Incremental Learning Methodologies for Addressing Catastrophic Forgetting: Analysis and Experimental Evaluation

MIQUEL SERRA-PERELLO, Dep. Mathematics and Computer Science & Artificial Intelligence Research Institute (IAIB), University of the Balearic Islands, Spain
ALBERTO ORTIZ*, Dep. Mathematics and Computer Science & Artificial Intelligence Research Institute (IAIB), University of the Balearic Islands, Spain

Artificial neural networks have been reported to exhibit, and in some cases surpass, human level performance on individual rigid tasks. However, these networks remain static entities of knowledge for those specific tasks, which can lead to catastrophic forgetting —i.e., forgetting old tasks— when attempting to learn new tasks. The main objective of Incremental Learning (IL) is to address this issue. In order to tackle catastrophic forgetting, various approaches have been proposed so far. We survey those approaches and organize them in nine categories: regularization-based methods, exemplar replay-based methods, variational continual learning-based methods, parameter isolation-based methods, dynamic architectures-based methods, distillation-based methods, generative methods, data-free methods and unsupervised methods. Moreover, this review distinguishes between two scenarios, Task Incremental Learning (Task-IL) and Class Incremental Learning (Class-IL), and reports on the results obtained for a number of experiments that compare the performance achieved by a selection of diverse methods for both scenarios on the datasets most used by the related research community.

JAIR Associate Editor: Bo Han

JAIR Reference Format:
Miquel Serra-Perello and Alberto Ortiz. 2025. Incremental Learning Methodologies for Addressing Catastrophic Forgetting: Analysis and Experimental Evaluation. *Journal of Artificial Intelligence Research* 83, Article 39 (August 2025), 39 pages. DOI: 10.1613/jair.1.18405

1 Introduction

In recent years, machine learning models have been reported to exhibit or even surpass human level performance on individual tasks, such as e.g. in object recognition [76]. While these results are impressive, they are obtained with static models incapable of adapting their behavior over time. In real contexts connected to robot applications requiring high levels of autonomy, the static assumption is quickly violated, giving rise to unstable behaviour as soon as new categories are not correctly handled and the robot has to make decisions on the basis of the perceptual information received. Clearly, this can only be counteracted by tightly controlling the environment of operation.

If the perception model, e.g. an artificial neural network, tries to learn different tasks one after the other, in an attempt to counteract the aforementioned by means of an incremental approach that incorporates new knowledge as new data becomes available, it can suffer from *catastrophic forgetting* of old tasks as the new tasks are learned.

*Corresponding Author.

Authors' Contact Information: Miquel Serra-Perello, ORCID: 0000-0003-2326-970X, miquel.serra@uib.es, Dep. Mathematics and Computer Science & Artificial Intelligence Research Institute (IAIB), University of the Balearic Islands, Palma de Mallorca, Balearic Islands, Spain; Alberto Ortiz, ORCID: 0000-0002-8253-7455, alberto.ortiz@uib.es, Dep. Mathematics and Computer Science & Artificial Intelligence Research Institute (IAIB), University of the Balearic Islands, Palma de Mallorca, Balearic Islands, Spain.

DOI: 10.1613/jair.1.18405

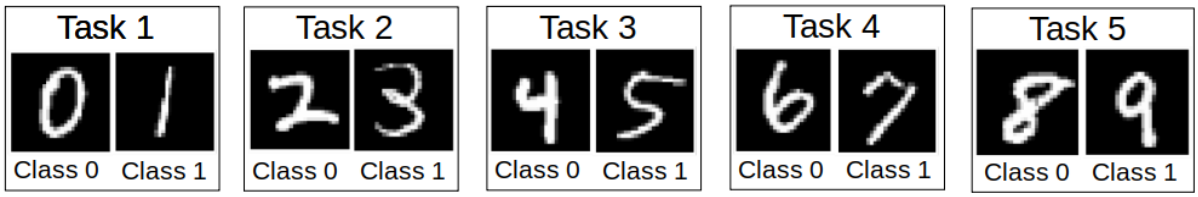

Fig. 1. Task division of the MNIST dataset.

Addressing catastrophic forgetting requires striking a delicate balance between preserving past knowledge and accommodating new information. To overcome this challenge, a wide range of algorithms and methodologies have been proposed.

Different names have been used to refer to the study of a solution to catastrophic forgetting, such as continual learning [80], lifelong learning [65] or incremental learning [13]. In this survey, we will use the term incremental learning (IL) and will focus on neural network models.

IL encompasses two types of inference configurations: Task-IL and Class-IL. In Task-IL (or multi-head configuration), each task is associated with a dedicated head (classification layer) in the network, so when the task is known, the model can classify between the classes within a specific task. However, when the task ID is not available (Class-IL or single-head configuration), the model has a shared output layer and needs to classify among the classes for all tasks.

Specifically, in this work, we investigate state of the art methods from the Task-IL and Class-IL scenarios, which, employing the terminology adopted by the research community, include approaches based on regularization, exemplar replay, variational continual learning, parameter isolation, dynamic neural network architectures, distillation, generative and data-free methodologies, and unsupervised learning. The survey comprises a detailed compilation of the different IL methods published so far, with expanded descriptions for each. Moreover, paying particular attention to the fact that most of the published methodologies actually use some sort of combination of the *pure categories* enumerated above, we analyze them and specifically indicate the respective combinations adopted. Finally, we report the results of a comparison comprising a selection of those methods with the aim of providing more insight into the respective strengths and weaknesses.

This paper is organized as follows: in section 2, we describe the different incremental learning scenarios that are typically considered; in section 3, we outline the different approaches involved in achieving incremental learning, while the specific algorithms are detailed in section 4; the experiments performed for a selection of methods and their results are reported and discussed in section 5; lastly, we finish with a summary of conclusions in section 6.

2 Incremental Learning Scenarios

While considering the incremental learning problem, in which a single model needs to sequentially learn a series of tasks, we find that an important difference between the experimental protocols for incremental learning is whether at test time information about the task identity is available or not. Yet this consideration is not always clearly stated and differences in this regard are sometimes not appreciated. To enable fairer and more meaningful comparisons, we present three distinct scenarios for incremental learning of increasing difficulty as identified in [88].

In the first scenario, models are always informed of the task to be performed. This is the easiest continual learning scenario, and we refer to it as *task-incremental learning* (Task-IL). Since task identity is always provided, it is possible to train models with task-specific components. A typical neural network architecture used in this

scenario has a *multi-headed* output-layer, meaning that each task has its own output units but the rest of the network is (potentially) shared between tasks. For example, in the task division shown in Figure 1, the model should classify between class 0 or class 1 from the task at hand.

In the second scenario, which we refer to as *domain-incremental learning* (Domain-IL), the task identity is not available at test time. Models however only need to solve the task at hand, i.e. predict the class; they are not required to infer which task it is. Typical examples of this scenario are protocols whereby the structure of the tasks is always the same, but the input-distribution is changing. A relevant real-world example is an agent who needs to learn to survive in different environments, without the need to explicitly identify the environment it is confronted with. For example, in the task division shown in Figure 1, the model distinguishes between the first or the second class, i.e. the sample belongs to classes [0, 2, 4, 6, 8] or to classes [1, 3, 5, 7, 9].

Finally, in the third scenario, models need to be able to both solve each task seen so far and infer which task they are presented with. We refer to this last scenario as *class-incremental learning* (Class-IL), as it comprises methodologies in which a model needs to incrementally learn to recognize new classes. For example, in the task division shown in Figure 1, the model is expected to be able to classify input samples between all digits, from 0 to 9, once all tasks have supposedly been accomplished.

Since few works fall in the second scenario, and the majority of the research to this date falls either in the first scenario (Task-IL) or in the third scenario (Class-IL), in this survey we will focus on the Task-IL and Class-IL scenarios.

3 Incremental Learning Approaches

In this section we describe the characteristics of the different incremental learning approaches. In Table 1, we enumerate all the methods that are discussed, indicating which of the approaches are used by the different methods.

3.1 Regularization

This line of work introduces an extra regularization term in the loss function, which aims at retaining the knowledge gained from previous tasks while adapting to new ones. This method can be further divided into data-focused and prior-focused methods:

(1) *Data-focused* methods use previous task model outputs to improve the training of the model in the new data [81]. For instance, LwF [53] uses the previous model output as soft labels for previous tasks. Other works, LFL [42] and DMC [101], introduced related ideas; however, it has been shown that this strategy is vulnerable to domain shift between tasks [6].
(2) *Prior-focused* methods estimate a distribution over the model parameters, which is used as prior when learning from new data. The importance of all neural network parameters is estimated assuming they are independent to ensure feasibility. Changes to important parameters are penalized during training of later tasks. EWC [44] was the first to establish this approach. Other works are MAS [4], SI [98] and IMM [51].

3.2 Exemplar Replay

This approach is intended to store selected samples from previous tasks in a specific memory, most times of limited capacity. To alleviate forgetting, the stored samples are replayed while learning a new task to refresh the previous tasks, e.g. they are involved in posterior trainings. They can be either used as model inputs for rehearsal, or to constrain the optimization of the new task loss to prevent previous task interference.

Some of the most notable methods in this approach are iCaRL [74], GEM [57], A-GEM [15] and BiC [92]. We can comment on them regarding the types of memory that are used and the sampling strategy:

(1) Regarding memories, there are two types of memories. If the memory is allowed to grow, a number of samples is stored for each class. If the memory has a fixed size, the number of samples for each class depends on the number of classes seen at that time and the maximum size of the memory.
(2) As for sampling strategies, we can consider different strategies when selecting the samples. The simplest way is to select the samples randomly. Another strategy used in iCaRL [74] selects exemplars based on which one causes the average feature vector over all exemplars to best approximate the average feature vector over all training examples, what is known as *herding*. In RWalk [14] two other sampling strategies are proposed. The first one calculates the entropy of the softmax outputs and selects exemplars with higher entropy. The second one selects exemplars based on how close they are to the decision boundary. The order in which exemplars are chosen follows a decreasing order of importance. MIR [5], ER [17], SER [39] and CoPE [21] apply *reservoir sampling*, which ensures every sample has the same probability to be stored in the memory.

3.3 Variational Continual Learning

This family is based on the Bayesian inference framework. These methods typically employ Bayesian neural networks and variational inference to maintain a distribution over model parameters instead of a single set of parameters. This distribution captures uncertainty and allows for flexible updates to the model as new data arrives.

Some examples of methods using this approach are VCL [63], VGR [26], DGR [80], UCL [2], FBCL [18], IUVCL [83], CLAW [1], UCB [24], BGD [99] and FOO-VB [100].

3.4 Parameter Isolation

The parameter isolation approach dedicates different model parameters to each task. If there are no constraints on the architecture size, PNN [77] and RCL [93] grow new branches in the network for new tasks, while freezing previous task parameters. Other works keep the architecture static and mask out previous task parts during training of new tasks; this is the case of PathNet [27], PackNet [59] and HAT [79]. These methods fall in the Task-IL scenario, since they need to know the branch or mask corresponding to the task being predicted at the moment.

3.5 Dynamic Architectures

The dynamic architecture-based approach constitute another method inside the parameter isolation family, where the neural network architecture adapts and modifies as the model encounters new tasks or data. The key idea is to grow the network capacity incrementally and selectively to meet the demands of new tasks while preserving past knowledge. Some of these methods also fall in the Task-IL scenario, since they become impractical when the task-ID is not known.

Some examples of methods following this approach are DER [95], Expert Gate [6], DAN [75], RCL [93], PNN [77], DEN [97], RPS [69], P&C [78], ACL [25], Growing a Brain [90], PROPRE [30], NGPCA [3], DYNG [11], TOPIC [84], ILVQ [94], FOSTER [91], AR1 [60] and AdaNet [20].

3.6 Distillation

The distillation methods *transfer knowledge* from a previously trained model (often referred to as the teacher model) to a new or smaller model (the student model) in the context of incremental learning scenarios [37]. The knowledge transfer is done by training the student model to mimic the behavior of the teacher model. Instead of using hard target labels, which the teacher model used, the student model is trained using soft target labels, which are the probability distributions over classes produced by the teacher model for a given input.

Table 1. Qualitative comparison among the different methods surveyed.

	RM	ER	VA	PI	DA	DI	GM/DF	US
iCaRL [74], BiC [92]		✓				✓		
LwF [53], DMC [101]	✓					✓		
EWC [44], R-EWC [54], SI [98], MAS [4], LFL [42], IMM [51], RWalk [14], EBLL [72], CAB [36]	✓							
GEM [57], A-GEM [15]	✓	✓						
LwM [22], GD [50]						✓		
DER [95], ACL [25]		✓			✓			
MERLIN [41], ITAML [70], GDumb [67]		✓						
MIR [5], GSS [7], ER [17], SER [39], TEM [16], CoPE [21]		✓						
EEIL [13], IL2M [10], LUCIR [38]		✓				✓		
VCL [63], FBCL [18], IUVCL [83]	✓	✓	✓					
DGR [80], PR [8], LGM [71], CCLUGM [48], FeTrIL [66]							GM	
MeRGAN [19]		✓					GM	
VGR [26]			✓				GM	
DGM [64]					✓		GM	
PackNet [59], PathNet [27], HAT [79], Piggyback [58]				✓				
Growing a Brain [90], PROPRE [30], TOPIC [84], ILVQ [94]					✓			
Expert Gate [6], CN-DPM [52], DAN [75], RCL [93], PNN [77], DEN [97]				✓	✓			
AR1 [60], AdaNet [20]	✓				✓			
FOSTER [91]					✓	✓		
RPS [69]		✓			✓	✓		
P&C [78]	✓					✓		
UCL [2], CLAW [1], UCB [24], BGD [99], FOO-VB [100]			✓					
ARM [40], CBCL-PR [9]							DF	
ABD [82], DI, ADI [96], RTF [87], BI-R [85] [86]						✓	DF	
SOINN [28] ESOINN [29] RE-SOINN [68], CURL [73], CF [32], AutoNovel [33, 34]								✓
iLAP [43]		✓						✓

RM: Regularization-based method; ER: Exemplar replay-based method (with memory); VA: Variational continual learning-based method; PI: Parameter isolation-based method; DA: Dynamic architecture-based method; DI: Distillation-based method; GM/DF: Generative or data-free method; US: Unsupervised method.

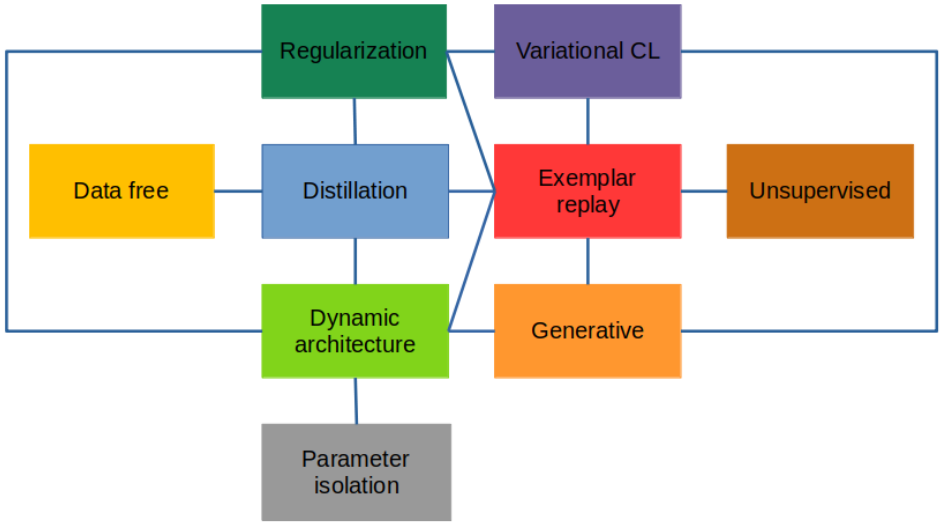

Fig. 2. Interrelationships between the different categories identified in the related literature.

Some examples of methods using this approach are iCaRL [74], LwF [53], BiC [92], LwM [22], DMC [101], GD [50], EEIL [13], IL2M [10], LUCIR [38], DI, ADI [96], DD [12], DFKD [56], ABD [82], RTF [87] and BI-R [85, 86].

3.7 Generative/Data-free Methods

Within this category, we differentiate between generative methods and data-free methods.

The first group uses an extra network to generate additional training data or samples to aid in the learning process, particularly when adapting to new tasks or domains. On the other side, the main idea behind generative methods is to reinforce learning on past tasks or to simulate data from new tasks.

The second group also generates additional samples, but in this case there is no extra network for generating the additional samples. Instead, the samples are generated using the feedback connections, the implicit memory within the model or inverting the inference network itself.

Some examples of methods using these approaches are DGR [80], PR [8], CCLUGM [48], LGM [71], DGM [64], ARM [40], DI, ADI [96], DD [12], DFKD [56], ABD [82], RTF [87], FeTrIL [66], CBCL-PR [9] and BI-R [85, 86].

3.8 Unsupervised Methods

Unsupervised approaches in incremental learning refer to techniques and methods that aim to learn from data without explicit supervision, labeled information or task boundaries, similarly to generic machine unsupervised learning methods.

