Class-incremental learning: survey and performance evaluation

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, Joost van de Weijer

arXiv:2010.15277v1 [cs.LG] 28 Oct 2020

***Abstract*—For future learning systems incremental learning is desirable, because it allows for: efficient resource usage by eliminating the need to retrain from scratch at the arrival of new data; reduced memory usage by preventing or limiting the amount of data required to be stored – also important when privacy limitations are imposed; and learning that more closely resembles human learning. The main challenge for incremental learning is catastrophic forgetting, which refers to the precipitous drop in performance on previously learned tasks after learning a new one. Incremental learning of deep neural networks has seen explosive growth in recent years. Initial work focused on task incremental learning, where a task-ID is provided at inference time. Recently we have seen a shift towards class-incremental learning where the learner must classify at inference time between all classes seen in previous tasks without recourse to a task-ID. In this paper, we provide a complete survey of existing methods for incremental learning, and in particular we perform an extensive experimental evaluation on twelve class-incremental methods. We consider several new experimental scenarios, including a comparison of class-incremental methods on multiple large-scale datasets, investigation into small and large domain shifts, and comparison on various network architectures.**

I. Introduction

Incremental learning, also often referred to as *continual* or *lifelong learning*, aims to develop artificially intelligent systems that can continuously learn to address new tasks from new data while preserving knowledge learned from previously learned tasks [1], [2]. In most incremental learning (IL) scenarios, tasks are presented to a learner in a sequence of delineated *training sessions* during which only data from a single task is available for learning. After each training session, the learner should be capable of performing all previously seen tasks on unseen data. The biological inspiration for this learning model is clear, as it reflects how humans acquire and integrate new knowledge: when presented with new tasks to learn, we leverage knowledge from previous ones and integrate newly learned knowledge into previous tasks [3].

This contrasts markedly with the prevailing supervised learning paradigm in which labeled data for all tasks is jointly available during a single training session of a deep network. Incremental learners only have access to data from a single task at a time while being evaluated on all learned tasks so far. The main challenge in incremental learning is to learn from data from the current task in a way that prevents forgetting of previously learned tasks. The naive approach of finetuning, so fruitfully applied to domain transfer problems, suffers from the lack of data from previous tasks and the resulting classifier is unable to classify data from them. This drastic drop in performance on previously learned tasks is a phenomenon known as *catastrophic forgetting* [4], [5], [6]. Incremental learning aims to prevent catastrophic forgetting, while at the same time avoiding the problem of *intransigence* which inhibits adaptation to new tasks [7].

In addition to the biological motivations for study, there are numerous practical advantages to incremental learning. Incremental learners optimize computational resources better than classical supervised learning. In the classical supervised learning approach, when presented with new tasks to incorporate into the scope of an artificially intelligent system, the best (often only) option is to jointly retrain over all tasks, previous and new. However, in many scenarios data from previous tasks may not be available due to privacy considerations.

We adopt the viewpoint on continual learning first proposed along with the iCaRL approach [1] and the terminology used in [8]. In incremental learning the training is divided into a sequence of tasks, and in any training session the learner has only access to the data of the current task (optionally, some methods can consider a small amount of stored data from previous tasks). Most early methods for incremental learning considered the scenario, known as *task-incremental learning* (task-IL), in which the algorithm has access to a task-ID at inference time. This has the clear advantage that methods do not have to discriminate between classes coming from different tasks. More recently, methods have started addressing the more difficult scenario of *class-incremental learning* (class-IL),[1] where the learner does not have access to the task-ID at inference time, and therefore must be able to distinguish between all classes from all tasks. In the last few years a wide variety of methods for class-incremental learning have been proposed, and we believe the time is ripe to provide a broad overview and experimental comparison of them in one place.

In this survey we set out to identify the main challenges for class-IL, and we organize the proposed solutions in three main categories: *regularization-based* solutions that aim to minimize the impact of learning new tasks on the weights that are important for previous tasks; *exemplar-based* solutions that store a limited set of exemplars to prevent forgetting of previous tasks; and solutions that directly address the problem of *task-recency bias*, a phenomenon occurring in class-IL

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta and Joost van de Weijer are from the LAMP team at the Computer Vision Center, Barcelona, Spain (e-mail: {mmasana, xialei, btwardowski, mikel.menta, joost}@cvc.uab.es). Andrew D. Bagdanov is from the Media Integration and Communication Center, Florence, Italy (e-mail: andrew.bagdanov@unifi.it).

