing, volume 2: Explorations in the microstructure of cognition: Psychological and biological models*, volume 2. MIT press, 1987.

[23] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[24] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 1990.

[25] Stephen Grossberg. *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition and Motor Control*. Springer Science & Business Media, 1982.

[26] Alexandre Pouget, Peter Dayan, and Richard Zemel. Information processing with population codes. *Nature Reviews Neuroscience*, 1(2):125–132, 2000.

[27] Peter Dayan, Laurence F Abbott, et al. Theoretical neuroscience: computational and mathematical modeling of neural systems. *Journal of Cognitive Neuroscience*, 15(1):154–155, 2003.

[28] Peter Foldiak. Sparse coding in the primate cortex. *The handbook of brain theory and neural networks*, 2003.

[29] Michael Beyeler, Emily L Rounds, Kristofor D Carlson, Nikil Dutt, and Jeffrey L Krichmar. Neural correlates of sparse coding and dimensionality reduction. *PLoS computational biology*, 15(6):e1006908, 2019.

[30] Yadin Dudai. The neurobiology of consolidations, or, how stable is the engram? *Annu. Rev. Psychol.*, 55(1):51–86, 2004.

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

[32] Marcus K Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nature neuroscience*, 19(12):1697–1706, 2016.

[33] Michael A Yassa and Craig EL Stark. Pattern separation in the hippocampus. *Trends in neurosciences*, 34(10):515–525, 2011.

[34] Antoine D Madar, Laura A Ewell, and Mathew V Jones. Pattern separation of spike-trains in hippocampal neurons. *Scientific Reports*, 9(1):5282, 2019.

[35] Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends Cogn Sci*, 20(7):512–534, Jul 2016.

[36] James L McClelland, Bruce L McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychol Rev*, 102(3):419–457, Jul 1995.

[37] Anna C Schapiro, Nicholas B Turk-Browne, Matthew M Botvinick, and Kenneth A Norman. Complementary learning systems within the hippocampus: a neural network modelling approach to reconciling episodic memory with statistical learning. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1711):20160049, 2017.

[38] Daoyun Ji and Matthew A. Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nat Neurosci*, 10(1):100–107, Jan 2007.

[39] Matthew P. Walker and Robert Stickgold. Sleep-dependent learning and memory consolidation. *Neuron*, 44(1):121–133, Sep 2004.

[40] Dhairyya Singh, Kenneth A. Norman, and Anna C. Schapiro. A model of autonomous interactions between hippocampus and neocortex driving sleep-dependent memory consolidation. *Proceedings of the National Academy of Sciences of the United States of America*, 119(44), November 2022.

[41] Mircea Steriade, David A. McCormick, and Terrence J. Sejnowski. Thalamocortical oscillations in the sleeping and aroused brain. *Science*, 262(5134):679–685, Oct 1993.

[42] Giri P. Krishnan, Sylvain Chauvette, Isaac Shamie, Sara Soltani, Igor Timofeev, Sydney S. Cash, Eric Halgren, and Maxim Bazhenov. Cellular and neurochemical basis of sleep stages in the thalamocortical network. *Elife*, 5, Nov 2016.

[43] Nicolas Deperrois, Mihai. A. Petrovici, Walter Senn, and Jakob Jordan. Learning cortical representations through perturbed and adversarial dreaming. *Elife*, 11, Apr 2022.

[44] Camilo Giedelman, Marcio Covas Moschovas, Seetharam Bhat, Lauren Brunelle, Gabriel Ogaya-Pinies, Shannon Roof, Cathy Corder, Vipul Patel, and Kenneth J

Palmer. Establishing a successful robotic surgery program and improving operating room efficiency: literature review and our experience report. *Journal of robotic surgery*, 15(3):435–442, 2021.