Some examples of methods using this approach are: SOINN [28], ESOINN [29], RE-SOINN [68], CURL [73], CF [32], iLAP [43] and AutoNovel [33] [34].

3.9 Summary

Figures 2 and 3 illustrate with visual representations the combination of the different approaches described in this section. Figure 2 connects each approach with the other approaches with which it has been combined in at least one method. Figure 3 classifies all the methods discussed in accordance to the approaches they use.

4 Review of IL Methodologies

In this section, we describe in more detail the methods that have been enumerated above. As already commented, a fast summary can be found in Table 1, as a complement of the information contained in Figures 2 and 3. Below, we examine all methods following the same order and groupings as of Table 1.

iCaRL [74] and BiC [92] are exemplar replay and distillation-based methods. iCaRL learns classifiers and a feature representation simultaneously from a data stream in Class-IL. iCaRL relies on sets of exemplar images that it selects dynamically out of the data stream using the herding strategy. An exemplar set is stored for each observed class so far. The total number of exemplar images never exceeds a fixed parameter K (fixed-size memory). To ensure this, the number of exemplars in each set is reduced each time a new class gets introduced to accommodate the exemplars of the new class. BiC trains an additional linear layer to remove bias with a separate validation set. This is done in two stages. First, the convolutional layers and the fully connected layer are trained. In the second stage, both convolutional and fully connected layers are frozen, and the additional linear layer is trained by using a small validation set.

LwF [53] and DMC [101] are regularization and distillation-based methods. LwF has shared parameters and task-specific parameters, its goal is to add task-specific parameters for a new task and to learn parameters that work well on old and new tasks, using images from only the new task. LwF tries to keep the representation of previous data from drifting too much while learning new tasks using knowledge distillation. The teacher model is the model after learning the last task, and the student model is the model trained with the current task. DMC performs IL in two steps: the first step is to train a new classifier for the new classes, which is trained by standard back-propagation; the second step is to consolidate the old model and the new model via double distillation by exploiting publicly available unlabeled auxiliary data.

Some approaches only using regularization are EWC [44], R-EWC [54], SI [98], MAS [4], LFL [42], IMM [51], RWalk [14], EBLL [72] and CAB [36]. EWC, R-EWC, SI and MAS estimate the importance of the parameters of the network for solving past tasks. Novel tasks can then be learned by penalizing changes to the most important parameters, preventing important knowledge related to previous tasks from being overwritten. LFL freezes the weights of the softmax layer to maintain the previous knowledge. Then the network is trained to simultaneously minimize the total loss function. IMM approximates the posterior distribution of model parameters using Gaussian distributions and leverages techniques like mean-IMM and mode-IMM to match moments and facilitate learning tasks sequentially. Additionally, transfer techniques like weight-transfer, L2-transfer, and drop-transfer are used to improve the learning process. RWalk is a generalization of an efficient version of EWC [44] and the SI [98] path integral with a theoretically grounded KL-divergence-based perspective. They also introduce two metrics to quantify forgetting and intransigence. EBLL learns an under-complete autoencoder for each task, capturing the features that are crucial. During training, an additional regularization term prevents the reconstruction of the features from these autoencoders from changing. At the same time, the features are given space to adjust to the most recent environment. CAB is a variant of the back-propagation algorithm that uses conceptors to shield gradients against the degradation of previously learned tasks.

GEM [57] and A-GEM [15] are regularization and exemplar replay methods that prevent forgetting by constraining the parameters update with samples in a memory buffer. Both ensure that the loss for past tasks does not increase.

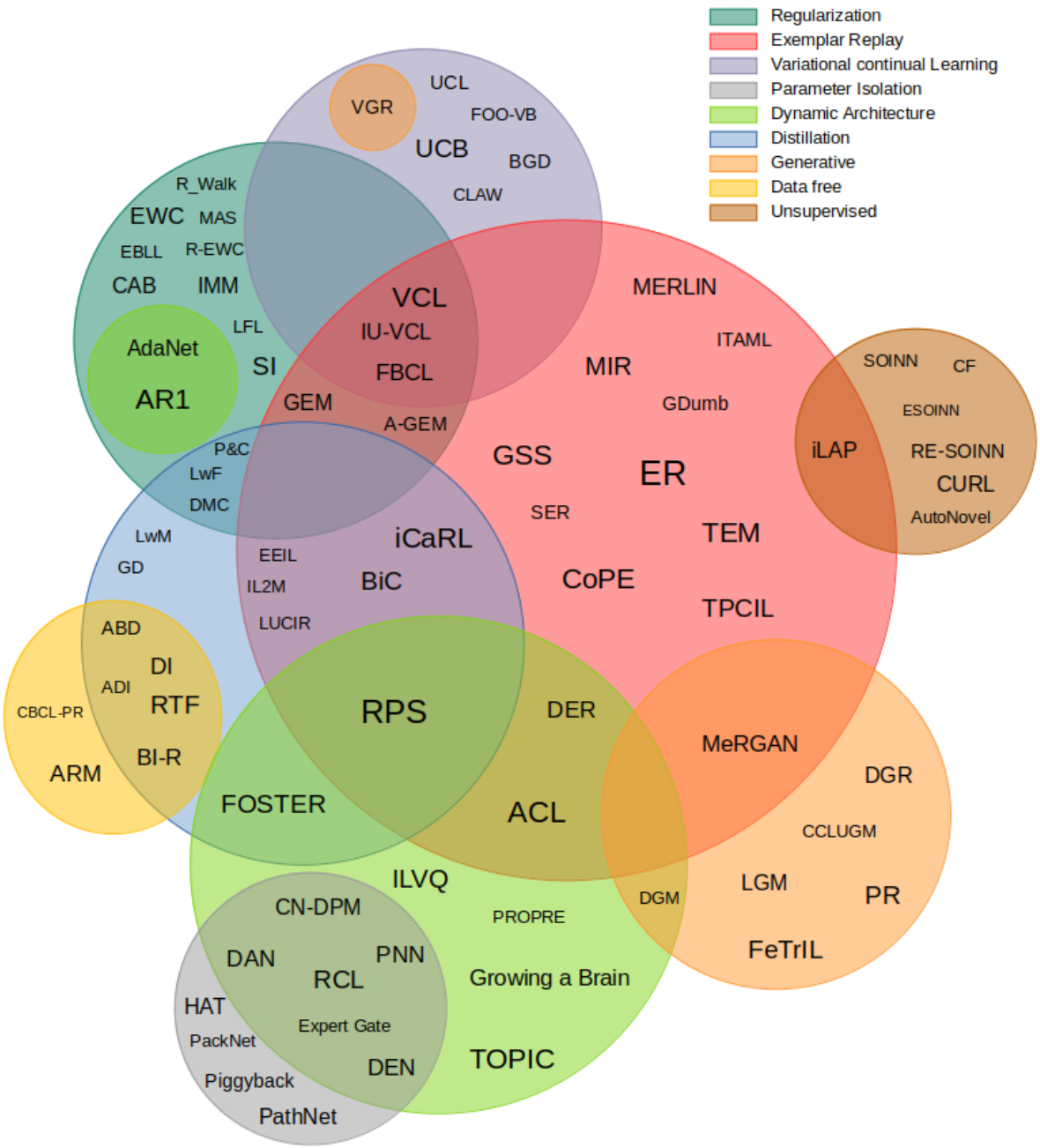

Fig. 3. Graphical representation of the combinations between the different approaches. Font sizes obey aesthetic purposes. Best viewed in color.

LwM [22] and GD [50] are distillation-based approaches that do not need memory for past classes. LwM proposes an information-preserving penalty using an attention distillation loss that captures the changes in the classifier attention maps in order to preserve past knowledge. GD first trains a teacher on the current task. Then, a model is trained on the current task by triple-distillation from the teacher, the previous model and their ensemble. Finally, they perform fine-tuning on the task-specific parameters with data weighting.

DER [95] and ACL [25] are dynamic architectures that use exemplar replay. DER employs a memory-based rehearsal strategy and divides the learning process into two stages: representation learning and classifier learning. During representation learning, a super-feature extractor expands the existing representation with new features to maintain a balance between stability and adaptability. Catastrophic forgetting is mitigated by freezing the previous representation and allowing flexibility for the new parameters. Dynamic expansion is achieved using differentiable channel-level masks, and sparsity is encouraged through a sparsity loss. The second stage focuses on reducing the weights bias by means of retraining with a balanced fine-tuning method. ACL learns a disjoint representation for the task-invariant and task-specific features required to solve a sequence of tasks. This allows to learn shared features that are more robust to forgetting.

Some methods based only on exemplar replay are MERLIN [41], ITAML [70] and GDumb [67]. MERLIN assumes that the weights of a neural network for solving a task come from a meta-distribution which is learned and consolidated incrementally. A set of base models is trained on random subsets of the training samples an then it is used to learn a task-specific parameter distribution using a a Variational Auto Encoder (VAE)-like strategy. This is followed by a meta-consolidation phase, where model parameters for all tasks seen so far are sampled from the decoder of the VAE. Models from the parameter distributions for each task are sampled and used to evaluate on the test data. MERLIN works in both Class-IL and Task-IL scenarios. ITAML learns a set of generalized parameters, that are neither specific to old nor to new tasks, with a novel meta-learning approach that seeks to maintain an equilibrium between all the encountered tasks. This proposed meta-learning approach involves two updates, an *inner loop* which generates task-specific models for each task, and an *outer loop*, which combines task-specific models into a final generic model. GDumb is a baseline that ensures a balanced training set. It greedily stores samples in memory in a class-balanced way as they arrive. For testing, a model is trained from scratch using only samples in the memory.

MIR [5], GSS [7], ER [17], SER [39], TEM [16] and CoPE [21] are other replay-based methods that focus on the memory-update strategy. MIR, ER, SER and CoPE apply reservoir sampling, which ensures every sample has the same probability to be stored in the memory. On the other hand, GSS tries to diversify the gradient space between the samples in the memory. Finally TEM populates the memory with a good trade-off between randomizing the examples in the memory and keeping enough samples for each class.

EEIL [13], IL2M [10] and LUCIR [38] add distillation to the exemplar replay strategy. EEIL uses new data and a small exemplar set corresponding to samples from the old classes with fixed-size memory and a herding-based selection strategy. This method is based on a loss composed of a distillation measure to retain the knowledge acquired from the old classes, and a cross-entropy loss to learn the new classes. They keep the entire framework end-to-end, i.e., learning the data representation and the classifier jointly. IL2M uses fine-tuning to update deep models for each incremental state. It has a first memory that stores exemplar images of past classes, and a second memory that stores past class statistics obtained when they were initially learned. Since fine-tuning generates a prediction bias in favor of new classes, IL2M uses the contents of its memories to rectify the scores of past classes and make them more comparable to those of the new classes in order to correct the prediction bias. LUCIR is based on fine-tuning with an integrated objective function. Authors propose the following contributions: 1) cosine normalization to balance magnitudes of past and new classifiers, 2) a less-forget constraint to preserve the geometry of past classes, and 3) inter-class separation to maximize the distances between past an new classes.

VCL [63], FBCL [18] and IUVCL [83] are methods that add variational continual learning to the regularization and exemplar replay strategies. VCL is a simple but general framework for incremental learning that merges online

and Monte Carlo variational inference for neural networks. It can successfully train both deep discriminative and deep generative models in incremental learning settings where existing tasks evolve over time and entirely new tasks emerge. FBCL uses natural gradients and Stein gradients together with coresets (small collections of samples of every learned task) to better estimate posterior distributions over the parameters, which can be used to select the coreset samples. IUVCL improves on VCL by establishing a new best practice approach to mean-field variational Bayesian neural networks. They also test a version using coresets.

Among the generative strategies, DGR [80], PR [8], LGM [71], CCLUGM [48] and FeTrIL [66] generate synthetic samples using an additional network. In DGR, an auxiliary classifier is used to assign ground truth labels to all the samples generated with an unconditional generative adversarial network (GAN). PR applies pseudo-rehearsal to both a classification model and a GAN model. LGM has a two-model Variational Autoencoder architecture which allows it to learn and preserve all the distributions seen so far, without the need to retain the past data nor the past models. CCLUGM builds on LGM and extends it to include a classifier that remembers all past classification tasks. FeTrIL introduces a method which combines a fixed feature extractor and a pseudo-features generator to improve the stability-plasticity balance. Features of new classes and pseudo-features of past classes are fed into a linear classifier which is trained incrementally to discriminate between all classes.

Additionally within this group, MeRGAN [19] improves on DGR by using a conditional GAN and exemplar replay alignment.

Lastly, VGR [26] is a variational inference generalization of DGR which falls under likelihood-focused approaches. Likelihood-focused methods are intended to express well-calibrated uncertainty about whether data have been seen before.

DGM [64] makes use of the advantages of conditional GANs combined with synaptic plasticity using neural masking with two variants applied to 1) layer activations and 2) to connection weights. Further, a dynamic network expansion mechanism that ensures sufficient model capacity to accommodate for incoming tasks is introduced.

Within the parameter-isolation group of strategies, PackNet [59], PathNet [27], HAT [79] and Piggyback [58] fall into the task-IL scenario. PackNet exploits redundancies in large deep networks to free up parameters that can then be employed to learn new tasks. This is achieved in two training phases. First, the network is trained on a new task with parameters subsets from previous tasks frozen. Then a portion of the weights of the network is pruned away, selecting by lowest magnitude. In the second stage, the network is retrained for a small number of epochs. The weights that did not get pruned get frozen and training on the next task begins. PathNet learns paths, i.e. subsets of parameters, using evolutionary strategies that are then frozen and used for that specific task. Then a new path is learned for the new task. HAT is a hard attention mask mechanism that only requires one training phase. A mask is learned concurrently for every task, through Stochastic Gradient Descent, and previous masks are preserved and exploited to condition such learning. Piggyback adapts a fixed backbone to multiple tasks by learning binary masks on network weights appropriate for the task at hand.

Some dynamic architectures-based methods that can be found are Growing a Brain [90], PROPRE [30], TOPIC [84] and ILVQ [94]. Growing a Brain widens existing layers or deepens the overall network. The newly-added units must be appropriately normalized to allow for a pace of learning that is consistent with existing units. PROPRE is based on NG [61] and SOMs [45], which implements an extra supervised readout layer, as well as a concept drift detection mechanism in order to make the SOM usable in an IL context. TOPIC adapts NG [61] to few-shot class incremental learning. Firstly, the NG network learns feature space topologies for knowledge representation and the network grows to learn new classes while also dealing with changes in the feature space due to deep model update. Secondly, TOPIC preserves past knowledge by stabilizing the topology of the NG network using an Anchor Loss term. ILVQ applies a dynamic threshold-based insertion criterion to grow the network. The number of prototypes for each class can be generated automatically and adjusted adaptively during learning. A deletion policy is taken to eliminate the possible noise that comes into the network during learning.

Combining the two last strategies that have been mentioned, parameter-isolation and dynamic architectures, one can find Expert Gate [6], CN-DPM [52], DAN [75], RCL [93], PNN [77] and DEN [97]. Expert Gate is based on a Network of Experts. A new expert model is added for each new task. A set of gating autoencoders learn the representations for the tasks, and, at test time, forward the test sample to the relevant expert model. CD-DPM consists of a set of neural network experts that are in charge of a subset of the data. During training, each sample is evaluated by every expert. If none of the existing experts is responsible for that sample, the data is stored in a short-term memory. When this memory is full, a new expert is created from the data stored in the short-term memory. Otherwise, if an expert is responsible, it is learned by the corresponding expert. On the other side, DAN adapts a pretrained CNN for a new task by introducing controller modules to each layer. These modules create new convolutional filters specific to the new task, enabling the network to perform multiple tasks efficiently. RCL searches for the best neural architecture for each coming task by reinforcement learning. RCL consists of three networks: the controller, trained with reinforcement learning, decides the number of filters or nodes to add for each new task; the value network that approximates the state value; and the task network that adapts itself to prevent forgetting. Contrarily, PNN adds a new neural network for each task and adds lateral connections to features of previously learned networks, allowing them to reuse, modify or ignore the previously learned features. Lastly, DEN performs selective retraining, dynamically expands the network capacity upon arrival of each task with only the necessary number of units, and prevents semantic drift by splitting/duplicating units and timestamping them.

AR1 [60] and AdaNet [20] are methods based on regularization and dynamic architectures. AR1 intermediate weights are adapted without negative impact in terms of forgetting by applying SI regularization [98]. For each batch, the output layers are extended with new neurons/weights for the new classes, and are initialized to zero. AdaNet simultaneously learns a neural network architecture and its parameters by balancing a trade-off between model complexity and empirical risk minimization.

FOSTER [91] is a dynamic architecture and distillation-based method. FOSTER first expands dynamically new modules to fit the residuals between the target and the output of the original model, and introduces logits alignment to alleviate the classification bias and feature enhancement to balance the representation learning of the old and new classes. Then, redundant parameters and feature dimension are removed through distillation to maintain the single backbone model.

Adding rehearsal-based replay to the last method, one can find RPS [69], which progressively chooses optimal paths for the new tasks while encouraging parameter sharing. It uses a simple controller to dynamically balance the model plasticity in order to maintain an equilibrium between previous and newly acquired knowledge.