Manuscript submitted on July 2020.

[1] We do not refer to the scenario where each task only contains a single class, but consider adding a group of classes for each task.

methods that refers to the bias towards recently-learned tasks. In addition to an overview of progress in class-IL in recent years, we also provide an extensive experimental evaluation of existing methods. We evaluate several of the more popular regularization methods (often proposed for task-IL), and extend them with exemplars for a more fair comparison to recently developed methods. In addition, we perform extensive experiments comparing twelve methods on several scenarios and also evaluate class-IL methods on a new, more challenging multi-dataset setting. Finally, we are the first to compare these methods on a wide range of network architectures. We will provide code for our extensible class-IL evaluation framework that will ensure reproducibility of all results.

This paper is organized as follows. In Sec. II we describe related work, focusing mainly on the incremental learning literature not included in our survey. In Sec. III we define class-incremental learning and the main challenges which need to be addressed. In Sec. IV we start by defining the scope of methods we consider for our experimental evaluation based on a list of desiderata. Then we introduce the main approaches that have been proposed for class-IL. In Sec. V, we outline our experimental setup and follow with an extensive experimental evaluation in Sec. VI. In Sec. VII we discuss several emerging trends in Class-IL and then finish with conclusions in Sec. VIII.

II. Related work

In this section we broadly review related work from the literature on incremental learning. Detailed discussion of class-incremental methods specifically covered by this review is deferred until Sec. IV

Existing surveys. The problem of catastrophic forgetting has been acknowledged for many years. Already in the eighties, McCloskey and Cohen [6] showed that while human subjects suffered from gradual forgetting of previously learned tasks, algorithms trained with backpropagation suffered from catastrophic forgetting. Radcliff [9] confirmed this finding on a wider range of tasks trained with backpropagation. An excellent review on early approaches to mitigating catastrophic forgetting is by French [3]. This review also discusses how the brain prevents catastrophic forgetting and lays out possible solutions for neural network design. With the resurgence of deep learning from around 2011 [10], after breakthroughs in hardware (GPUs) and availability of large labeled datasets (like ImageNet [11]), the problem of catastrophic forgetting quickly gained renewed attention [4], [5]. This led to a surge of work in incremental learning, continual learning and lifelong learning.

This surge of new research has also resulted in recent surveys on the subject. Parisi et al. [12] provide an extensive survey on lifelong learning. This review is especially valuable because of its in-depth discussion of how biological systems address lifelong learning. They thoroughly discuss biologically-motivated solutions, such as structural plasticity, memory replay, curriculum and transfer learning. Another review [13] focuses on continual learning for robotics, and puts special effort into unifying evaluation methodologies between continual learning for robotics and non-robotics applications, with the aim of increasing cross-domain progress in continual learning. These reviews, however, do not provide an experimental performance evaluation of existing methods in the field.

Some recent surveys do include evaluation of methods. Pfulb and Gepperth [14] propose a training and evaluation paradigm for task-IL methods. Their evaluations are limited to two tasks. Lomonaco and Maltoni [15] evaluate several strategies on weight initialization: random initialization, starting from a pretrained network and starting from a pretrained but only learning the classifier. De Lange et al. [16] perform an extensive survey of task-IL with an experimental evaluation, including an analysis of model capacity, weight decay, and dropout regularization within context of task-IL. In addition, they propose a framework for continually determining the stability-plasticity trade-off of the learner – which we also apply in our evaluation. Existing surveys focus on task-IL, and to the best of our knowledge there is no survey which categorizes and broadly evaluates class-IL approaches. Given the increased attention to class-IL in recent years, we think such a survey is timely.

Task-incremental learning. As discussed in [16], most task-IL methods can be grouped into families of techniques having similar characteristics. Most regularization-based and replay-based methods can be applied to both task-IL and class-IL. Those are described more in depth in Sec. IV. Parameter isolation methods are usually applied to task-IL problems since they become computationally expensive or under-perform without access to the task-ID.