[45] Iulia Andras, Elio Mazzone, Fijs WB van Leeuwen, Geert De Naeyer, Matthias N van Oosterom, Sergi Beato, Tessa Buckle, Shane O'Sullivan, Pim J van Leeuwen, Alexander Beulens, et al. Artificial intelligence and robotics: a combination that is changing the operating room. *World journal of urology*, 38:2359–2366, 2020.

[46] Yeisson Rivero-Moreno, Sophia Echevarria, Carlos Vidal-Valderrama, Luigi Pianetti, Jesus Cordova-Guilarte, Jhon Navarro-Gonzalez, Jessica Acevedo-Rodríguez, Gabriela Dorado-Avila, Luisa Osorio-Romero, Carmen Chavez-Campos, et al. Robotic surgery: a comprehensive review of the literature and current trends. *Cureus*, 15(7), 2023.

[47] Andrea Moglia, Konstantinos Georgiou, Evangelos Georgiou, Richard M Satava, and Alfred Cuschieri. A systematic review on artificial intelligence in robot-assisted surgery. *International Journal of Surgery*, 95:106151, 2021.

[48] Shih-Yu Lee, Caroline Wuertz, Rebecca Rogers, and Yu-Ping Chen. Stress and sleep disturbances in female college students. *American journal of health behavior*, 37(6):851–858, 2013.

[49] Saba Asif, Azka Mudassar, Talala Zainab Shahzad, Mobeen Raouf, and Tehmina Pervaiz. Frequency of depression, anxiety and stress among university students. *Pakistan journal of medical sciences*, 36(5):971, 2020.

[50] Na Li, Yan Wang, Yijiao Dong, Xiaoxue Chen, Bin Zhang, Xianghua Chen, Kejian Wang, and Ying Sun. The impact of psychological stress on physiological indicators in healthcare workers: a cross-sectional study. *Frontiers in Public Health*, 12:1393743, 2024.

[51] Gido M van de Ven and Andreas S Tolias. Three continual learning scenarios. In *Neural Information Processing Systems Workshops*, 2018.

[52] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2009.

[53] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[54] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[55] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, 2011.

[56] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[57] Ian J Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *stat*, 1050:4, 2015.

[58] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

[59] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[60] Stanford. Tiny ImageNet Challenge (CS231n), 2015. https://www.kaggle.com/c/tiny-imagenet.

[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[62] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8340–8349, 2021.

[63] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[64] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[65] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

[66] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

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

[68] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Neural Information Processing Systems Workshops*, 2015.

[69] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

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

[71] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1320–1328, 2017.

[72] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

[73] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018.

[74] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[75] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision*, pages 139–154, 2018.

[76] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[77] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.

[78] Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[79] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

[80] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

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

[82] Woo-Young Kang and Byoung-Tak Zhang. Continual learning with generative replay via discriminative variational autoencoder. In *NeurIPS workshop on continual learning*, volume 1, 2018.

[83] Valeriya Khan, Sebastian Cygert, Bartlomiej Twardowski, and Tomasz Trzciński. Looking through the past: better knowledge retention for generative replay in continual learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3496–3500, 2023.

[84] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6619–6628, 2019.

[85] Rui Gao and Weiwei Liu. Ddgr: Continual learning with deep diffusion-based generative replay. In *International Conference on Machine Learning*, pages 10744–10763. PMLR, 2023.

[86] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

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

[88] 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.

[89] Ari S Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. In *International Conference on Learning Representations*, 2019.

[90] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019.

[91] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *NeurIPS*, 2019.

[92] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations*, 2019.

[93] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019.

[94] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[95] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. GDumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

[96] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In *NeurIPS*, 2020.

[97] Matteo Boschini, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Continual semi-supervised learning through contrastive interpolation consistency. *Pattern Recognition Letters*, 162:9–14, 2022.

[98] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo. Class-incremental learning with pre-allocated fixed classifiers. In *International Conference on Pattern Recognition*, 2021.

[99] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. In *Advances in Neural Information Processing Systems*, 2021.

[100] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ale Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[101] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *IEEE International Conference on Computer Vision*, 2021.

[102] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems*, 2020.