P&C [78] is a regularization and distillation method that has two different components. A knowledge base, capable of solving previously encountered problems, is connected to the second component, an active column that is employed to learn the current task. After learning a new task, the active column is distilled into the knowledge base, taking care to protect any previously acquired skills with EWC [44].

The methods based only on variational continual learning that have been found are UCL [2], CLAW [1], UCB [24], BGD [99] and FOO-VB [100]. UCL gives an interpretation of the Kullback-Liebler divergence term of the variational lower bound for Gaussian mean-field approximation. UCL proposes the notion of node-wise uncertainty to improve the per-weight regularization. Moreover, UCL adds two regularization terms that enforce stability by freezing important parameters and allow plasticity by controlling the learning parameters of a new task. CLAW extends VCL [63] by applying an attention mechanism, based on probabilistic modeling, which adaptively identifies which parts of the network to share in a data-driven way. UCB defines an uncertainty value for each weight, identifying which are the weights that should change and which should stay fixed. It also uses uncertainty for weight pruning and retains task performance after pruning by saving binary masks for each task. BGD uses a diagonal Gaussian distribution and the *labels trick*, which consists in training only the heads that correspond to labels that exist in the current batch, instead of training all heads. FOO-VB derives novel

fixed-point equations for the online variational Bayes optimization problem. It updates the posterior parameters using the prior distribution estimated and the update rules derived from the fixed-point equations.

Among the data-free methods, we can refer to ARM [40] and CBCL-PR [9]. Instead of relying on memory or generator networks, ARM utilizes the implicit memory within the model itself to generate specialized auxiliary samples on the fly. CBCL-PR classifies the different classes in the form of sets of clusters and generates pseudo-exemplars from those clusters.

ABD [82], DI/ADI [96], RTF [87] and BI-R [85]-[86] add distillation to the data-free approach. ABD proposes a novel incremental distillation strategy contributing with a modified cross-entropy training and importance-weighted feature distillation. It also explores the concept of model-inversion image synthesis, where the trained inference network can be inverted to obtain images with activations in the network similar to the training data. DI uses the running average statistics stored in the BatchNorm layers to generate images. It consists of two steps: 1) model inversion, and 2) application-specific knowledge distillation. The same authors also propose ADI, an enhanced image generation scheme that encourages the synthesized images to cause student-teacher disagreement, introducing an additional loss that penalizes output distribution similarities. RTF adapts the main model to generate replay data by adding: 1) feedback connections that are trained to reconstruct inputs from their hidden representations and 2) a layer of stochastic latent variables that are trained to follow a known distribution from which it is easy to sample. BI-R uses context-modulated feedback connections to generate internal or hidden representations to be replayed. The replays are network embeddings rather than pixel-level images.

Finally, in the unsupervised incremental strategy, one can find SOINN [28], ESOINN [29], Re-SOINN [68], CURL [73], CF [32] and AutoNovel [33] [34]. SOINN consists of two layers. The first layer learns the density distribution of the input data. The second layer separates clusters by detecting low-density areas in the input data. When learning is finished, SOINN reports the number of clusters and gives typical prototype nodes for every cluster. When a new node is inserted, it is evaluated to ensure that this insertion decreases the error. ESOINN improves SOINN in the sense that it only needs the first layer. ESOINN separates nodes into different subclasses and deletes edges that lie in overlapped areas. It finds the overlapped area in the composite class and avoids building a connection between different classes. RE-SOINN incorporates the regression method into ESOINN in order to get continuous values. CURL performs task inference via a mixture of Gaussians latent space, and uses dynamic expansion and mixture generative replay to instantiate new concepts and minimize catastrophic forgetting. CF adjusts a set of concepts to environmental changes without labels. First, a supervised learner is used to learn the initial concepts. Then, CF adapts the concept drift in an unsupervised way. AutoNovel combines three ideas: 1) uses self-supervised learning to train on the unlabelled data; 2) employs ranking statistics to improve the clustering; and, 3) joint optimizing both labelled and unlabelled data, it reinforces the two tasks while avoiding forgetting. The authors also propose a method to estimate the number of classes in the unlabelled data.

Lastly, iLAP [43] is a replay-based unsupervised incremental method. iLAP trains using exposures, which are sets of images belonging to a single class. Two exemplar sets (train and validation) are maintained at all times, which contain samples for each class the model has currently determined. Every incoming exposure is treated as a new class assigning it label $k + 1$, where k is the number of classes seen so far. A copy of the model is trained using this exposure and the exemplars. If the accuracy of a class decreases more than a threshold, this exposure is likely to belong to that class. Otherwise, $k + 1$ is the appropriate label for the new class.

5 Comparison among Relevant Methods

In this section, we evaluate a selection of the methods surveyed above through a series of image classification experiments. First, we present the experimental methodology in Section 5.1, the datasets included in our benchmark are described in Section 5.2, while the experimental setup is outlined in Section 5.3. Sections 5.4 and 5.5

evaluate the influence of the unavailability/availability of memory, either for the Task-IL and for the Class-IL scenarios, while Section 5.6 assesses, for the latter case, the influence of the amount of available memory. Other crucial parameters are the complexity of the base network, which is evaluated in Section 5.7, and the ordering of tasks, considered in Section 5.8. To finish, Section 5.9 brings into the discussion the results available from one of the most used datasets for evaluating IL algorithms, MNIST, to complete the evaluation as this dataset is not involved in the preceding comparisons. This section considers both Task-IL and Class-IL methodologies and a selection of data-free methods.

5.1 Experimental Methodology

The algorithms that we include in the comparison as a whole are well-known, relevant methods from a varied set of categories among those identified in Section 3: methods adding regularization terms to the loss function to counteract forgetting (iCaRL, LwF, EWC, MAS, SI, RWalk, DGR), methods based on knowledge distillation (iCaRL, BiC, LwF, LwM, DI, RTF, DKDF), methods making use of exemplar replay (iCaRL, DGR), variational (DGR) and unsupervised (CURL) approaches, and data-free methods (ARM, DI, BI-R, RTF, DKDF). To carry out the comparative study, we have made use of the FACIL framework by [62] [1]. Moreover, for comparison purposes, we incorporate into the analysis the baselines *Finetuning* and *Joint (training)* as, respectively, lower- and upper-bounds:

(1) *Finetuning* corresponds to a model that does not incorporate any specific feature to prevent forgetting, and hence starts from the previous task model to greedily train each task, naturally leading to catastrophic forgetting. This baseline represents the minimum desired performance.
(2) *Joint (training)* assumes all data in the task sequence is available simultaneously, instead of task by task, hence violating the continual learning setup. This baseline is intended to represent the target performance.

For a principled and varied evaluation, we consider several datasets commonly accepted by the research community as part of IL benchmarks (they are detailed in Section 5.2). If not stated otherwise, full datasets are split 80% for training and 20% for test. Each experiment is repeated 10 times for datasets *CIFAR-100* and *Core50*, and 3 times for dataset *MiniImageNet*.

To measure performance in the continual learning setup, following standard practices in the field [23, 46], we evaluate the performance of a model on all tasks k up to t once that model has finished learning task t, so that, given T tasks, we can define a matrix $P \in \mathbb{R}^{T\times T}$, with $P_{i,j}$ as the *test performance* for task $t_i \leq t_j$ after training on task t_j:

$$P = \begin{bmatrix} P_{1,1} & - & - & - \\ P_{2,1} & P_{2,2} & - & - \\ \vdots & \vdots & \ddots & - \\ P_{T,1} & P_{T,2} & \dots & P_{T,T} \end{bmatrix} \tag{1}$$

where $P_{i,j} = \sum_{k=1}^{n_{t_j}} p_{i,j,k}/n_{t_j}$ with $p_{i,j,k}$ as the performance for class c_k computed after task t_i, being c_k part of task t_j, comprising n_{t_j} classes. Column i of P collects thus the performance levels achieved for task t_i after training task $j \geq i$ (entry j).

We make use of *accuracy* as the main performance metric to fill matrix P. Hence, $p_{i,j,k}$ is the accuracy for class c_k computed after task t_i. After training all tasks, from P we calculate the *average accuracy per task* A_i and the *average forgetting per task* F_i, as well as the *average accuracy* A and the *average forgetting* F [2] as global

[1] https://github.com/mmasana/FACIL
[2] also known as *backward transfer*

performance metrics:

$$i = 1\,..\,T \;:\quad A_i = \frac{1}{i}\sum_{j=1}^{i} P_{i,j}\;, \qquad\qquad A = \frac{1}{T}\sum_{i=1}^{T} A_i \tag{2}$$

$$i = 1\,..\,T-1 \;:\quad F_i = \frac{1}{T-i}\sum_{j=i+1}^{T} \left(P_{i,i} - P_{j,i}\right), \quad F = \frac{1}{T-1}\sum_{i=1}^{T-1} F_i \tag{3}$$

The *average forgetting* for task T is $F_T = 0$.

From a methodological point of view, through the experiments described in Sections 5.4 - 5.9 and the datasets involved, we take into account diverse crucial aspects affecting the performance of IL algorithms. More precisely, the assessment carried out contemplates:

(1) both the Task-IL (TIL) and the Class-IL (CIL) scenarios, respectively denoted under the terms *task-aware* and *task-agnostic* in FACIL;
(2) the unavailability/availability of memory for storing selected samples, i.e. *exemplars*;
(3) whether the storage capability is implemented through a *fixed-size memory*, with capacity for a certain number of exemplars among all classes, or by means of a *growing memory*, with a limit in the number of exemplars per class that can be stored;
(4) the amount of memory available in case exemplars can be stored;
(5) the influence on performance of the complexity of the base network, i.e. number of trainable parameters;
(6) the impact of task/class ordering; and
(7) the perturbation associated to a varying background in the specific case of image classification (through the Core50 dataset, see Section 5.2).

5.2 Datasets

To evaluate the performance of the different methods under different circumstances, we conduct a number of image classification experiments on three datasets taken under different conditions, namely *CIFAR-100*, *Core50* and *MiniImageNet*. The features of each dataset are briefly reviewed next:

1. *CIFAR-100* [47, 74] contains 32×32-pixel colour images for 100 classes, with 600 samples for each class divided into 500 samples for training and 100 samples for testing. For data augmentation, a padding of 4 is added to each side, and crops of 32×32 are randomly selected during training, while the center crop is used during testing. This dataset was divided in 10 tasks, each task containing 10 classes.
2. *Core50* [55] comprises images of 50 domestic objects belonging to 10 categories. Classification can be performed at object level (50 classes) or at category level (10 classes: plug adapters, mobile phones, scissors, light bulbs, cans, glasses, balls, markers, cups and remote controls). The dataset was collected in 11 distinct sessions characterized by different backgrounds and lighting (8 indoor sessions and 3 outdoor sessions). A 15 seconds video (at 20 fps) was recorded for each object and for each session, using a Kinect 2.0 sensor. Objects were hand-held by the operator, and the camera point-of-view was that of the operator eyes. The operator was required to extend his arm and smoothly move/rotate the object in front of the camera. The grabbing hand (left or right) changed throughout the sessions and relevant object occlusions were often produced by the hand itself. The full dataset consists of 164,866 RGB-D images of 128 × 128 pixels each. This dataset was split into 10 tasks, 5 classes per task.
   Furthermore, for experimentation purposes, we considered three configurations for this dataset: *Core50/2*, which collects data from two indoor sessions (S1 & S2), which were mixed and split 80% for training and 20% for test; *Core50/4/1*, which collects data from five sessions, four sessions for training (three indoor, S1-3, and one outdoor, S4) and one session for test (indoor, S5); and *Core50/4*, which collects data from four

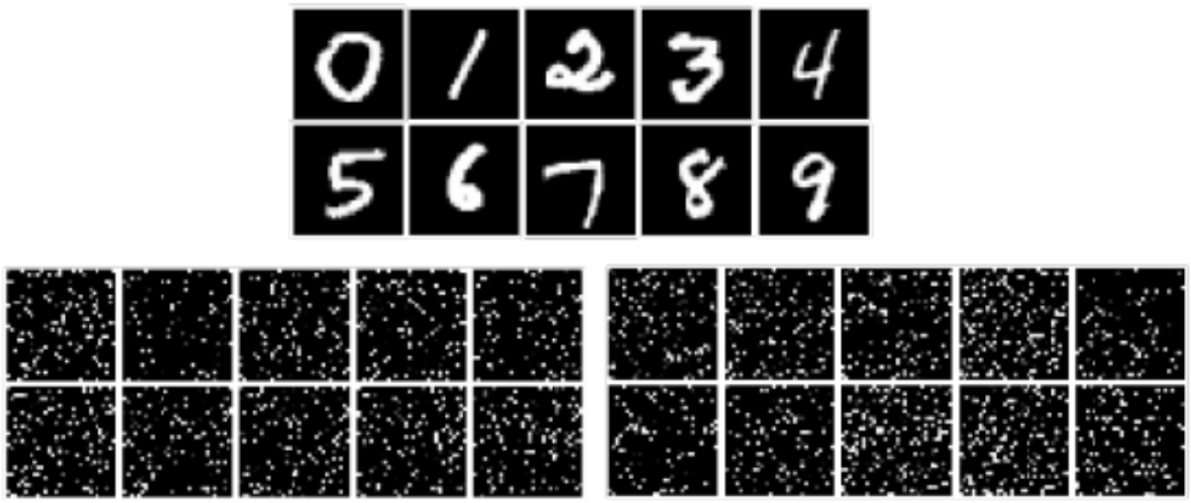

Fig. 4. Example tasks from the *Permuted MNIST* dataset. The original permutation is shown at the top, without any modification, while the bottom row shows two permutation examples.

sessions (three indoor, S1-3, and one outdoor, S4), which were mixed and split 80% for training and 20% for test.

3. *MiniImageNet* was proposed by [89] for few-shot learning evaluation. Its complexity is high due to the use of *ImageNet* images, although it requires fewer resources and infrastructure than running on the full *ImageNet* dataset. In total, there are 100 classes with 600 samples of 84×84 color images per class. In our experiments, we use all classes for training and for testing, unlike other setups where 80 classes are used for training and the remaining 20 classes for testing. This dataset was split into 10 tasks, 10 classes per task.

Apart from the aforementioned, we also involve into the discussion the dataset *MNIST* and two of its variations, given its widespread use within the research community:

4. *MNIST* [49] consists of 28×28-pixel grayscale images for 10 classes (one for each digit), with 7000 samples for each class divided into 6000 samples for training and 1000 samples for testing. This dataset is typically divided into 10 tasks, each digit corresponding to one task.
   We consider the following two variants of the *MNIST* dataset:
   - *Split MNIST*. In this case, the 10 classes are divided in 5 tasks as follows (see Figure 1): task 1 – digits 0 and 1; task 2 – digits 2 and 3; task 3 – digits 4 and 5; task 4 – digits 6 and 7; task 5 – digits 8 and 9.
   - *Permuted MNIST* (introduced in [31]). In this case, each of the 10 tasks is a multi-class classification problem for a different random permutation of the input pixels (see Figure 4). This variation of the *MNIST* dataset is mostly intended for the Domain-IL case.

   For the specific case of *MNIST*, we bring into the discussion performance values from the methods' original publications.

This benchmark intends to cover datasets commonly used by the incremental learning research community and, at the same time, consider a varied collection of objects, backgrounds and data capture conditions.

Table 2. Values employed for the generic hyper-parameters during model training.

parameter	**value(s)**
learning rate (*lr*) scheduler	
— initial value (first task)	0.5, 0.1, 0.05
— initial value (second and next tasks)	0.1, 0.05, 0.01, 0.005, 0.001
— patience	10 epochs
— decreasing factor	3
(*lr* is divided by *decreasing factor* after *patience* epochs)	
— minimum value	0.0001
(training stops when the *lr* gets below the minimum)	
momentum	0.9
weight decay	0.0002
gradient clipping	10000
(when its norm is above the *clipping* value)	
exemplars selection strategy	*herding*
(whenever memory is available, either fixed or growing size)	

5.3 Experimental Setup

Unless otherwise stated, the base network employed by default in all experiments is a ResNet-32-like model, as implemented in FACIL. ResNet-32 is a tiny CNN model adapted from the original ResNet models [35] to process small input images, e.g. 32×32 pixels; this model comprises 31 convolutional layers and 1 fully-connected layer, you can find the specific details of the network in Table 8 under the label ResNet-32*. All models are trained for a maximum of 200 epochs using SGD on mini-batches of 128 samples and the categorical cross-entropy loss, with dynamic learning rate, and using early stopping and momentum. General training hyper-parameters are detailed in Table 2. As indicated in the table, initial values for the learning rate are optimized through grid-search for all tasks in all cases, and differently for the first and subsequent tasks; it is done in this way for a more realistic setting of the correct learning rate and also to somehow counteract models' inability to learn a new task (i.e. *model intransigence*, as introduced in [14]). The values used for the particular main parameters of each of the IL algorithms involved in the comparison can be found in Table 3.