Mask-based methods reduce or completely eliminate catastrophic forgetting by applying masks to each parameter or to each layer's representations. However, by learning useful paths for each task in the network structure, the simultaneous evaluation of all learned tasks is not possible. This forces several forward passes with different masks, which makes such methods very effective for task-aware evaluation, but impractical for task-agnostic settings [16], [17]. Piggyback learns masks on network weights while training a backbone [18]. PackNet learns weights and then prunes them to generate masks [19]. HAT [20] applies attention masks on layer activations to penalize modifications to those that are important for a specific task. TFM [17] applies masks on the activations during training so that weights can be reused but not modified if they relate to already learned tasks. DAN [21] combines existing filters to learn filters for new tasks. Finally, PathNet [22] learns selective routing through the weights using evolutionary strategies.

Architecture growing methods dynamically increase the capacity of the network to reduce catastrophic forgetting. They rely on promoting a more intransigent model capable of maintaining previous task knowledge, while extending that model in order to learn new tasks. This makes some of these methods impractical when the task-ID is not known, or adds too many parameters to the network which makes them unfeasible for large numbers of tasks. EG [23] duplicates the model for each new task in order completely eliminate forgetting. PNN [24] duplicates each layer and adds lateral connections between duplicates for each task. Old weights are fixed, allowing access to that information while learning the new task. However, at each task the networks grows quadratically. To solve that issue, P&C [25] proposes duplicating the network only once to keep the number of parameters fixed, and use Elastic Weight Consolidation (EWC [5]) to mitigate forgetting. RCL [26] adaptively expands each layer of the network and uses an RNN controller to determine the number of filters to add for each task. In LtG [27], the authors propose an architecture search approach to avoid catastrophic forgetting. A network structure optimization component allows each individual layer to be reused, extended or duplicated, while a learning component finetunes parameters.

Online learning. Online methods are based on streaming frameworks where learners are allowed to observe each example only once instead of iterating over a set of examples in a training session [28]. Instances are non-i.i.d and usually have temporal correlation. Lopez-Paz [29] establishes definitions and evaluation methods for this setting and describes GEM, which uses a per-task exemplar memory to constrain gradients so that the approximated loss from previous tasks is not increased. A-GEM [30] improves on GEM in efficiency by constraining based on the average of gradients from previous class exemplars. However, the authors of [31] show that simply training on the memorized exemplars, similar to the well-established technique in reinforcement learning [32], outperforms previous results. GSS [33] performs gradient-based exemplar selection based on the GEM and A-GEM procedure to allow training without knowing the task boundaries. MIR [34] trains on the exemplar memory by selecting exemplars that will have a bigger loss increase after each training step. In [35] the memory is used to store discrete latent embeddings from a Variational Autoencoder that allows generation of previous task data for training. MER [36] combines experience replay with a modification of the meta-learning method Reptile [37] to select replay samples which minimize forgetting.

Variational continual learning. Variational continual learning is based on the Bayesian inference framework. VCL [38] proposes to merge online and Monte Carlo variational inference for neural networks yielding variational continual learning. It is general and applicable to both discriminative and generative deep models. VGL [39] introduces Variational Generative Replay, a variational inference generalization of Deep Generative Replay (DGR), which is complementary to VCL. UCL [40] proposes uncertainty-regularized continual learning based on a standard Bayesian online learning framework. It gives a fresh interpretation of the Kullback-Leibler (KL) divergence term of the variational lower bound for the Gaussian mean-field approximation case. FBCL [41] proposes to use Natural Gradients and Stein Gradients to better estimate posterior distributions over the parameters and to construct coresets using approximated posteriors. IUVCL [42] proposes a new best-practice approach to mean-field variational Bayesian neural networks. These methods normally consider only the task-aware setting. BGD [43] updates the posterior in closed form and that does not require a task-ID.

Pseudo-rehearsal methods. In order to avoid storing exemplars and privacy issues inherent in *exemplar rehearsal*, some methods learn to generate examples from previous tasks. DGR [44] generates those synthetic samples using an unconditional GAN. An auxiliary classifier is needed to assign ground truth labels to each generated sample. An improved version is proposed in MeRGAN [45], where a label-conditional GAN and replay alignment are used. DGM [46] combines the advantages of conditional GANs and synaptic plasticity using neural masking. A dynamic network expansion mechanism is introduced to ensure sufficient model capacity. Lifelong GAN [47] extends image generation without catastrophic forgetting from label-conditional to image-conditional GANs. As an alternative to exemplar rehearsal, some methods perform *feature replay* [48], [49], which need a fixed backbone network to provide good representations.