[103] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New Insights on Reducing Abrupt Representation Change in Online Continual Learning. In *International Conference on Learning Representations*, 2022.

[104] 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.

[105] 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 *Proceedings of the European Conference on Computer Vision*, pages 631–648. Springer, 2022.

[106] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual

learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11909–11919, 2023.

[107] Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11483–11493, 2023.

[108] Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024.

[109] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[110] Moritz Böhle, Mario Fritz, and Bernt Schiele. B-cos networks: Alignment is all we need for interpretability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10329–10338, 2022.

[111] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[112] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[113] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

[114] K Simonyan, A Vedaldi, and A Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR, 2014.

[115] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[116] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks

via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[117] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[118] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[119] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.

[120] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

[121] Moritz Bohle, Mario Fritz, and Bernt Schiele. Convolutional dynamic alignment networks for interpretable classifications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10029–10038, 2021.

[122] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020.

[123] Emanuele Marconato, Andrea Passerini, and Stefano Teso. Glancenets: Interpretable, leak-proof concept-based models. *Advances in Neural Information Processing Systems*, 35:21212–21227, 2022.

[124] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019.

[125] Andrea Cossu, Francesco Spinnato, Riccardo Guidotti, and Davide Bacciu. Drifting explanations in continual learning. *Neurocomputing*, 597:127960, 2024.

[126] Dawid Rymarczyk, Joost Van De Weijer, Bartosz Zieliński, and Bartlomiej Twardowski. Icicle: Interpretable class incremental continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1887–1898, 2023.

[127] Lu Yu, Haoyu Han, Zhe Tao, Hantao Yao, and Changsheng Xu. Language guided concept bottleneck models for interpretable continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14976–14986, 2025.

[128] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.

[129] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

[130] O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, A Karpathy, A Khosla, M Bernstein, et al. Imagenet large scale visual recognition challengeint. *J. Comput. Vis*, 115(3):21, 2014.

[131] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

[132] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[133] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *British Machine Vision Conference (BMVC)*, 2018.

[134] Divya Pandey, Madhoolika Agrawal, and Jai Shanker Pandey. Carbon footprint: current methods of estimation. *Environmental monitoring and assessment*, 178(1):135–160, 2011.

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

[136] Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *ICML*, 2022.

[137] Marc Masana, Tinne Tuytelaars, and Joost Van de Weijer. Ternary feature masks: zero-forgetting for task-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3570–3579, 2021.

[138] Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34:5250–5263, 2021.

[139] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[140] Yujun Shi, Li Yuan, Yunpeng Chen, and Jiashi Feng. Continual learning via bit-level information preserving. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 16674–16683, 2021.

[141] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. In *Advances in Neural Information Processing Systems*, 2020.

[142] Fu-En Yang, Chien-Yi Wang, and Yu-Chiang Frank Wang. Efficient model personalization in federated learning via client-specific prompt generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19159–19168, 2023.

[143] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2009.

[144] Jeremy Howard. Imagenette.

[145] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[146] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

[147] Emanuele Frascaroli, Riccardo Benaglia, Matteo Boschini, Luca Moschella, Cosimo Fiorini, Emanuele Rodolà, and Simone Calderara. Latent spectral regularization for continual learning. *Pattern Recognition Letters*, 184:119–125, 2024.

[148] Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6993–7001, 2021.

[149] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8250–8259, October 2021.

[150] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2021.

[151] Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *International Conference on Machine Learning*, 2022.

[152] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

[153] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint*, 2022.

[154] Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, May 1995.

[155] Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

[156] Timothy. Tadros, Giri P. Krishnan, Ramayaa. Ramyaa, and Maxim Bazhenov. Sleep-like unsupervised replay reduces catastrophic forgetting in artificial neural networks. *Nat Commun*, 13(1):7742, Dec 2022.

[157] Md Yousuf Harun, Jhair Gallardo, Tyler L. Hayes, Ronald Kemker, and Christopher Kanan. SIESTA: Efficient online continual learning with sleep. *Transactions on Machine Learning Research*, 2023.