The equipment involved in the different tests consisted of two desktop computers, each equipped with an Intel Core i9-9900K @3.6GHz processor with 8 cores/16 threads and 64 GB of RAM. One of the computers was in turn fitted with an Nvidia GeForce RTX-4090 24GB GPU card, and the other with an Nvidia GeForce RTX 2080Ti 11GB GPU card.

5.4 Learning Performance when Memory is not Available

This section addresses the scenario where no memory is available and therefore exemplars are not stored for training previous tasks. On the one hand, Figures 5 and 6 show the evolution of the average accuracy task by task for EWC, LWM, SI, LwF, MAS and RWalk. The *Finetuning* and *Joint* baselines are also included. On the other hand, Table 4 provides, dataset by dataset, accuracy values averaged over all tasks (odd rows) as well as average forgetting values (even rows, in red).

As could be expected, Figures 5 and 6 show that, generally speaking, *Finetuning* acts as a lower-bound since it does not apply any modification to prevent forgetting, while *Joint* training acts as an upper-bound since it has all data available for training. On the other side, all methods perform better for the TIL scenario than

Table 3. Values for the specific parameters of each IL algorithm. (Below, λ behaves as a forgetting-intransigence trade-off mostly, see the corresponding work for details.)

IL algorithm	parameter	value
BiC [92]	T – temperature scaling	2
EWC [44]	λ – loss function balancing factor	10000
iCaRL [74]	λ – loss function balancing factor	4
LwF [53]	λ – loss function balancing factor	10
	T – temperature scaling	2
LwM [22]	β – distillation loss trade-off	2
	γ – attention loss trade-off	1
MAS [4]	λ – loss function balancing factor	400
RWalk [14]	λ – loss function balancing factor	20
SI [98]	λ – loss function balancing factor	10

for the more challenging CIL scenario. As for forgetting, Table 4 shows that, with some minor exceptions, the baselines *Finetuning* and *Joint* usually exhibit the, respectively, highest and lowest forgetting levels, as could also be foreseen.

Regarding the specific IL methods considered, LwF and LwM consistently exhibit better accuracy in both the TIL and CIL scenarios with very similar quantitative performance, irrespective of the dataset, although one may win in one case over the other; actually, for the TIL case and the *Core50/4/1* dataset, both methods even outperform the *Joint* baseline. In more detail, as reported in Table 4, LwF achieves the highest accuracy values after learning the 10 tasks for datasets *CIFAR-100* (TIL: 76.2, CIL: 44.0), *Core50/4/1* (TIL: 51.6, CIL: 20.6) and *MiniImageNet* (TIL: 53.8, CIL: 26.3), while LwM outperforms all for the *Core50/2* dataset (TIL: 67.3, CIL: 26.5); the two methods apply distillation, and LwF also makes use of regularization. The other four methods, EWC, MAS, RWalk and SI, achieve the weakest performance, at a significant distance from LwF and LwM; this happens for both scenarios and irrespective of the dataset. More precisely, for TIL: RWalk is the worst for *CIFAR-100* (TIL: 59.3) and *MiniImageNet* (TIL: 40.1), and SI for *Core50/2* (TIL: 48.5) and for *Core50/4/1* (CIL: 42.5); for CIL: EWC is the worst for *CIFAR-100* (CIL: 26.8) and *Core50/4/1* (CIL: 13.9), MAS for *Core50/2* (CIL: 18.5) and SI for *MiniImageNet* (CIL: 15.7).

On the other side, regarding forgetting, LwF consistently shows the lowest forgetting for TIL, even below the *Joint* baseline for the case of dataset *Core50/4/1*, while it is MAS the method that reduces forgetting most for CIL, also outperforming *Joint* for dataset *Core50/4/1*, and repeating for *MiniImageNet*. Surprisingly, LwM reaches the highest forgetting for CIL, even above the *Finetuning* baseline, for dataset *Core50/4/1*, while it is RWalk the one that, for TIL, exhibits the greatest tendency to forget. LwM gets intermediate positions in this scenario.

Finally, it is clear that the *Core50/4/1* and *MiniImageNet* datasets are the most challenging for the different methods evaluated, with accuracies dropping to levels between 15 and 20%, and around 30-40% for the upper-bound baseline *Joint*, in the CIL case.

5.5 Learning Performance when Memory is Available

We consider two types of memory for this evaluation: a fixed-size memory with capacity for 2000 exemplars, and a growing-size memory capable of storing 20 exemplars per class, which gives rise to a storage space of up to 1000 exemplars and 2000 exemplars when all tasks are considered for, respectively, datasets *Core50* and *CIFAR-100* / *MiniImageNet*. On this occasion, we incorporate BiC and iCaRL into the comparison as they rely on

Table 4. Summary of *average accuracy* (black) and *forgetting* (red) values for the *no-memory case* and all datasets. (The best results for each case [discarding the baselines] are highlighted in bold over non-white background. *Finetuning*, *Joint*: baselines.)

	Class-IL				Task-IL			
Method	**CIFAR-100**	**Core50/2**	**Core50/4/1**	**MiniImageNet**	**CIFAR-100**	**Core50/2**	**Core50/4/1**	**MiniImageNet**
EWC [44]	26.82	19.26	13.93	16.25	60.31	50.54	42.97	42.66
	15.13	18.74	11.76	6.70	14.65	25.23	13.43	9.83
LwF [53]	**43.99**	25.30	**20.59**	**26.34**	**76.20**	66.56	**51.63**	**53.83**
	14.01	20.76	16.84	8.75	**0.36**	**2.33**	**1.63**	**0.54**
LwM [22]	39.15	**26.48**	19.75	24.20	74.03	**67.30**	51.59	52.14
	34.74	28.76	22.03	19.79	6.73	6.00	3.97	5.72
MAS [4]	27.25	18.46	16.05	17.27	62.42	52.41	48.71	42.93
	5.94	**7.79**	**7.09**	**3.37**	7.66	14.61	9.11	6.44
RWalk [14]	26.83	19.02	15.56	15.78	59.30	48.69	44.51	40.10
	14.83	20.85	11.47	6.98	16.03	29.18	15.21	13.54
SI [98]	27.84	18.97	14.87	15.74	60.66	48.50	42.48	41.78
	10.39	19.09	9.83	5.25	12.82	28.00	14.71	9.70
Joint	69.82	59.97	27.91	40.15	86.12	79.54	51.26	62.72
	1.54	3.92	8.90	3.94	-1.02	-0.47	3.32	-0.92
Finetuning	23.75	19.59	12.34	15.93	49.81	46.22	37.89	35.15
	40.06	31.09	14.60	23.52	38.53	36.90	19.76	26.43

exemplars and therefore on the availability of memory, while the *Joint* baseline is discarded just because of the opposite since it does not make use of exemplars.

Figures 7 - 8 and 9 - 10 show the evolution of the average accuracy task by task for, respectively, the fixed-size and the growing-size memory cases. The numerical values for average accuracy and forgetting are reported in, respectively, Tables 5 and 6.

With no surprise, accuracy levels in the TIL scenario are again better than in the CIL scenario, because of the availability of the Task ID, as already pointed out in the previous section. On the other side, in contrast, on this occasion, *Finetuning* performs more like the *Joint* baseline since it has memory for past classes and this allows it to achieve significantly better performance, e.g. the highest for dataset *Core50/2*. Regarding forgetting, *Finetuning* is not affected by the highest forgetting levels in (almost-)all cases, contrary to the no-memory scenario considered in the previous Section 5.4: the worst combination is TIL/growing-memory, for which it exhibits the highest forgetting in three of the four datasets; for the other combinations, it reaches intermediate positions in the ranking, for both the fixed- and growing-size memory scenarios.

As for specific methods, when memory is incorporated, the two best performing algorithms for the no-memory case, LwF and LwM, still keep at the highest positions of the ranking, particularly LwM, being the best in 10 out

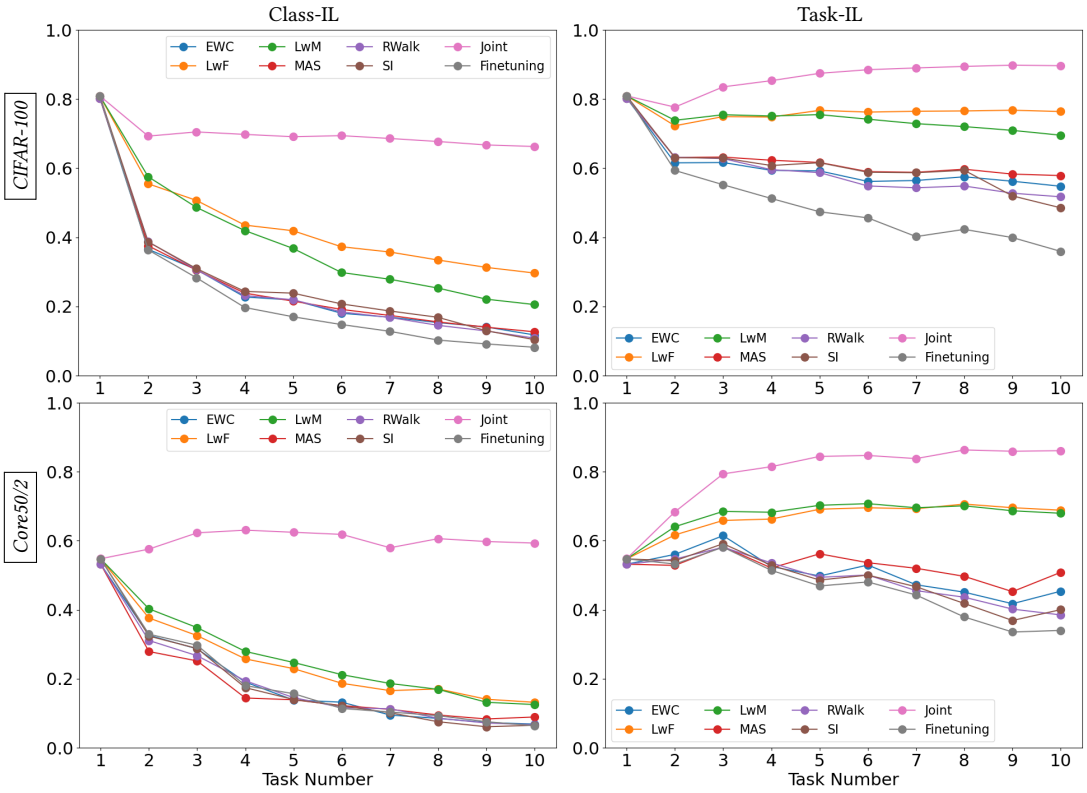

Fig. 5. *Accuracy* after learning each task for different datasets and IL algorithms and the *no-memory case* (1): Class-IL scenario [left], Task-IL scenario [right].

of the 16 dataset/IL scenario combinations, with best performance in the TIL case: LwM reaches accuracies of 79.0 [fixed] and 77.0 [growing] for *CIFAR-100*, 76.2 [fixed] and 72.7 [growing] for *Core50/2*, 55.4 [fixed] and 55.6 [growing] for *Core50/4/1*, and 55.5 [fixed] and 55.5 [growing] for *MiniImageNet*. In the CIL case, LwM gets the best for datasets *Core50/2* (50.5 [fixed], 41.5 [growing]), dataset *Core50/4/1* (25.0 [growing]) and dataset *MiniImageNet* (22.9 [growing]). With memory available, BiC, based on exemplar replay, outperforms the other methods on several occasions and actually becomes the second-best. The highest performance is observed in the CIL case for fixed-size memory: 53.6 for *CIFAR-100*, 46.5 for *Core50/2*, 27.4 for *Core50/4/1*, and 32.2 for *MiniImageNet*; moreover, BiC gets very close in performance to the best (LwM) when it is not the best. Finally, iCaRL is the winner for the *MiniImageNet*/growing-memory combination, with an accuracy of 29.9, although it always gets very close to the best method for the other datasets for the aforementioned combination. Notice that LwM, BiC and iCaRL all apply distillation, and iCaRL and BiC specifically make use of exemplars to replay during training. SI exhibits the worst performance irrespective of the dataset and both for TIL and CIL (in 8 out of 16 combinations), together with MAS (in 6 out of 16 combinations); both make use of regularization.

As for forgetting, without doubt, the best performer is iCaRL, being the least affected in 14 of the 16 combinations between datasets, CIL/TIL scenarios and fixed/growing-size memory, even showing negative forgetting [3], i.e. for the TIL/fixed-memory case. BiC is the second best performer, with forgetting levels comparable to those of

[3]In the task learning sequence, the accuracy gets better for previous tasks after learning posterior tasks.

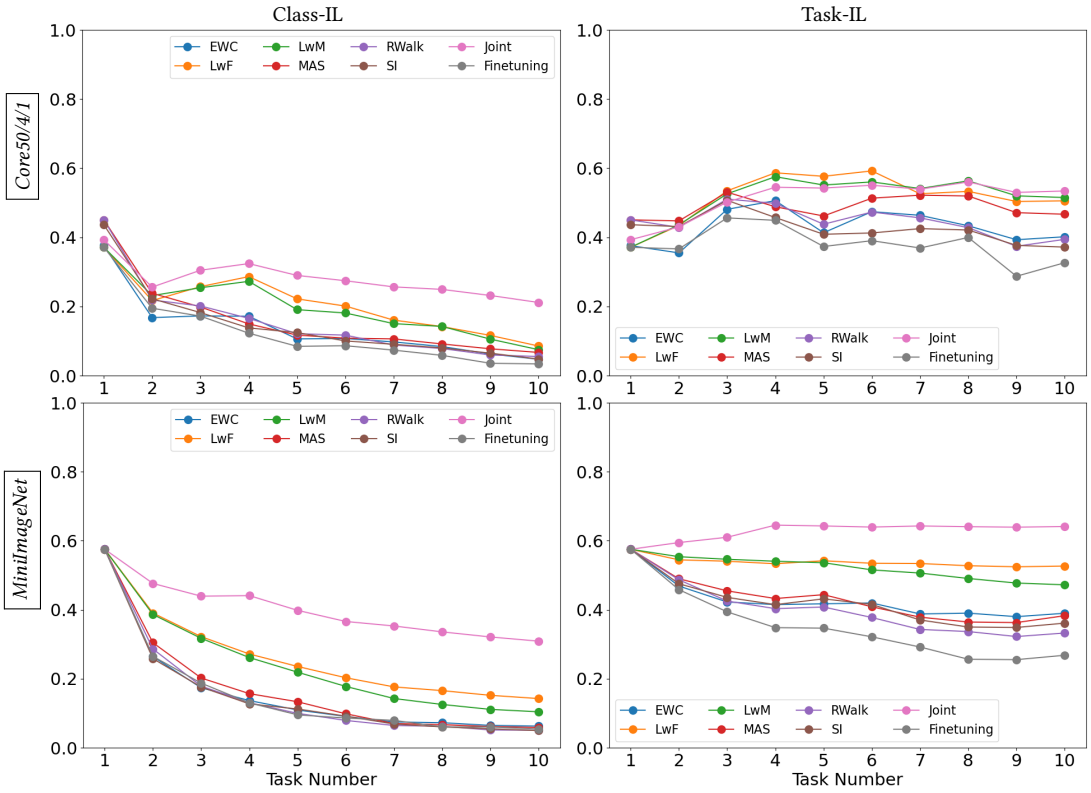

Fig. 6. *Accuracy* after learning each task for different datasets and IL algorithms and the *no-memory case* (2): Class-IL scenario [left], Task-IL scenario [right].

iCaRL. LwF gives rise to the lowest forgetting values in the TIL scenario for *CIFAR-100* and *Core50/4/1*, being very close to iCaRL for *Core50/2* and *MiniImageNet*. For the other methods, forgetting is considerably higher for CIL, while the values are closer and reduced for TIL. Moreover, the forgetting levels, though being quite contained, are larger for TIL/growing-memory than for TIL/fixed-memory. SI is the method with the worst behaviour in 9 out of the 16 combinations, being concentrated in the fixed-size memory scenario; EWC, RWalk and MAS have occasionally exhibited also maximum values of forgetting.

Finally, the availability of memory does not prevent *Core50/4/1* and *MiniImageNet* from also being the most challenging datasets for the different methods evaluated, similarly to the no-memory case. On this occasion, accuracies drop to levels between 20 and 35% in the CIL case, clearly above those of the no-memory case.

5.6 Learning Performance for Different Memory Sizes

As a continuation of Section 5.5, in this section we contemplate the effect on performance of the memory size for both the TIL and CIL cases. For the fixed-size memory case, we consider capacities of 1000, 2000, 3000 and 4000 exemplars, while, for growing memory, we evaluate the impact of 10, 20, 30 and 40 exemplars per class. Moreover, we restrict the comparison to BiC and iCaRL, as they are two of the methods considered in this survey that make use of exemplar replay (and hence can be mostly affected by the actual amount of available memory). Given the good performance exhibited by both, as reported in Section 5.5, our aim is to determine whether a

Table 5. Summary of *average accuracy* (black) and *forgetting* (red) values for the *fixed-size memory case* (2000 samples) and all datasets. (The best results for each case [discarding the baselines] are highlighted in bold over non-white background. *Finetuning*, *Joint*: baselines.)