Incremental Learning beyond image classification. Shmelkov et al. [50] propose to learn object detectors incrementally. They use Fast-RCNN [51] as the network and propose distillation losses on both bounding box regression and classification outputs. Additionally, they choose to distill the region proposal with the lowest background scores, which filters out most background proposals. Hao et al. [52] extend Faster-RCNN [53] with knowledge distillation. Similarly, Michieli et al. [54] propose to distill both on the output logits and on intermediate features for incremental semantic segmentation. Recently, Cermelli et al. [55] model the background by revisiting distillation-based methods and the conventional cross entropy loss. Specifically, previous classes are seen as background for the current task and current classes are seen as background for distillation. Incremental semantic segmentation has also been applied to remote sensing [56] and medical data [57].

Catastrophic forgetting has been mainly studied in feed-forward neural networks. Only recently the impact of catastrophic forgetting in recurrent LSTM networks was studied [58]. In this work, they observe that catastrophic forgetting is even more notable in recurrent networks than feed-forward networks. This is because recurrent networks amplify small changes of the weights. To address catastrophic forgetting an expansion layer technique for RNNs was proposed in [59]. A Net2Net technique [60] was combined with gradient episodic memory in [61]. In addition, they propose a benchmark of tasks for training and evaluating models for learning sequential problems. Finally, preventing forgetting for the task of captioning was studied in [62].

III. Class-incremental learning

In this section, we define the specifics of the class-incremental learning setup we consider and discuss the main challenges that class-IL methods must address.

A. General class-incremental learning setup

Our investigation focuses on class-incremental learning scenarios in which the algorithm must learn a sequence of tasks. By *task*, we refer to a set of classes disjoint from classes in other (previous or future) tasks. In each *training session* the learner only has access to data from a single task. We optionally consider a small memory that can be used to store some exemplars from previous tasks. Tasks consist of a number of classes, and learners are allowed to process the training data of the current task multiple times during the training session. We do not consider the online learning setting used in some papers [29] in which each data sample is only seen once.

More formally, an incremental learning problem $\mathcal{T}$ consists of a sequence of n tasks:

$$\mathcal{T} = [(C^1, D^1), (C^2, D^2), \dots, (C^n, D^n)], \tag{1}$$

where each task t is represented by a set of classes $C^t = \{c^t_1, c^t_2 ..., c^t_{n^t}\}$ and training data D^t. We use N^t to represent the total number of classes in all tasks up to and including task t: $N^t = \sum_{i=1}^{t} |C^i|$. We consider class-incremental classification problems in which $D^t = \{\, (\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_{m^t}, \mathbf{y}_{m^t}) \,\}$, where $\mathbf{x}$ are input features for a training sample, and $\mathbf{y} \in \{\, 0, 1 \,\}^{N^t}$ is a one-hot ground truth label vector corresponding to $\mathbf{x}_i$. During training for task t, the learner only has access to D^t, and the tasks do not overlap in classes (i.e. $C^i \cap C^j = \varnothing$ if $i \neq j$).

Our incremental learners are deep networks parameterized by weights θ and we use $\mathbf{o}(\mathbf{x}) = h(\mathbf{x}; \theta)$ to indicate the output logits of the network on input $\mathbf{x}$. We further split the neural network in a feature extractor f with weights ϕ and linear classifier g with weights V according to $\mathbf{o}(\mathbf{x}) = g(f(\mathbf{x}; \phi); V)$. We use $\hat{\mathbf{y}} = \sigma(h(\mathbf{x}; \theta))$ to identify the network predictions, where σ indicates the softmax function. After training on task t, we evaluate the performance of the network on all classes $\bigcup_{i=1}^{t} C^i$. This contrasts with task-IL where the task-ID t is known and evaluation is only over task C^t at inference time.