[158] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

[159] Jary Pomponi, Simone Scardapane, and Aurelio Uncini. Structured ensembles: An approach to reduce the memory footprint of ensemble methods. *Neural Networks*, 144:407–418, 2021.

[160] Magdalena J. Fosse, Roar Fosse, J. Allan Hobson, and Robert J. Stickgold. Dreaming and episodic memory: a functional dissociation? *J Cogn Neurosci*, 15(1):1–9, Jan 2003.

[161] Sue Llewellyn. Dream to Predict? REM Dreaming as Prospective Coding. *Front Psychol*, 6:1961, 2015.

[162] Sophie Schwartz. Are life episodes replayed during dreaming? *Trends Cogn Sci*, 7(8):325–327, Aug 2003.

[163] Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Lorenzo Bonicelli, Matteo Boschini, and Simone Calderara. Effects of auxiliary knowledge on continual learning. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1357–1363. IEEE, 2022.

[164] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[165] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017.

[166] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[167] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, and Timothee Masquelier. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.

[168] Giulio Tononi and Chiara Cirelli. Sleep and the price of plasticity: from synaptic and cellular homeostasis to memory consolidation and integration. *Neuron*, 81(1):12–34, 2014.

[169] Lisa Marshall and Jan Born. The contribution of sleep to hippocampus-dependent memory consolidation. *Trends in cognitive sciences*, 11(10):442–450, 2007.

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

[171] Timothy Tadros, Giri P Krishnan, Ramyaa Ramyaa, and Maxim Bazhenov. Sleep-like unsupervised replay reduces catastrophic forgetting in artificial neural networks. *Nature communications*, 13(1):7742, 2022.

[172] Amanda Rios and Laurent Itti. Closed-loop memory gan for continual learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3332–3338, 2019.

[173] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 226–227, 2020.

[174] Quentin Jodelet, Xin Liu, Yin Jun Phua, and Tsuyoshi Murata. Class-incremental learning using diffusion model for distillation and replay. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3425–3433, 2023.

[175] Zichong Meng, Jie Zhang, Changdi Yang, Zheng Zhan, Pu Zhao, and Yanzhi Wang. Diffclass: Diffusion-based class incremental learning. In *European Conference on Computer Vision*, pages 142–159. Springer, 2024.

[176] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018.

[177] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[178] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv e-prints*, 2022.

[179] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023.

[180] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

[181] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 2555–2563, 2023.

[182] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International conference on learning representations*, 2021.

[183] Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *Proceedings of the European Conference on Computer Vision*, pages 692–709. Springer, 2022.

[184] Oleksiy Ostapenko, Timothee Lesort, Pau Rodriguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on lifelong learning agents*, pages 60–91. PMLR, 2022.

[185] 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 *IEEE conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[186] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[187] Daniele Sportillo, Alexis Paljic, Mehdi Boukhris, Philippe Fuchs, Luciano Ojeda, and Vincent Roussarie. An immersive virtual reality system for semi-autonomous driving simulation: a comparison between realistic and 6-dof controller-based interaction. In *Proceedings of the 9th international conference on computer and automation engineering*, pages 6–10, 2017.

[188] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6238. IEEE, 2020.

[189] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *CVPR*, 2019.

[190] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation. In *CVPR*, 2020.

[191] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D pose tracker with anchor-based keypoints. In *ICRA*, 2020.

[192] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *CVPR*, 2019.

[193] Ruiqi Wang, Xinggang Wang, Te Li, Rong Yang, Minhong Wan, and Wenyu Liu. Query6DoF: Learning sparse queries as implicit shape prior for category-level 6DoF pose estimation. In *ICCV*, 2023.

[194] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowe Zhou. OnePose: One-shot object pose estimation without CAD models. In *CVPR*, 2022.

[195] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and Xiaowei Zhou. OnePose++: Keypoint-free one-shot object pose estimation without CAD models. *NeurIPS*, 2022.