	Class-IL				Task-IL			
Method	**CIFAR-100**	**Core50/2**	**Core50/4/1**	**MiniImageNet**	**CIFAR-100**	**Core50/2**	**Core50/4/1**	**MiniImageNet**
BiC [92]	**53.61**	46.53	**27.44**	**32.20**	77.73	72.93	54.30	**56.09**
	7.63	11.85	14.12	9.44	0.59	-0.02	1.57	0.72
EWC [44]	43.22	41.95	25.53	24.07	73.18	69.50	52.31	51.20
	40.65	21.20	20.24	33.64	1.85	2.12	3.40	1.57
iCaRL [74]	47.87	43.06	24.52	21.91	72.83	72.51	53.87	37.44
	6.49	**5.64**	**6.11**	**3.15**	**-0.15**	**-0.43**	**0.17**	**-0.13**
LwF [53]	45.99	43.03	24.91	25.61	77.39	72.83	54.46	55.71
	34.99	22.72	25.94	33.96	-0.13	-0.43	0.73	0.01
LwM [22]	50.51	**50.46**	26.99	26.94	**79.04**	**76.21**	**55.38**	55.45
	34.84	20.36	22.28	34.45	1.08	0.28	1.36	1.14
MAS [4]	36.36	32.11	24.38	22.03	68.08	62.24	54.40	48.85
	42.27	24.15	28.26	32.55	1.41	2.92	3.27	1.25
RWalk [14]	37.17	38.27	24.95	23.00	65.99	66.87	53.37	49.75
	38.71	24.07	26.17	33.11	2.51	3.59	4.59	1.47
SI [98]	36.05	33.07	22.78	21.66	66.33	61.81	51.86	48.32
	40.87	31.01	33.64	34.65	2.43	6.28	6.09	3.46
Finetuning	50.98	52.35	26.00	27.41	77.87	76.33	52.43	54.54
	37.53	22.68	23.38	36.04	3.53	1.94	2.67	3.53

larger memory improves performance. For testing, we make use of the *Core50/4* dataset, where, as already said, classification performance can be affected by changing backgrounds, in particular due to the combination of images from 3 indoor scenarios and 1 outdoor scenario.

Figures 11 and 12 plot average accuracy and average forgetting values per task for TIL/CIL and fixed-memory/growing-memory scenarios. The plots show a similar performance task per task for all combinations of memory and algorithm, decreasing in accuracy for CIL and increasing for TIL, as already observed in the previous sections. As for forgetting, one can observe, task per task, two clusters that group the four cases of BiC and the four cases of iCaRL in the CIL scenario, with BiC always at higher forgetting levels than iCaRL. For the TIL scenario, the two clusters cannot be distinguished, although the range of values covered is larger for growing-size memory (around 0) than for fixed-size memory (from 0 to 0.05).

On the other side, the numerical values for the global averages can be found in Table 7. As can be observed, increasing the amount of available memory increases the accuracy for both BiC and iCaRL. The increase in accuracy between the smallest (1000/10) and the biggest memory (4000/40) cases, although not terribly significant,

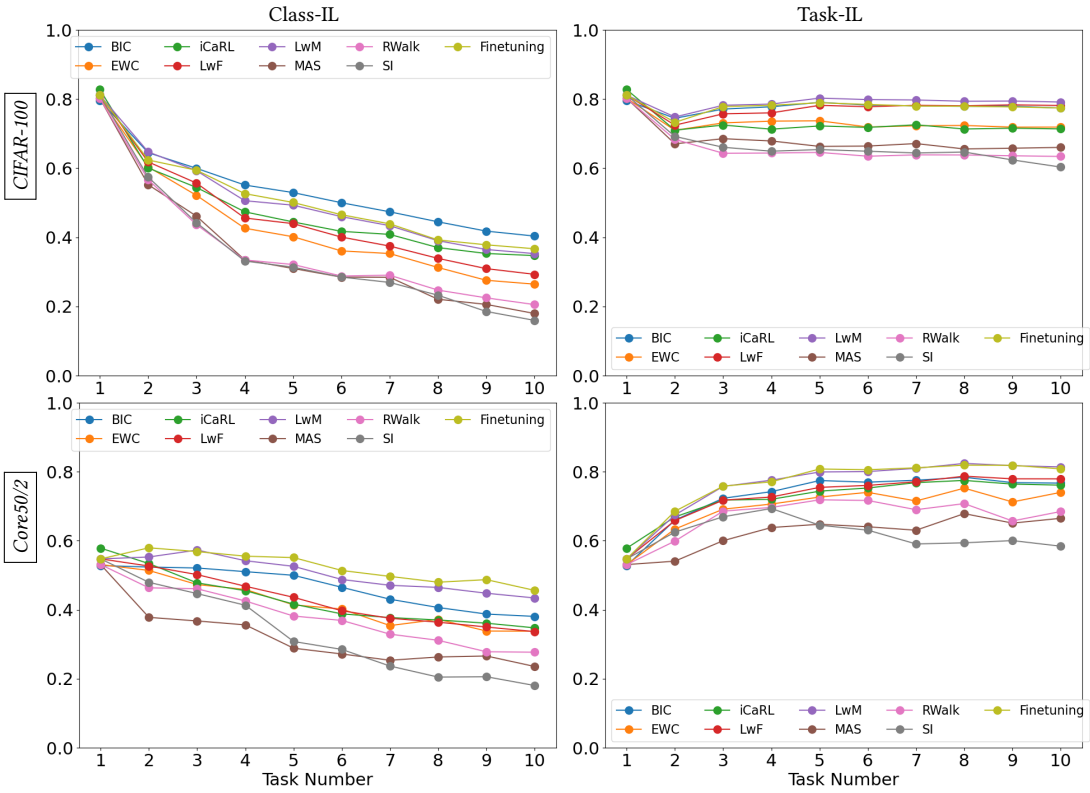

Fig. 7. *Accuracy* after learning each task for different datasets and IL algorithms and the *fixed-size memory case* with a capacity of 2000 samples (1): Class-IL scenario [left], Task-IL scenario [right].

is larger for the CIL scenario than for the TIL scenario, and higher for BiC than for iCaRL (of the order of 3 times for BiC and 1.5 times for iCaRL). A larger memory also benefits from the forgetting point of view, again larger for CIL than for TIL, and higher for BiC than for iCaRL (almost 4 times for BiC and almost 2.5 times for iCaRL).

In absolute terms, the largest accuracy is attained by BiC with the biggest memory, while the lowest forgetting is for iCaRL for the largest capacity (growing-memory) or for the second to last largest capacity (fixed-memory).

5.7 Influence of the Complexity of the Base Network

Similarly to Section 5.6, in this section, we also consider whether an increase in the computational resources available for the algorithm gives rise to an increase in performance. On this occasion, we evaluate the impact of the network complexity, whether a higher amount of elaborated information about the input samples mean better performance. To this end, since all tests up to now have been performed using a ResNet-32-like network, we follow the same kind of architecture (making use of residual blocks and skip connections) and add layers and/or feature maps. Table 8 describes the five networks for which we report performance:

- they all share the same first convolutional layer (*conv1*) as in the original publication [35], but the number of feature maps is divided by four in all layers (*conv1* and *conv2* - *conv5*);
- all the convolutional layers use 3×3 kernels, contrary to the original architecture that features a 7×7 convolution kernel in the *conv1* layer;

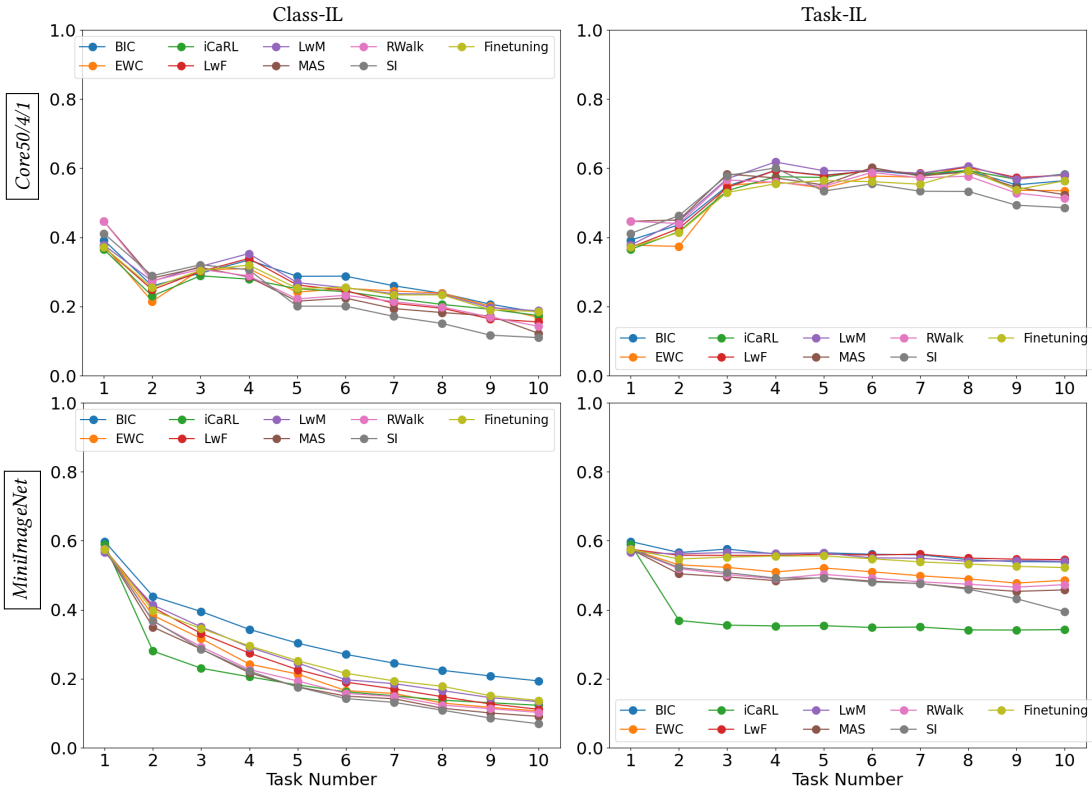

Fig. 8. *Accuracy* after learning each task for different datasets and IL algorithms and the *fixed-size memory case* with a capacity of 2000 samples (2): Class-IL scenario [left], Task-IL scenario [right].

- finally, all models use the same basic block comprising two convolutional layers with the same number of planes to learn the residuals, what coincides with the original models up to ResNet-34 [35], but not with models involving 50 layers or more, which, in each basic block, employ three convolutional layers and a higher number of planes in the last one.

Because of those differences, we denote the resulting models as ResNet-*nn**. Our aim is to test homogeneous changes in complexity, not drastic changes in the architecture.

The last rows of the table informs on (1) the total number of layers —always $nn - 1$ convolutional layers *c* and a last fully connected layer *fc*, which is replaced by the IL algorithms to implement the specific incremental learning approach—, (2) the total number of feature maps across all layers, and (3) the resulting total number of trainable parameters, which ultimately determines the model complexity. The latter is orientative, assuming the output layer contains as many neurons as tasks.

For testing each model, we make use of dataset *Core50/2*, the fixed-size memory scenario and consider the effects on the behaviour of the BiC, EWC, iCaRL, LwF, LwM and RWalk algorithms when the base network is changed. We consider *Core50/2* because of its diversity. The plots of Figures 13 and 14 show the evolution of the average accuracy per task for the aforementioned six IL methods, all networks and the TIL/CIL scenarios, while Table 9 reports the final average values for accuracy and forgetting.

Table 6. Summary of *average accuracy* (black) and *forgetting* (red) values for the *growing-size memory case* (20 samples/task) and all datasets. (The best results for each case [discarding the baselines] are highlighted in bold over non-white background. *Finetuning*, *Joint*: baselines.)

	Class-IL				Task-IL			
Method	**CIFAR-100**	**Core50/2**	**Core50/4/1**	**MiniImageNet**	**CIFAR-100**	**Core50/2**	**Core50/4/1**	**MiniImageNet**
BiC [92]	**47.92**	39.93	24.96	29.77	75.28	69.42	54.64	55.00
	13.91	16.23	17.01	13.60	2.08	1.29	1.20	2.07
EWC [44]	35.68	36.86	23.40	19.73	71.23	67.22	52.89	49.22
	49.60	37.49	31.67	39.31	3.44	4.70	3.03	2.93
iCaRL [74]	46.54	40.32	24.78	**29.90**	72.41	70.51	53.92	53.69
	8.28	**6.94**	**6.89**	**5.91**	0.37	**0.01**	1.45	**0.02**
LwF [53]	37.08	32.91	22.61	21.07	75.95	69.01	54.42	55.39
	43.92	35.36	33.73	39.70	**0.07**	0.16	**1.08**	0.22
LwM [22]	41.67	**41.52**	**25.00**	22.85	**77.04**	**72.68**	**55.58**	**55.48**
	43.42	32.39	31.09	40.61	1.83	1.01	1.66	1.85
MAS [4]	30.26	25.97	21.25	17.98	66.54	59.81	54.45	47.39
	49.51	42.69	42.41	37.21	2.58	4.93	4.41	2.33
RWalk [14]	31.92	34.98	22.48	19.29	65.42	66.31	52.38	48.93
	45.51	37.93	35.45	38.92	3.26	4.55	5.36	2.73
SI [98]	29.51	31.74	19.40	18.35	63.05	64.90	47.54	47.96
	45.99	44.19	37.84	38.56	2.16	5.70	8.23	2.25
Finetuning	41.63	39.26	22.43	22.27	74.14	68.46	50.56	51.47
	46.80	39.53	32.87	42.31	6.12	6.05	4.38	6.02

As can be observed in the plots, for all networks and algorithms, the accuracy curves follow almost exactly the same trend with slightly different numerical differences per task; only RWalk seems to have some problems with ResNet-18*, and for both the TIL and CIL cases. This suggests that there is not a large difference in the performance of models regarding the different IL algorithms and the dataset considered. Globally speaking, ResNet-18* gives rise to the highest accuracy most times (5 out of 12), followed by ResNet-34* (4/12), and occassionaly ResNet-20* (2/12) and ResNet-32* (1/12). It must be noted that ResNet-18* features the least number of layers, but not in terms of feature maps, so that its overall complexity is not the lowest but intermediate; ResNet-34* features almost twice the number of layers and also the largest number of feature maps (almost twice as many as ResNet-18*), leading to the highest-complexity model. So, it seems that, regarding accuracy, the different algorithms benefit from the extra complexity of the network although rather more in terms of the number of feature maps than in terms of the number of layers. As for forgetting, ResNet-18* clearly outperforms all models, winning most times (8/12) and mostly for the CIL case; occasionally, ResNet-50* (2/12), ResNet-20* (1/12) and ResNet-34* (1/12) lead to the least forgetting.

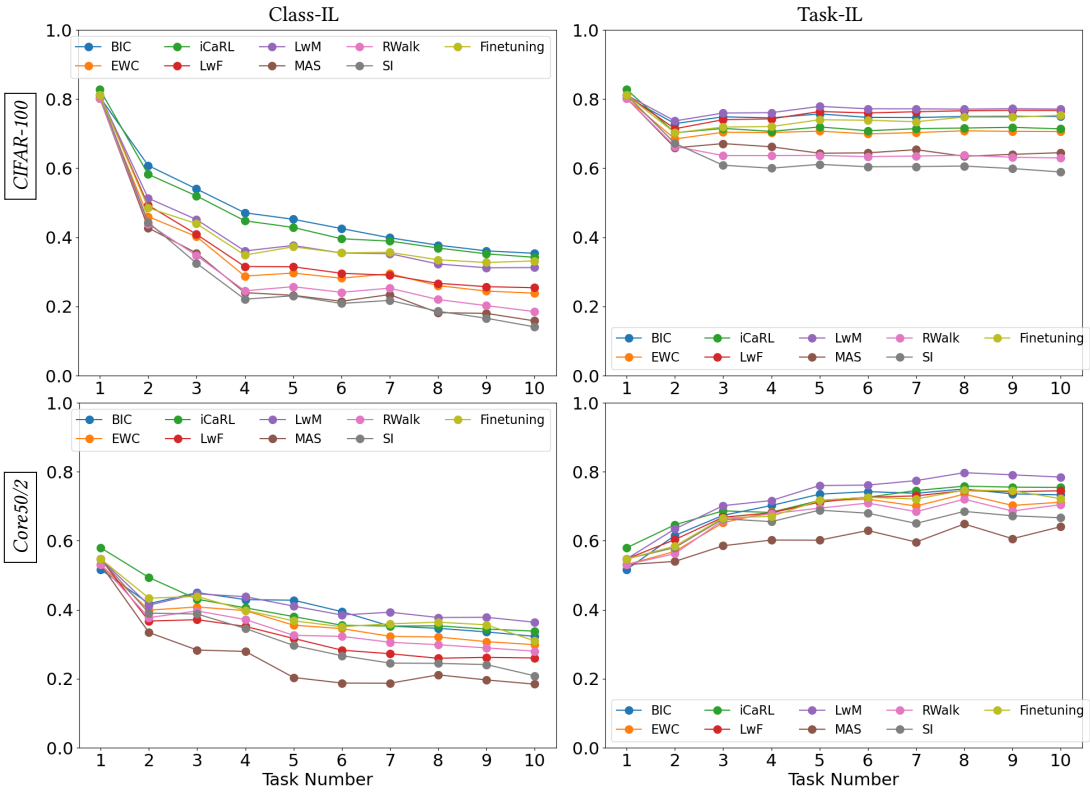

Fig. 9. *Accuracy* after learning each task for different datasets and IL algorithms and the *growing-size memory case* with a capacity of 20 samples/task (1): Class-IL scenario [left], Task-IL scenario [right].