Most class-IL classifiers are trained with a cross-entropy loss. When training only on data from the current task t, we can consider two cross-entropy variants. We can consider a cross-entropy over *all* classes up to the current task:

$$\mathcal{L}_c(\mathbf{x}, \mathbf{y}; \theta^t) = \sum_{k=1}^{N^t} y_k \log \frac{\exp(\mathbf{o}_k)}{\sum_{i=1}^{N^t} \exp(\mathbf{o}_i)}. \tag{2}$$

Note that in this case, since the softmax normalization is performed over *all* previously seen classes from all previous tasks, errors during training are backpropagated from all outputs – including those which do not correspond to classes belonging to the current task.

Instead, we can consider only network outputs for the classes belonging to the current task t and define the following cross-entropy loss:

$$\mathcal{L}_{c*}(\mathbf{x}, \mathbf{y}; \theta^t) = \sum_{k=1}^{|C^t|} y_{N^{t\text{-}1}+k} \log \frac{\exp(\mathbf{o}_{N^{t\text{-}1}+k})}{\sum_{i=1}^{|C^t|} \exp(\mathbf{o}_{N^{t\text{-}1}+i})} \tag{3}$$

This loss only considers the softmax-normalized predictions for classes from the *current* task. As a consequence, errors are backpropagated only from the probabilities related to these classes from task t.

When using exemplars representing data from previous tasks, it is natural to apply Eq. 2 which considers the estimated probabilities on both previous and new classes. However, in the results we will show that when no exemplars are used, Eq. 3 results in significantly less catastrophic forgetting. Interestingly, finetuning with Eq. 3 leads to a much stronger baseline than finetuning with Eq. 2, which is generally reported in literature.

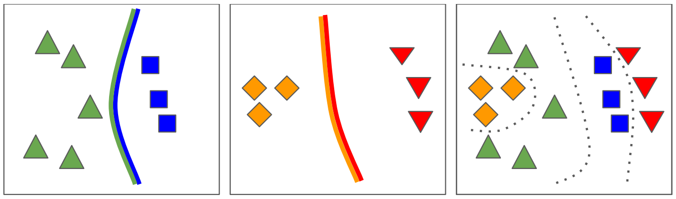


Fig. 1: A network trained continually to discriminate between task 1 (left) and task 2 (middle) is unlikely to have learned features to discriminate between the four classes (right). We call this problem *inter-task confusion*.

B. Challenges of class-incremental learning

The fundamental obstacles to effective class-incremental learning are conceptually simple, but in practice very challenging to overcome. These challenges originate from the sequential training of tasks and the requirement that at any moment the learner must be able to classify all classes from all previously learned tasks. Incremental learning methods must balance retaining knowledge from previous tasks while learning new knowledge for the current task. This problem is called the *stability-plasticity dilemma* [63]. A naive approach to class-IL which focuses solely on learning the new task will suffer from *catastrophic forgetting*: a drastic drop in the performance on previous tasks [4], [6]. Preventing catastrophic forgetting leads to a second important problem of class-IL, that of *intransigence*: the resistance to learn new tasks [7]. There are several causes of catastrophic forgetting in class-incremental learners:

- **Weight drift**: While learning new tasks, the network weights relevant to *old* tasks are updated to minimize a loss on the *new* task. As a result, performance on previous tasks suffers – often dramatically.
- **Activation drift**: Closely related to weight drift, changing weights result in changes to activations, and consequently in changes to the network output. Focusing on activations rather than on weights can be less restrictive since this allows weights to change as long as they result in minimal changes in layer activations.
- **Inter-task confusion**: in class-IL the objective is to discriminate all classes from all tasks. However, since classes are never jointly trained the network weights cannot optimally discriminate all classes (see Fig. 1). This holds for all layers in the network.
- **Task-recency bias**: Separately learned tasks might have incomparable classifier outputs. Typically, the most dominant task bias is towards more recent task classes. This effect is clearly observed in confusion matrices which illustrate the tendency to miss-classify inputs as belonging to the most recently seen task (see Fig. 2).