[196] Yisheng He, Yao Wang, Haoqiang Fan, Jian Sun, and Qifeng Chen. FS6D: Few-shot 6D pose estimation of novel objects. In *CVPR*, 2022.

[197] Taeyeop Lee, Byeong-Uk Lee, Inkyu Shin, Jaesung Choe, Ukcheol Shin, In So Kweon, and Kuk-Jin Yoon. UDA-COPE: Unsupervised domain adaptation for category-level object pose estimation. In *CVPR*, 2022.

[198] Taeyeop Lee, Jonathan Tremblay, Valts Blukis, Bowen Wen, Byeong-Uk Lee, Inkyu Shin, Stan Birchfield, In So Kweon, and Kuk-Jin Yoon. TTA-COPE: Test-time adaptation for category-level object pose estimation. In *CVPR*, 2023.

[199] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.

[200] Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *NeurIPS*, 2021.

[201] David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *NeurIPS*, 2024.

[202] Hexiang Hu, Ozan Sener, Fei Sha, and Vladlen Koltun. Drinking from a firehose: Continual learning with web-scale natural language. *IEEE TPAMI*, 45(5):5684–5696, 2022.

[203] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011.

[204] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *Robotics: Science and Systems*, 2018.

[205] Jun Zhou, Kai Chen, Linlin Xu, Qi Dou, and Jing Qin. Deep fusion transformer network with weighted vector-wise keypoints voting for robust 6D object pose estimation. In *ICCV*, 2023.

[206] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *CVPR*, 2019.

[207] Jiehong Lin, Zewei Wei, Yabin Zhang, and Kui Jia. Vi-net: Boosting category-level 6D object pose estimation via learning decoupled rotations on the spherical representations. In *ICCV*, 2023.

[208] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[209] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE TPAMI*, 13(04):376–380, 1991.

[210] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[211] Long Tian, Changjae Oh, and Andrea Cavallaro. Test-time adaptation for 6D pose tracking. *Pattern Recognition*, page 110390, 2024.

[212] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[213] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[214] Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *arXiv preprint arXiv:2012.02780*, 2020.

[215] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš 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.

[216] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. FFB6D: A full flow bidirectional fusion network for 6D pose estimation. In *IEEE conference on Computer Vision and Pattern Recognition*, 2021.

[217] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *ICAR*, 2015.

[218] Suzanne C Segerstrom and Gregory E Miller. Psychological stress and the human immune system: a meta-analytic study of 30 years of inquiry. *Psychological bulletin*, 130(4):601, 2004.

[219] Adelinda Araújo Candeias, Edgar Galindo, Konrad Reschke, Mariola Bidzan, and Marcus Stueck. The interplay of stress, health, and well-being: unraveling the psychological and physiological processes, 2024.

[220] Herman M Van Praag. Can stress cause depression? *Progress in neuro-psychopharmacology and Biological Psychiatry*, 28(5):891–907, 2004.

[221] Amir Muaremi, Bert Arnrich, and Gerhard Tröster. Towards measuring stress with smartphones and wearable devices during workday and sleep. *BioNanoScience*, 3(2):172–183, 2013.

[222] Shruti Gedam and Sanchita Paul. A review on mental stress detection using wearable sensors and machine learning techniques. *IEEE Access*, 9:84045–84066, 2021.

[223] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T Campbell. Studentlife: assessing mental

health, academic performance and behavioral trends of college students using smartphones. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing*, pages 3–14, 2014.

[224] Soowon Kang, Woohyeok Choi, Cheul Young Park, Narae Cha, Auk Kim, Ahsan Habib Khandoker, Leontios Hadjileontiadis, Heepyung Kim, Yong Jeong, and Uichin Lee. K-emophone: A mobile and wearable dataset with in-situ emotion, stress, and attention labels. *Scientific data*, 10(1):351, 2023.