Considering each algorithm separately, the method which is the most sensitive to the change in the base network is RWalk, in both the CIL and TIL scenarios, with differences of 10 - 11 points between best and worst performance across networks; the least sensitive is EWC for the CIL case (with a difference of 2.7), and LWM for the TIL case (1.4), being the latter also the second least sensitive for the CIL case (4.3); moreover, LwM achieves the highest accuracy for ResNet-18* in the TIL scenario (76.5) and also for CIL (52.5). Referring to forgetting, RWalk exhibits the largest dependence on the network (with a difference of 2.7 for TIL and 7.6 for CIL), while iCaRL is the least sensitive (0.5 for TIL and 0.6 for CIL). iCaRL achieves the best accuracy for ResNet-34* for both TIL (75.3) and CIL (46.5).

5.8 Impact of Task/Class Ordering

When testing incremental learning algorithms, it is usual to group classes in tasks in a sequential way. This can be beneficial or harmful for certain algorithms processing certain datasets, just because of the order in which classes are learned/tasks are solved, giving rise to a misleading measurement of performance. This section evaluates the effects of task/class ordering on the behaviour of a selection of IL methods making use of dataset *Core50/2* and the fixed-size memory scenario. For this experiment, we consider three different orderings: the first ordering (TO1) coincides with the one that has been used in the previous experiments, in which task 1 comprises classes 1 to 5, task 2 comprises classes 6 to 10, etc.; the second (TO2) and third orderings (TO3) come from two random

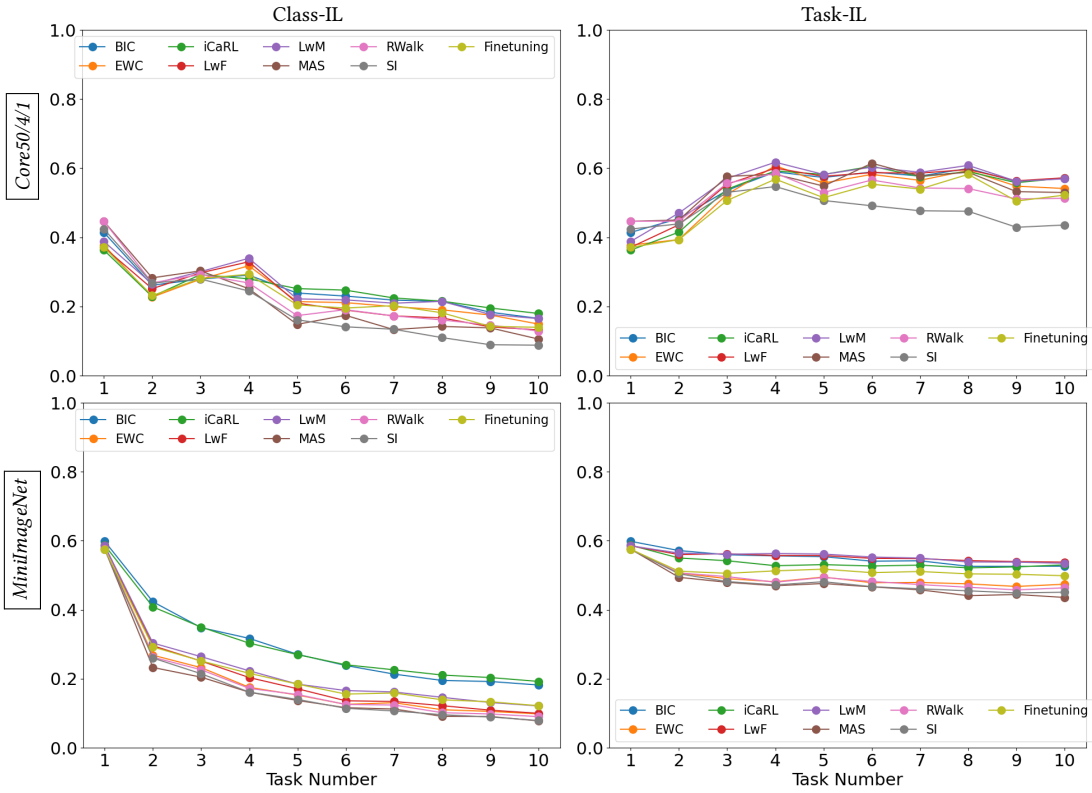

Fig. 10. *Accuracy* after learning each task for different datasets and IL algorithms and the *growing-size memory case* with a capacity of 20 samples/task (2): Class-IL scenario [left], Task-IL scenario [right].

permutations of classes 1 to 50. By way of example, the second ordering defines task 1 as the group involving classes 17, 35, 48, 9 and 36, while the third ordering defines task 1 as the group of classes 20, 12, 28, 40 and 9.

The plots in Figures 15 and 16 show accuracy values task by task for, respectively, the two scenarios TIL and CIL, and the algorithms BiC, EWC, iCaRL, LwF, LwM and RWalk. As can be observed, the learning for each ordering starts from a different accuracy value, as the first task is different for each case, but all curves show a trend that converges to a similar level of performance after the last task, irrespective of whether it is the TIL or the CIL case, what indicates that task ordering does not have a dramatic influence in the behaviour of the algorithms. Table 10 corroborates these findings, although the final values indicates that task ordering 2 seems to be more favourable for all algorithms, since it gives rise to a higher level of accuracy. iCaRL attains the highest accuracy for the TIL case (80.4), while LwM is the winner for the CIL case (56.8); both algorithms score as the second best in the reverse case. The lowest forgetting levels are distributed between TO2 and TO3 for the CIL case and between TO2 and TO1 for the TIL case, being iCaRL the global winner in the CIL case (4.5) and LwF in the TIL case (-0.5).

5.9 Discussion on Performance in MNIST

For the completeness of the experimental evaluation of IL algorithms, in this section, we consider *MNIST* as one of the datasets most used when evaluating the performance of IL methods. The comparison involves several of

Table 7. Summary of *average accuracy* (black) and *forgetting* (red) values for *Core50/4* and different memory sizes. (The best results for each case are highlighted in bold over non-white background. *Fixed*: fixed-size memory, *Grow*: growing-size memory.)

		Class-IL		Task-IL	
Method and memory size		**Fixed**	**Grow**	**Fixed**	**Grow**
BiC [92]	1000/10	46.08	40.22	75.73	72.72
		18.87	22.94	0.44	2.31
	2000/20	47.67	43.42	75.49	74.03
		16.00	21.07	0.10	1.13
	3000/30	49.08	44.06	75.90	74.09
		13.52	19.98	0.07	1.03
	4000/40	**50.57**	**45.77**	**76.49**	**75.37**
		13.20	19.91	-0.06	0.53
iCaRL [74]	1000/10	45.51	43.15	74.03	72.49
		7.74	11.72	-0.03	1.94
	2000/20	46.98	44.59	74.99	73.48
		6.46	8.68	-0.39	0.63
	3000/30	46.61	45.19	74.55	74.06
		6.14	7.99	**-0.54**	0.18
	4000/40	47.60	45.49	75.38	74.03
		6.55	**7.85**	-0.26	**0.10**

the algorithms considered in the previous sections, but also adds others using performance values collected from the original publications.

Table 11 shows accuracy values for iCaRL, LwF, EWC, SI and MAS, and also for GEM, DGR, RTF, Bi-R and CURL, contemplating both *Split MNIST* and *Permuted MNIST*. For the *Split MNIST*/TIL case, LwF (99.6) and RTF (99.7) obtain the best average accuracy; both methods apply distillation, LwF also uses regularization and RTF is a data-free method. To the contrary, the methods with lowest accuracy are EWC (98.6) and GEM (98.4), both based on regularization, although the performance level for all methods in this case is so high (almost perfect) that there is really nothing to complain about. For the *Split MNIST*/CIL case, iCaRL, which is a distillation and exemplar replay method, obtains the by far best result (94.6) with respect to LwF, EWC, SI and MAS; compared to GEM, DGR, RTF, Bi-R and CURL the differences are considerably smaller. Finally, for the *Permuted MNIST*/CIL case, the best result is obtained by RTF (96.2), far above LwF, EWC and MAS, but relatively close to SI, GEM, DGR, Bi-R and CURL. At this poitn, it is important to notice that the lower accuracy values achieved by some methods in both CIL cases are due to the fact that they do not use exemplars, while the highest performing methods have exemplar memory.

Next, we compare the performance of several data-free methods on the *MNIST* dataset. The mean accuracy after learning the 10 tasks is shown in Table 12. As can be seen, the best method among the ones in the comparison is RTF (92.6). RTF is also compared with other non-data-free methods in Table 11, where it achieves the best performance for the *Split MNIST*/TIL (99.7) and *Permuted MNIST*/CIL (96.2) combinations, while, for *Split MNIST*/CIL, it scores in the third best position (92.6).

Table 8. Features of the ResNet-like models considered ($k \times k$, p: kernel, planes [feature maps], *conv,c*: convolutional layer, *fc*: fully connected layer, *ap2D*: 2D average pooling with $kp \rightarrow o$ denoting kernel for pooling [kp] & output [o]).

block	**ResNet-18***	**ResNet-20***	**ResNet-32***	**ResNet-34***	**ResNet-50***
conv1			$3 \times 3, 16$ $\times 1$		
conv2.x	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 8$
conv3.x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 8$
conv4.x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 8$
conv5.x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$			$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$	
ap2D	kp: $4 \rightarrow 128$	kp: $8 \rightarrow 64$	kp: $8 \rightarrow 64$	kp: $4 \rightarrow 128$	kp: $8 \rightarrow 64$
fc	128×10	64×10	64×10	128×10	64×10
no. layers	18 = 17 c + 1 fc	20 = 19 c + 1 fc	32 = 31 c + 1 fc	34 = 33 c + 1 fc	50 = 49 c + 1 fc
feat. maps	976	688	1,136	1,904	1,808
no. params.	701,466	272,474	466,906	1,334,618	758,554

Table 9. Summary of *average accuracy* (black) and *forgetting* (red) values for the *Core50/2* dataset for different IL algorithms and networks, and the fixed-size memory case (2000 exemplars). (The best results for each IL algorithm and IL scenario are highlighted in bold over non-white background.)

	Class-IL						**Task-IL**					
	BiC [92]	EWC [44]	iCaRL [74]	LwF [53]	LwM [22]	RWalk [14]	BiC [92]	EWC [44]	iCaRL [74]	LwF [53]	LwM [22]	RWalk [14]
ResNet-18*	42.03	**43.63**	45.56	**45.45**	**52.47**	31.29	67.16	**70.82**	74.28	72.40	**76.48**	56.79
	10.82	**19.86**	5.59	**19.48**	**18.51**	**16.52**	**-0.39**	**1.82**	-0.26	-0.39	**-0.10**	4.27
ResNet-20*	46.32	40.93	44.00	42.92	49.58	**41.08**	**73.58**	68.58	73.20	72.81	76.25	68.36
	14.92	24.30	5.72	23.22	21.03	20.90	0.37	3.24	-0.53	-0.24	0.14	**1.56**
ResNet-32*	**46.53**	41.95	43.06	43.03	50.46	38.27	72.93	69.50	72.51	72.83	76.21	66.87
	11.85	21.20	5.64	22.72	20.36	24.07	-0.02	2.12	-0.43	-0.43	0.28	3.59
ResNet-34*	43.98	43.36	**46.49**	44.23	50.36	40.71	70.35	70.43	**75.27**	**73.41**	75.93	**68.42**
	12.81	20.29	**5.22**	21.77	20.25	21.87	0.07	2.30	-0.19	-0.41	0.13	4.23
ResNet-50*	38.80	41.84	41.39	40.33	48.13	38.25	67.05	69.92	70.59	71.15	75.05	67.10
	12.01	22.01	5.81	21.72	21.42	23.68	-0.21	2.72	**-0.72**	**-0.79**	0.69	3.17

Table 10. Summary of average accuracy values for the *Core50/2* dataset for different IL algorithms and task orderings, and the fixed-size memory case [2000 exemplars]. (The best ordering for each IL algorithm and IL scenario is highlighted in bold over non-white background. The best overall outcome for each IL scenario is highlighted in white text.)

	Class-IL			**Task-IL**		
Method	**TO1**	**TO2**	**TO3**	**TO1**	**TO2**	**TO3**
BiC [92]	46.53	**54.58**	47.61	72.93	**77.70**	75.30
	11.85	**10.38**	11.26	**-0.02**	0.12	0.15
EWC [44]	41.95	**50.16**	47.22	69.50	**75.11**	73.82
	21.20	20.47	**20.45**	**2.12**	2.56	2.70
iCaRL [74]	43.06	**55.78**	52.92	72.51	**80.40**	78.96
	5.64	**4.50**	5.85	**-0.43**	-0.32	-0.12
LwF [53]	43.03	**51.47**	51.29	72.83	76.94	**77.49**
	22.72	19.29	**19.14**	-0.43	**-0.46**	-0.36
LwM [22]	50.46	**56.83**	51.19	76.21	**79.50**	77.93
	20.36	**17.63**	20.56	0.28	0.00	**-0.06**
RWalk [14]	38.27	**45.65**	44.30	66.87	**72.95**	72.33
	24.07	**22.33**	22.74	3.59	**2.50**	3.73

Table 11. *Average accuracy* values for MNIST and different IL methods. (Best results for each case are highlighted in bold.)

	iCaRL [74]	LwF [53]	EWC [44]	SI [98]	MAS [4]	GEM [57]	DGR [80]	RTF [87]	Bi-R [85, 86]	CURL [73]
Split MNIST Task-IL	-	99.60	98.64	99.09	99.22	98.42	99.47	**99.66**	-	99.10
Split MNIST Class-IL	**94.57**	24.19	19.90	20.04	19.52	92.20	91.24	92.56	-	92.59
Permuted MNIST Class-IL	-	69.84	68.30	91.10	69.30	89.50	92.19	**96.23**	90.40	90.40

Table 12. *Average accuracy* values for MNIST and several data-free methods. (Best result highlighted in bold.)

ARM [40]	DI [96]	DFKD [56]	RTF [87]	BI-R [85, 86]
56.3	55.4	91.38	**92.6**	90.4

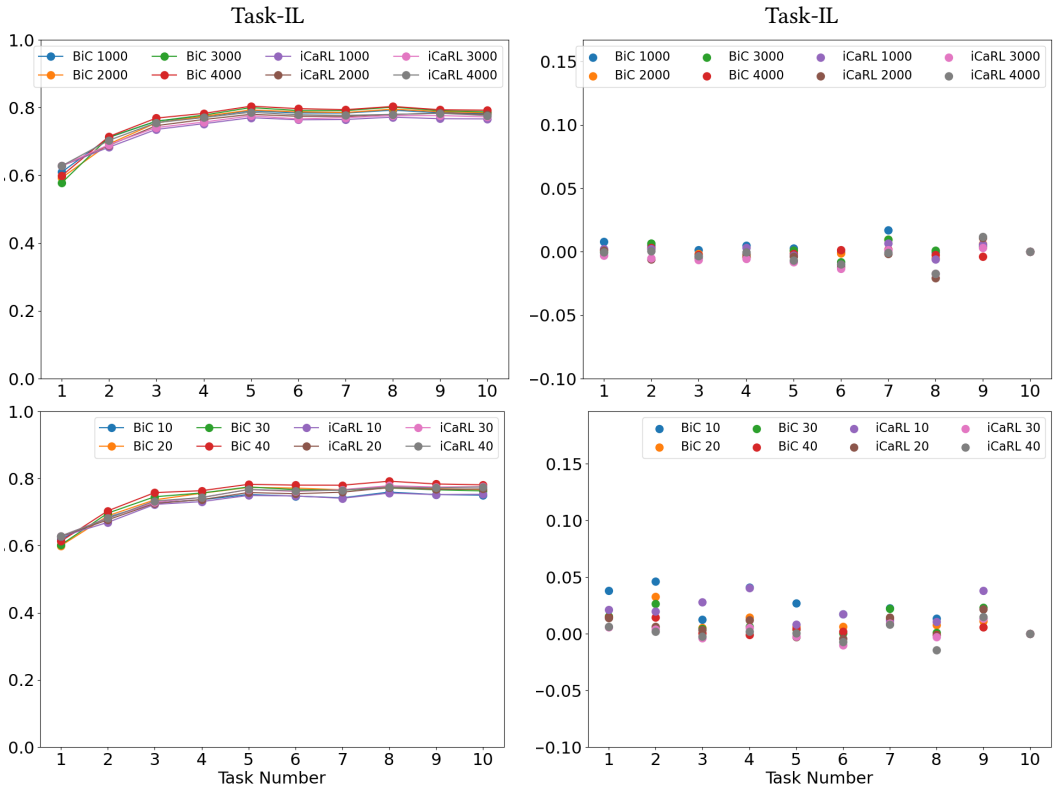

Fig. 11. *Accuracy* [left] and *forgetting* [right] for the *Core50/4* dataset and the BiC / iCaRL algorithms for different memory sizes for the Task-IL scenario: (row 1) fixed-size memory, (row 2) growing-size memory.