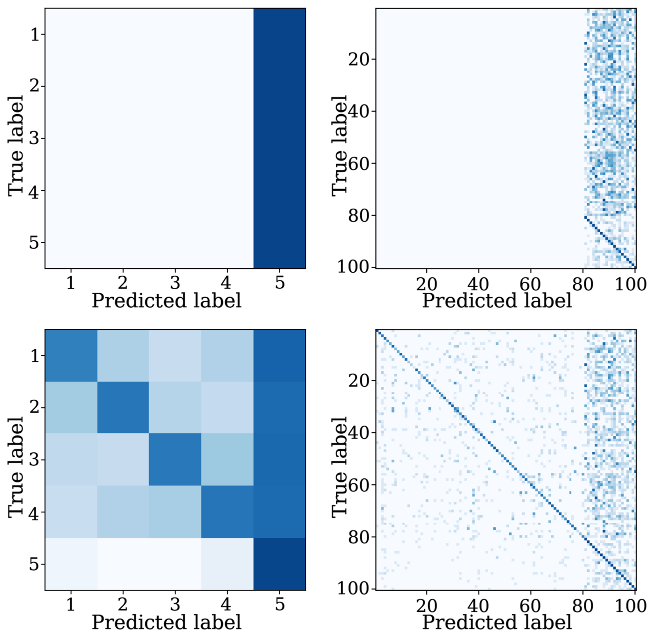


Fig. 2: Examples of task and class confusion matrices for Finetuning (top row) and Finetuning with 2,000 exemplars (bottom row) on CIFAR-100. Note the large bias towards the classes of the last task for Finetuning. By exploiting exemplars, the resulting classifier is clearly less biased.

The first two sources of forgetting are related to network drift and have been broadly considered in the task-IL literature. Regularization-based methods either focus on preventing the drift of important weights [5], [7], [64], [65] or the drift of activations [66], [67].

The last two points are specific to class-incremental learners since they have no access to a task-ID at inference time. Much of the research in class-IL has focused on reducing task imbalance [68], [69], [70], which addresses the task-recency bias. To prevent inter-task confusion and learn representations which are optimal to discriminate between all classes, rehearsal [1], [71] or pseudo-rehearsal [44], [45], [49] is commonly used.

IV. Approaches

In this section, we describe several approaches to address the above mentioned challenges of class-incremental learning. We divide them into three main categories: regularization-based methods, rehearsal-based methods, and bias-correction methods. First, we discuss the scope of this survey by articulating and motivating the properties that class-incremental learning methods should have for consideration in our experimental evaluation.

A. *Scope of our experimental evaluation*

The literature on IL is vast and growing, and several definitions and interpretations of class-IL have been proposed in recent years [1], [8], [16], [25]. In order to narrow the scope of this survey to a broad group of usefully comparable methods, we consider class-IL methods that are:

1) **Incremental**: methods that are trainable from a stream of data drawn from a non-stationary distribution.
2) **Task transferable**: class-incremental learners capable of exploiting knowledge from previous classes to improve learning of new ones (forward transfer), as well as exploiting new data to improve performance on previous tasks (backward transfer).
3) **Task-agnostic**: incremental learners able to predict classes from all previously learned tasks in a *task-agnostic* way (i.e. without recourse to an task oracle providing a subset possible classes).
4) **Offline**: methods in which data is presented in *training sessions* whose data is *i.i.d* and can be processed multiple times before moving on to the next task.
5) **Fixed network architecture**: methods using a fixed architecture for all tasks, without adding significant amount of parameters to the architecture for new tasks.
6) **Tabula rasa**: incremental learners trained from scratch which do not require pretraining on large labeled datasets. This property eliminates potential biases introduced by the class distributions seen during pretraining and any exploits derivable from that knowledge.
7) **Mature**: methods applicable to complex image classification problems.

Properties 1 and 2 are intrinsic characteristics of IL methods, and property 3 distinguishes class-incremental from task-incremental learning. While properties 4–7 are characteristics that we use to select methods for our evaluation.

Finally, we consider one additional property:

8) **Exemplar-free**: methods not requiring storage of image data from previous tasks. This is an important characteristic of methods which should be privacy-preserving (optional).

Our evaluation considers methods requiring image exemplars as well as those which do not and can therefore be applied in systems having stricter privacy considerations. Methods not requiring any data storage are seeing increased attention in a world where data privacy and security are fundamental for many users and are under increased legislative control.

B. *Regularization approaches*

Several approaches use regularization terms together with the classification loss in order to mitigate catastrophic forgetting. Some regularize on the weights and estimate an importance metric for each parameter in the network [5], [7], [64], [65], [72], [73], while others focus on the importance of remembering feature representations [66], [67], [74], [75], [76]. Most of these approaches have been developed within the context of task-IL and have been reviewed by other works [16]. Because of their importance also for class-IL, we discuss them briefly. Regularization of feature representations in particular is widely used in class-IL. Finally, we will describe several regularization techniques developed recently specifically for class-IL.