[225] Seyed Amir Hossein Aqajari, Sina Labbaf, Phuc Hoang Tran, Brenda Nguyen, Milad Asgari Mehrabadi, Marco Levorato, Nikil Dutt, and Amir M Rahmani. Context-aware stress monitoring using wearable and mobile technologies in everyday settings. *arXiv preprint arXiv:2401.05367*, 2023.

[226] Alex W DaSilva, Jeremy F Huckins, Rui Wang, Weichen Wang, Dylan D Wagner, and Andrew T Campbell. Correlates of stress in the college environment uncovered by the application of penalized generalized estimating equations to mobile sensing data. *JMIR mHealth and uHealth*, 7(3):e12084, 2019.

[227] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[228] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE, 2005.

[229] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[230] Yekta Said Can, Bert Arnrich, and Cem Ersoy. Stress detection in daily life scenarios using smart phones and wearable sensors: A survey. *Journal of biomedical informatics*, 92:103139, 2019.

[231] Irene Bonafonte, Cristina Bustos, Abraham Larrazolo, Gilberto Lorenzo Martínez Luna, Adolfo Guzmán Arenas, Xavier Baró, Isaac Tourgeman, Mercedes Balcells, and Agata Lapedriza. Analyzing the contribution of different passively collected data to

predict stress and depression. In *2023 11th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 1–4. IEEE, 2023.

[232] Philip Schmidt, Attila Reiss, Robert Duerichen, Claus Marberger, and Kristof Van Laerhoven. Introducing wesad, a multimodal dataset for wearable stress and affect detection. In *Proceedings of the 20th ACM international conference on multimodal interaction*, pages 400–408, 2018.

[233] Javier Hernandez, Dan Morris, and Rosalind W Picard. Using physiological, behavioral, and self-report data to infer affect in the workplace. *IEEE Transactions on Affective Computing*, 2018.

[234] Stephen M Mattingly, Julie M Gregg, Pino Audia, Ayse Elvan Bayraktaroglu, Andrew T Campbell, Nitesh V Chawla, Vedant Das Swain, Munmun De Choudhury, Sidney K D'Mello, Anind K Dey, et al. The tesserae project: Large-scale, longitudinal, in situ, multimodal sensing of information workers. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–8, 2019.

[235] Evgenia Lazarou and Themis P Exarchos. Predicting stress levels using physiological data: Real-time stress prediction models utilizing wearable devices. *AIMS neuroscience*, 11(2):76, 2024.

[236] Abhinav Shaw, Natcha Simsiri, Iman Deznaby, Madalina Fiterau, and Tauhidur Rahaman. Personalized student stress prediction with deep multitask network. *arXiv preprint arXiv:1906.11356*, 2019.

[237] Yunfei Luo, Iman Deznabi, Abhinav Shaw, Natcha Simsiri, Tauhidur Rahman, and Madalina Fiterau. Dynamic clustering via branched deep learning enhances personalization of stress prediction from mobile sensor data. *Scientific Reports*, 14(1):6631, 2024.

[238] 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.

[239] Camila González, Amin Ranem, Daniel Pinto dos Santos, Ahmed Othman, and Anirban Mukhopadhyay. Lifelong nnu-net: a framework for standardized medical continual learning. *Scientific Reports*, 13(1):9381, 2023.

[240] Dani Kiyasseh, Tingting Zhu, and David A Clifton. Clops: Continual learning of physiological signals. *arXiv preprint arXiv:2004.09578*, 2020.

[241] Adnan Ahmad, Bahareh Nakisa, and Mohammad Naim Rastgoo. Robust emotion recognition via bi-level self-supervised continual learning. *arXiv preprint arXiv:2505.10575*, 2025.

[242] Tianyi Zhang, Abdallah El Ali, Alan Hanjalic, and Pablo Cesar. Few-shot learning for fine-grained emotion recognition using physiological signals. *IEEE Transactions on Multimedia*, 25:3773–3787, 2022.

[243] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 7871. Association for Computational Linguistics, 2020.