6 Conclusions

In this paper, we have performed an extensive survey of Incremental Learning (IL) methodologies. The works collected have been organized in nine categories: regularization-based methods, exemplar replay-based methods, variational continual learning-based methods, parameter isolation-based methods, dynamic architectures-based methods, distillation-based methods, generative methods, data-free methods and unsupervised methods. They have been briefly described and analyzed with the aim of providing more insight into the respective strengths and weaknesses. In addition, we have reported the results of a diverse set of experiments comparing the performance of a selection of the studied methods on a wide range of IL scenarios, using the datasets commonly accepted by the related research community.

The main conclusions that can be drawn from the experimental validation are:

- When no memory is available, LwF and LwM have consistently exhibited better accuracy for both the TIL and CIL scenarios. The two methods apply distillation, and LwF also makes use of regularization. Regarding forgetting, LwF also shows the lowest levels for TIL, while LwM achieves intermediates positions in this scenario. MAS, also based on regularization, has been the least affected by forgetting for the CIL case without memory.

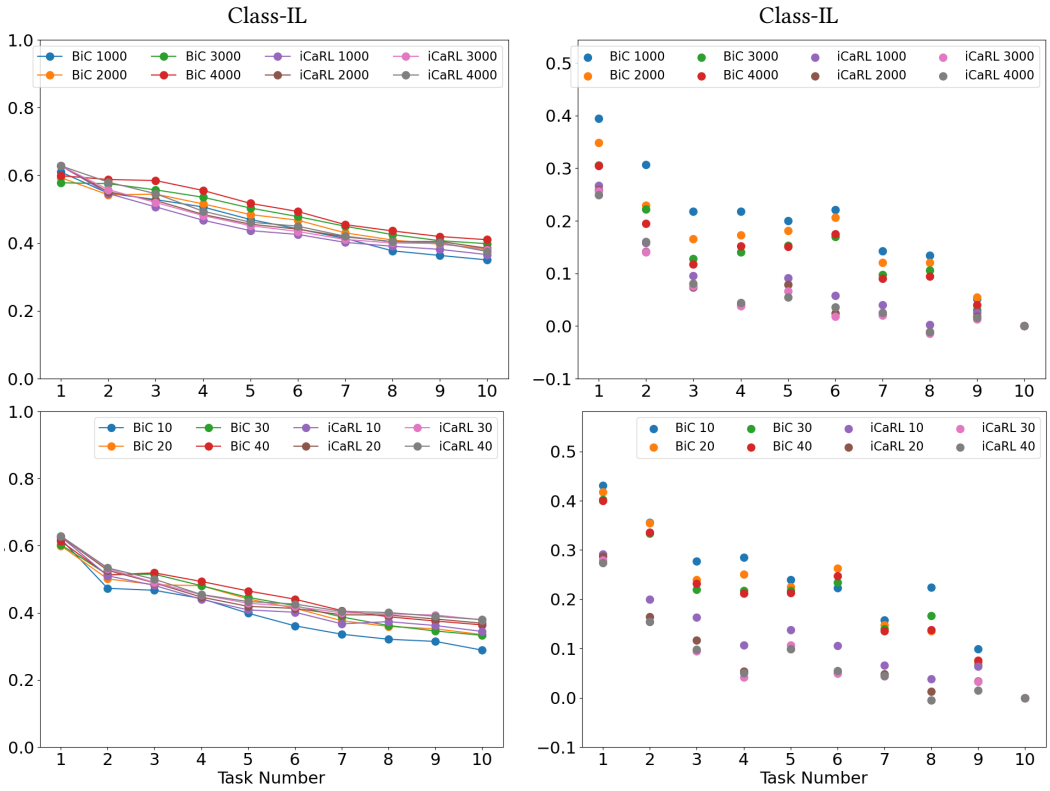

Fig. 12. *Accuracy* [left] and *forgetting* [right] for the *Core50/4* dataset and the BiC / iCaRL algorithms for different memory sizes and the Class-IL scenario: (row 1) fixed-size memory, (row 2) growing-size memory.

- When memory is available, LwF and LwM still keep at the highest positions of the ranking, particularly LwM. In this case, BiC, based on exemplar replay, outperforms the other methods on several occasions and actually gets second best; iCaRL, also based on exemplar replay, gets always very close to the best performing method. LwM, BiC and iCaRL all apply distillation. To the contrary, SI and MAS exhibit worst performance and both adopt a regularization approach. As for forgetting, the winner is, without doubt, iCaRL.
- With more memory available, both BiC and iCaRL improve in accuracy, being the difference larger for CIL than for TIL. A bigger memory also benefits from the forgetting point of view, again larger for CIL than for TIL, and higher for BiC than for iCaRL.
- When the complexity of the base network changes, the differences in behaviour do not seem to be large for any method, nor the dataset involved in the experiment. iCaRL seems to be the least sensitive for both TIL and CIL.
- The ordering and grouping of classes into tasks has not been observed to affect significantly the different methods evaluated.

Several other more global conclusions can be drawn from the performance measurements obtained through the different evaluations performed:

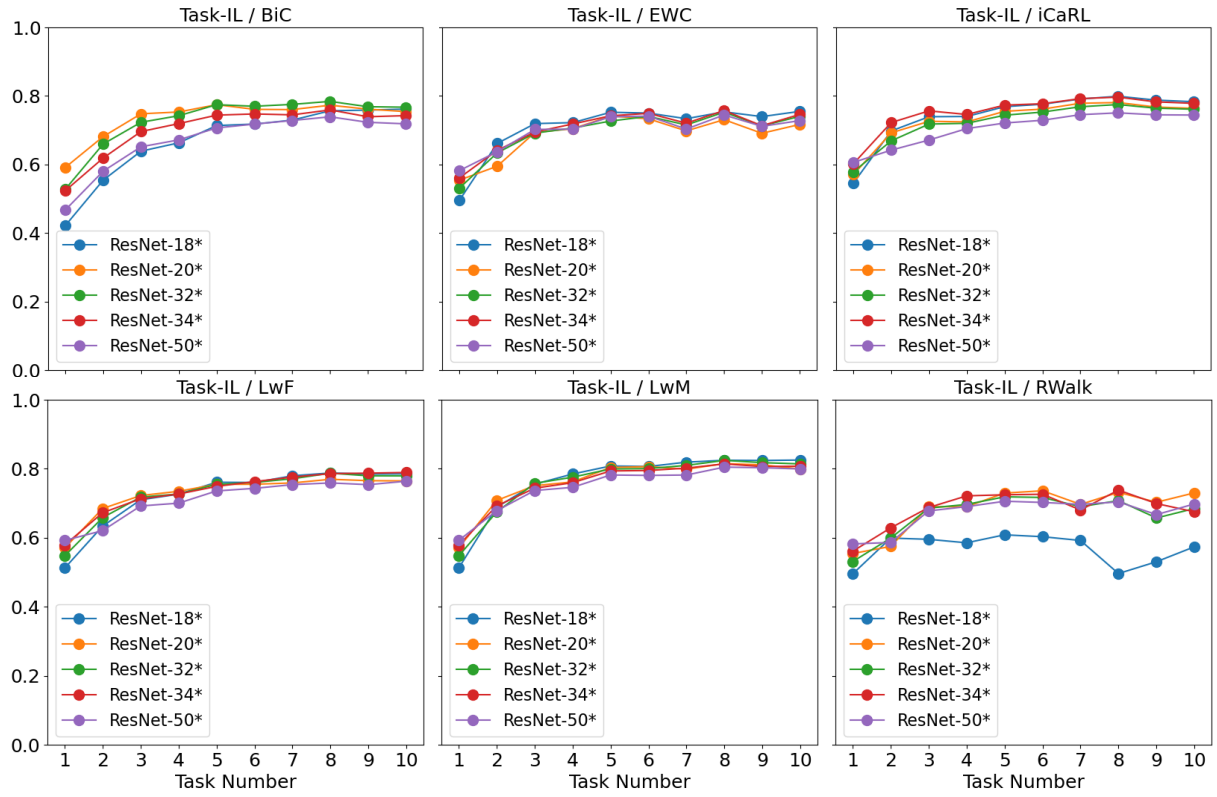

Fig. 13. *Accuracy* after learning each task for the *Core50/2* dataset for different IL algorithms and networks, and the Task-IL / fixed-size memory case (2000 exemplars).

- Most of the best performing methods use distillation in some way.
- The experimental evaluation that has been performed shows regularization as beneficial on many occasions, although, in some algorithms, it can lead to lower performance.
- When the availability of memory is not an issue, exemplar replay-based methods show advantages over other methods.
- Data-free methods seem to be a promising direction in which it may be feasible to attain higher levels of performance.
- Generally speaking, the different methods exhibit a big gap in performance between the TIL and CIL scenarios.

As a final note regarding exemplar replay, their use can be restricted in those cases where exemplars cannot be saved in memory due to data unavailability and/or privacy concerns. Because of this, data-free methods can be a good solution that allows the benefits of exemplar replay without incurring in data unavailability and privacy issues.

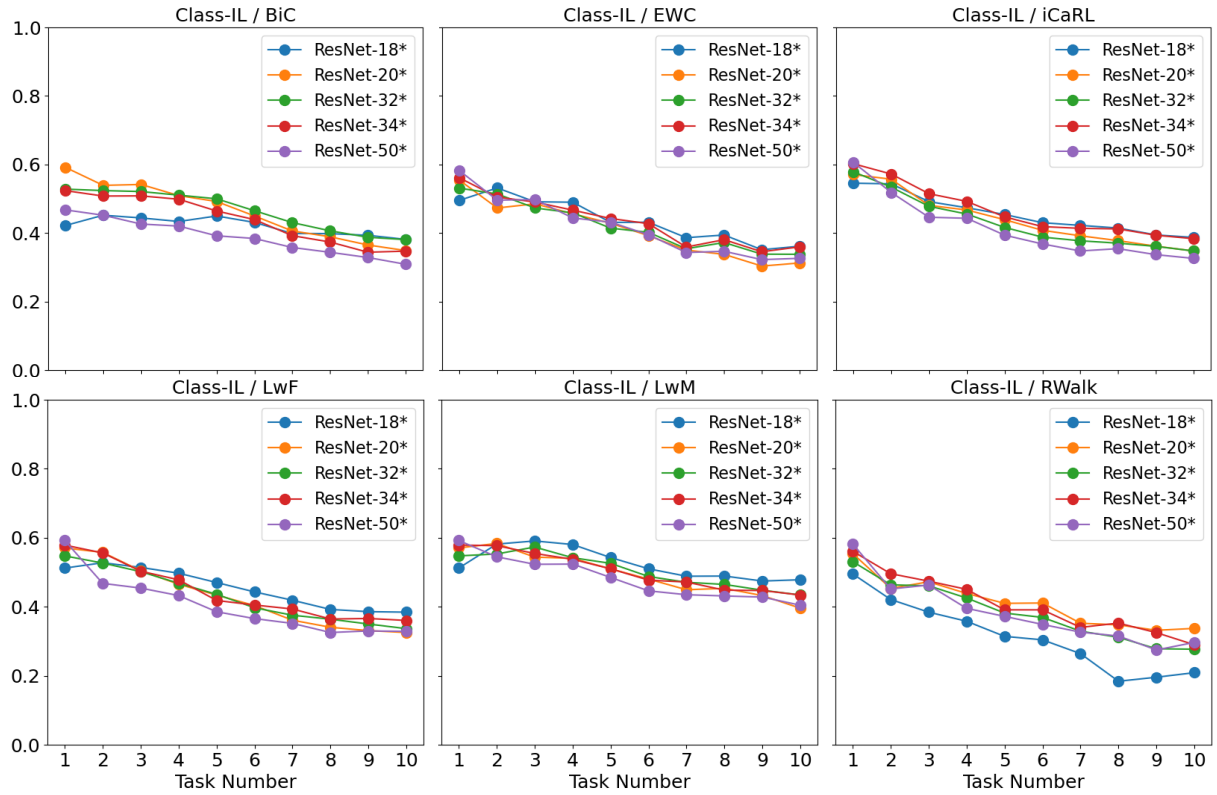

Fig. 14. *Accuracy* after learning each task for the *Core50/2* dataset for different IL algorithms and networks, and the Class-IL / fixed-size memory case (2000 exemplars).

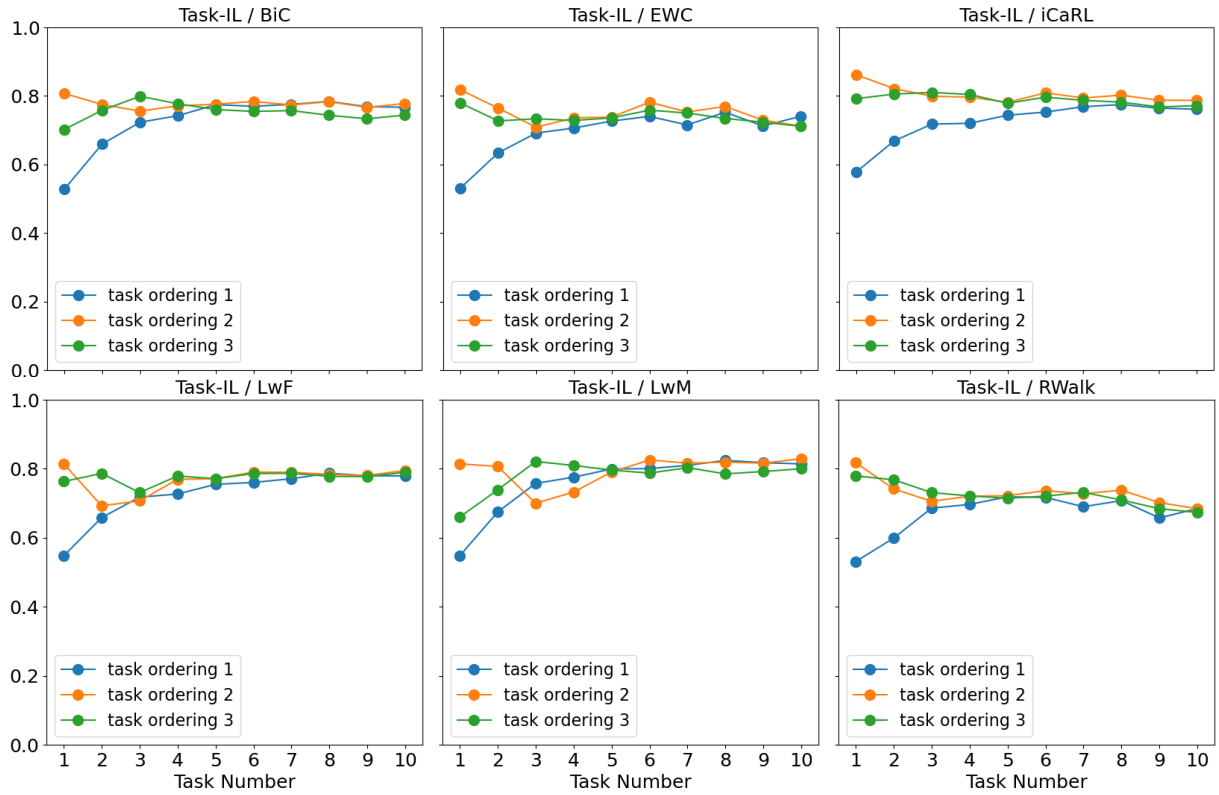

Fig. 15. *Accuracy* after learning each task for the *Core50/2* dataset for different IL algorithms and task orderings, and the Task-IL/fixed-size memory case (2000 exemplars).

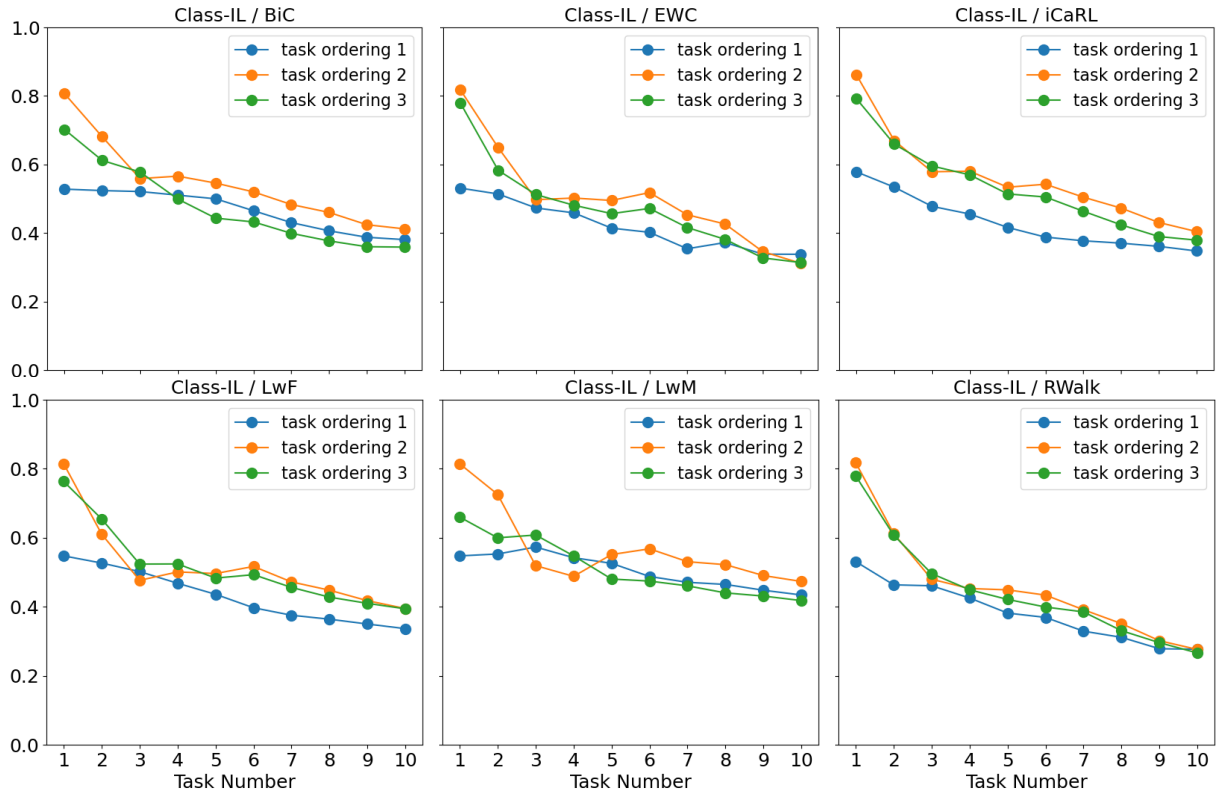

Fig. 16. *Accuracy* after learning each task for the *Core50/2* dataset for different IL algorithms and task orderings, and the Class-IL/fixed-size memory case (2000 exemplars).

Acknowledgments

For a start, we thank the anonymous referees and the associate editor for their careful reading and helpful comments.

This work has been partially supported by the EU-H2020 grant BUGWRIGHT2 (GA 871260) and by grant PID2022-139248NB-I00 (funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU). This publication reflects only the authors views and the EU is not liable for any use that may be made of the information contained therein.

The contributions of the authors to this work are as follows: conceptualization, formal analysis, investigation, software — equal; MSP — writing: original draft; AO — experimental methodology and validation; writing: draft review/editing, version after peer reviewing, and final version after acceptance; supervision and funding acquisition.

References

[1] T. Adel, H. Zhao, and R. E. Turner. 2019. Continual learning with adaptive weights (CLAW). *arXiv preprint arXiv:1911.09514*.

[2] H. Ahn, S. Cha, D. Lee, and T. Moon. 2019. Uncertainty-based continual learning with adaptive regularization. In *Proc. International Conference on Neural Information Processing Systems, article nr. 395* Article 395.

[3] I. Aleo, P. Arena, and L. Patané. 2010. Incremental learning for visual classification using neural gas. In *Proc. IEEE International Joint Conference on Neural Networks*, 1–6.

[4] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. 2018. Memory aware synapses: learning what (not) to forget. In *Proc. European Conference on Computer Vision*, 139–154.

[5] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars. 2019. Online continual learning with maximally interfered retrieval. In *Proc. International Conference on Neural Information Processing Systems, article nr. 1063* Article 1063.

[6] R. Aljundi, P. Chakravarty, and T. Tuytelaars. 2017. Expert gate: lifelong learning with a network of experts. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3366–3375.

[7] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. 2019. Gradient based sample selection for online continual learning. In *Proc. Annual Conference on Neural Information Processing Systems*, 11816–11825.

[8] C. Atkinson, B. McCane, L. Szymanski, and A. Robins. 2018. Pseudo-recursal: solving the catastrophic forgetting problem in deep neural networks. *arXiv preprint arXiv:1802.03875*.

[9] A. Ayub and A. R. Wagner. 2023. CBCL-PR: a cognitively inspired model for class-incremental learning in robotics. *IEEE Transactions on Cognitive and Developmental Systems*, 15, 4, 2004–2013.

[10] E. Belouadah and A. Popescu. 2019. Il2m: class incremental learning with dual memory. In *Proc. IEEE International Conference on Computer Vision*, 583–592.

[11] O. Beyer and P. Cimiano. 2013. DYNG: dynamic online growing neural gas for stream data classification. In *Proc. European Symposium on Artificial Neural Networks*, 497–502.

[12] K. Bhardwaj, N. Suda, and R. Marculescu. 2019. Dream distillation: a data-independent model compression framework. *arXiv preprint arXiv:1905.07072*.

[13] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. 2018. End-to-end incremental learning. In *Proc. European Conference on Computer Vision*, 233–248.

[14] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. 2018. Riemannian walk for incremental learning: understanding forgetting and intransigence. In *Proc. European Conference on Computer Vision*, 532–547.

[15] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. 2018. Efficient lifelong learning with A-GEM. *arXiv preprint arXiv:1812.00420*.

[16] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. Dokania, P. Torr, and M. Ranzato. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

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

[18] Y. Chen, T. Diethe, and N. Lawrence. 2019. Facilitating Bayesian continual learning by natural gradients and stein gradients. *arXiv preprint arXiv:1904.10644*.

[19] W. Chenshen, L. Herranz, L. Xialei, W. Yaxing, J. van de Weijer, and B. Raducanu. 2018. Memory replay gans: learning to generate images from new categories without forgetting. In *Proc. Conference on Neural Information Processing Systems*, 5966–5976.

[20] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang. 2017. AdaNet: adaptive structural learning of artificial neural networks. In *Proc. International Conference on Machine Learning*, 874–883. https://proceedings.mlr.press/v70/cortes17a.html.

[21] M. De Lange and T. Tuytelaars. 2021. Continual prototype evolution: learning online from non-stationary data streams. In *Proc. IEEE International Conference on Computer Vision*, 8250–8259.

[22] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. 2019. Learning without memorizing. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5138–5146.

[23] N. Diaz-Rodriguez, V. Lomonaco, D. Filliat, and D. Maltoni. 2018. Don't forget, there is more than forgetting: new metrics for continual learning. In *Proc. NeurIPS workshop on Continual Learning*.

[24] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach. 2019. Uncertainty-guided continual learning with Bayesian neural networks. *arXiv preprint arXiv:1906.02425*.

[25] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach. 2020. Adversarial continual learning. In *Proc. European Conference on Computer Vision*, 386–402.

[26] S. Farquhar and Y. Gal. 2019. A unifying Bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*.

[27] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. 2017. Pathnet: evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

[28] S. Furao and O. Hasegawa. 2006. An incremental network for on-line unsupervised classification and topology learning. *Neural networks*, 19, 1, 90–106.

[29] S. Furao, T. Ogura, and O. Hasegawa. 2007. An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, 20, 8, 893–903.

[30] A. Gepperth and C. Karaoguz. 2017. Incremental learning with self-organizing maps. In *Proc. International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization*. IEEE, 1–8.

[31] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

[32] D. Hadas, G. Yovel, and N. Intrator. 2011. Using unsupervised incremental learning to cope with gradual concept drift. *Connection Science*, 23, 1, 65–83.

[33] K. Han, S.-A. Rebuffi, S. Ehrhardt, A. Vedaldi, and A. Zisserman. 2020. Automatically discovering and learning new visual categories with ranking statistics. In *Proc. International Conference on Learning Representations*. https://openreview.net/forum?id=BJl2_nVFPB.

[34] K. Han, S.-A. Rebuffi, S. Ehrhardt, A. Vedaldi, and A. Zisserman. 2022. AutoNovel: automatically discovering and learning novel visual categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44, 10, 6767–6781.

[35] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 770–778.

[36] X. He and H. Jaeger. 2018. Overcoming catastrophic interference using conceptor-aided backpropagation. In *Proc. International Conference on Learning Representations*. https://openreview.net/forum?id=B1al7jg0b.

[37] G. Hinton, O. Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[38] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 831–839.

[39] D. Isele and A. Cosgun. 2018. Selective experience replay for lifelong learning. In *Proc. AAAI Conference on Artificial Intelligence* number 1. Vol. 32. DOI: 10.1609/aaai.v32i1.11595.

[40] X. Ji, J. Henriques, T. Tuytelaars, and A. Vedaldi. 2020. Automatic recall machines: internal replay, continual learning and the brain. *arXiv preprint arXiv:2006.12323*.

[41] K. J. Joseph and V. N. Balasubramanian. 2020. Meta-consolidation for continual learning. In *Proc. International Conference on Neural Information Processing Systems, article nr. 1205*.

[42] H. Jung, J. Ju, M. Jung, and J. Kim. 2016. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*.

[43] S. Khare, K. Cao, and J. Rehg. 2021. Unsupervised class-incremental learning through confusion. *arXiv preprint arXiv:2104.04450*.

[44] J. Kirkpatrick et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proc. National Academy of Sciences*, 114, 13, 3521–3526.

[45] T. Kohonen. 1990. The self-organizing map. *Proceedings of the IEEE*, 78, 9, 1464–1480.

[46] M. A. Konaté, A.-F. Yao, T. Chateau, and P. Bouges. 2023. Toward industrial use of continual learning: new metrics proposal for class incremental learning. In *Proc. IEEE International Joint Conference on Neural Networks*, 1–7. DOI: 10.1109/IJCNN54540.2023.10191657.

[47] A. Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Master's thesis. University of Toronto.

[48] F. Lavda, J. Ramapuram, M. Gregorova, and A. Kalousis. 2018. Continual classification learning using generative models. *arXiv preprint arXiv:1810.10612*.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 11, 2278–2324.

[50] K. Lee, K. Lee, J. Shin, and H. Lee. 2019. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proc. IEEE International Conference on Computer Vision*, 312–321.

[51] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *Proc. International Conference on Neural Information Processing Systems*, 4655–4665.

[52] S. Lee, J. Ha, D. Zhang, and G. Kim. 2020. A neural Dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*.

[53] Z. Li and D. Hoiem. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40, 12, 2935–2947.

[54] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov. 2018. Rotate your networks: better weight consolidation and less catastrophic forgetting. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2262–2268.

[55] V. Lomonaco and D. Maltoni. 2017. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In *Proc. Annual Conference on Robot Learning*. S. Levine, V. Vanhoucke, and K. Goldberg, (Eds.) Vol. 78. PMLR, 17–26.

[56] R. G. Lopes, S. Fenu, and T. Starner. 2017. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*.

[57] D. Lopez-Paz and M. Ranzato. 2017. Gradient episodic memory for continual learning. In *Proc. International Conference on Neural Information Processing Systems*, 6470–6479.

[58] A. Mallya, D. Davis, and S. Lazebnik. 2018. Piggyback: adapting a single network to multiple tasks by learning to mask weights. In *Proc. European Conference on Computer Vision*, 67–82.

[59] A. Mallya and S. Lazebnik. 2018. Packnet: adding multiple tasks to a single network by iterative pruning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7765–7773.

[60] D. Maltoni and V. Lomonaco. 2019. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116, 56–73.

[61] T. M. Martinetz and K. J. Schulten. 1991. A "Neural-Gas" network learns topologies. In *Proc. International Conference on Artificial Neural Networks*, 397–402.

[62] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45, 5, 5513–5533.

[63] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. 2017. Variational continual learning. *arXiv preprint arXiv:1710.10628*.

[64] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, and M. Nabi. 2019. Learning to remember: a synaptic plasticity driven framework for continual learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11321–11329.

[65] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. 2019. Continual lifelong learning with neural networks: a review. *Neural networks*, 113, 54–71.

[66] G. Petit, A. Popescu, H. Schindler, D. Picard, and B. Delezoide. 2023. Fetril: feature translation for exemplar-free class-incremental learning. In *Proc. IEEE Winter Conference on Applications of Computer Vision*, 3911–3920.

[67] A. Prabhu, P. H. Torr, and P. K. Dokania. 2020. GDumb: a simple approach that questions our progress in continual learning. In *Proc. European Conference on Computer Vision*, 524–540.

[68] B. K. Puah, L. W. Chong, Y. W. Wong, K. Begam, N. Khan, M. A. Juman, and R. K. Rajkumar. 2021. A regression unsupervised incremental learning algorithm for solar irradiance prediction. *Renewable Energy*, 164, 908–925.

[69] J. Rajasegaran, M. Hayat, S. Khan, F. S. Khan, and L. Shao. 2019. Random path selection for incremental learning. In *Proc. International Conference on Neural Information Processing Systems, article nr. 1135* Article 1135.

[70] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, and M. Shah. 2020. Itaml: an incremental task-agnostic meta-learning approach. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13588–13597.

[71] J. Ramapuram, M. Gregorova, and A. Kalousis. 2020. Lifelong generative modeling. *Neurocomputing*, 404, 381–400.

[72] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. 2017. Encoder based lifelong learning. In *Proc. IEEE International Conference on Computer Vision*, 1320–1328.

[73] D. Rao, F. Visin, A. A. Rusu, Y. W. Teh, R. Pascanu, and R. Hadsell. 2019. Continual unsupervised representation learning. In *Proc. International Conference on Neural Information Processing Systems, article nr. 687* Article 687.

[74] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. 2017. iCaRL: Incremental Classifier and Representation Learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2001–2010.

[75] A. Rosenfeld and J. K. Tsotsos. 2018. Incremental learning through deep adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 3, 651–663.

[76] O. Russakovsky et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 211–252.

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

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

[79] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *Proc. International Conference on Machine Learning*, 4548–4557.

[80] H. Shin, J. K. Lee, J. Kim, and J. Kim. 2017. Continual learning with deep generative replay. In *Proc. International Conference on Neural Information Processing Systems*, 2994–3003.

[81] D. L. Silver and R. E. Mercer. 2002. The task rehearsal method of life-long learning: overcoming impoverished data. In *Advances in Artificial Intelligence*. Springer, 90–101.

[82] J. Smith, Y.-C. Hsu, J. Balloch, Y. Shen, H. Jin, and Z. Kira. 2021. Always be dreaming: a new approach for data-free class-incremental learning. In *Proc. IEEE International Conference on Computer Vision*, 9374–9384.
[83] S. Swaroop, C. V. Nguyen, T. D. Bui, and R. E. Turner. 2019. Improving and understanding variational continual learning. *arXiv preprint arXiv:1905.02099*.
[84] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong. 2020. Few-shot class-incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12183–12192.
[85] G. M. Van de Ven, H. T. Siegelmann, and A. S. Tolias. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11, 1, 4069.
[86] G. M. Van de Ven, H. T. Siegelmann, and A. S. Tolias. 2020. Brain-like replay for continual learning with artificial neural networks. *Nature Communications*, 11, Article 4069, article nr. 4069.
[87] G. M. Van de Ven and A. S. Tolias. 2018. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*.
[88] G. M. Van de Ven and A. S. Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.
[89] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. 2016. Matching networks for one shot learning. In *Proc. Advances in Neural Information Processing Systems*. D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, (Eds.) Vol. 29. Curran Associates, Inc.
[90] Y.-X. Wang, D. Ramanan, and M. Hebert. 2017. Growing a brain: fine-tuning by increasing model capacity. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2471–2480.
[91] F.-Y. Wang, D.-W. Zhou, H.-J. Ye, and D.-C. Zhan. 2022. Foster: feature boosting and compression for class-incremental learning. In *Proc. European Conference on Computer Vision*, 398–414.
[92] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. 2019. Large scale incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 374–382.
[93] J. Xu and Z. Zhu. 2018. Reinforced continual learning. In *Proc. International Conference on Neural Information Processing Systems*, 907–916.
[94] Y. Xu, S. Furao, O. Hasegawa, and J. Zhao. 2009. An online incremental learning vector quantization. In *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1046–1053.
[95] S. Yan, J. Xie, and X. He. 2021. Der: dynamically expandable representation for class incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3014–3023.
[96] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz. 2020. Dreaming to distill: data-free knowledge transfer via deepinversion. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8715–8724.
[97] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. 2018. Lifelong learning with dynamically expandable networks. In *Proc. International Conference on Learning Representations*.
[98] F. Zenke, B. Poole, and S. Ganguli. 2017. Continual learning through synaptic intelligence. In *Proc. International Conference on Machine Learning*, 3987–3995.
[99] C. Zeno, I. Golan, E. Hoffer, and D. Soudry. 2018. Task agnostic continual learning using online variational Bayes. In *Proc. Bayesian Deep Learning Workshop - NeurIPS*. http://bayesiandeeplearning.org/2018/papers/58.pdf.
[100] C. Zeno, I. Golan, E. Hoffer, and D. Soudry. 2021. Task-agnostic continual learning using online variational bayes with fixed-point updates. *Neural Computation*, 33, 11, 3139–3177.
[101] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo. 2020. Class-incremental learning via deep model consolidation. In *Proc. IEEE Winter Conference on Applications of Computer Vision*, 1131–1140.

Received 20 February 2025; accepted 10 August 2025