Weight regularization. The first class of approaches focuses on preventing weight drift determined to be relevant for previous tasks. They do so by estimating a prior importance of each parameter in the network (which are assumed to be independent) after learning each task. When training on new tasks, the importance of each parameter is used to penalize changes to them. That is, in addition to the cross-entropy classification loss, an additional loss is introduced:

$$\mathcal{L}_{\text{reg}}(\theta^t) = \frac{1}{2}\sum_{i=1}^{|\theta^{t\text{-}1}|} \Omega_i(\theta_i^{t\text{-}1} - \theta_i^t)^2, \tag{4}$$

where θ_i^t is weight i of the network currently being trained, $\theta_i^{t\text{-}1}$ is the value of this parameter at the end of training on task $t\text{-}1$, $|\theta^{t\text{-}1}|$ is the number of weights in the network, and Ω_i contains importance values for each network weight.

Kirkpatrick et al. [5] proposed *Elastic Weight Consolidation* (EWC) in which Ω_i is calculated as diagonal approximation of the empirical Fisher Information Matrix. However, this captures the importance of the model at the minimum after each task is learned, while ignoring the influence of those parameters along the learning trajectory in weight space. In [73], they improve EWC by rotating the parameter space to one that provides a better approximation of the Fisher Information Matrix. However, the model has to be extended with fixed parameters during training, which does not increase the capacity of the network but incurs in a computational and memory cost.

In contrast, [65] proposed the *Path Integral* approach (PathInt), that accumulates the changes in each parameter online along the entire learning trajectory. As noted by the authors, batch updates to the weights might lead to overestimating the importance, while starting from pretrained models might lead to underestimating it. To address this, *Memory Aware Synapses* (MAS) [64] also proposes to calculate Ω_i online by accumulating the sensitivity of the learned function (the magnitude of the gradient). In [7], the *Riemanian Walk* (RWalk) algorithm is proposed: both Fisher Information Matrix approximation and online path integral are fused to calculate the importance for each parameter. In addition, RWalk uses exemplars to further improve results.

Data regularization. The second class of regularization-based approaches aims to prevent activation drift and is based on knowledge distillation [77], [78] which was originally designed to learn a more compact student network from a larger teacher network. Li et al. [67] proposed to use the technique to keep the representations of previous data from drifting too much while learning new tasks. Their method, called *Learning without Forgetting* (LwF) applies the following loss:

$$\mathcal{L}_{dis}\left(\mathbf{x};\theta^t\right) = \sum_{k=1}^{N^{t\text{-}1}} \pi_k^{t\text{-}1}(\mathbf{x}) \log \pi_k^t(\mathbf{x}), \tag{5}$$

where $\pi_k(\mathbf{x})$ are temperature-scaled logits of the network:

$$\pi_k(\mathbf{x}) = \frac{e^{\mathbf{o}_k(\mathbf{x})/T}}{\sum_{l=1}^{N^{t\text{-}1}} e^{\mathbf{o}_l(\mathbf{x})/T}}, \tag{6}$$

and $\mathbf{o}(\mathbf{x})$ is the output of the network before the softmax is applied, and T is the temperature scaling parameter. We use $\pi^{t\text{-}1}$ to refer to the predictions of the network after training task $t\text{-}1$. The temperature scaling was introduced in [78] to help with the problem of having the probability of the correct class too high. Many methods use $T=2$ [67], [69], [70], [71] and we also apply that in this survey. It is important to note that when using exemplars the distillation loss is typically also applied to the exemplars of previous classes [1], [69], [70], [71].

A very similar approach, called *less-forgetting learning* (LFL), was proposed by Jung et al. [66]. LFL preserves previous knowledge by freezing the last layer and penalizing differences between the activations before the classifier layer. However, since this can introduce larger issues when the domain shift is too large, other approaches introduced modifications to deal with it. Encoder-based lifelong learning [74] extends LwF by optimizing an undercomplete autoencoder which projects features to a manifold with fewer dimensions. One autoencoder is learned per task, which makes the growth linear, although the autoencoders are small compared to the total model size.

The learning without forgetting loss in Eq. 5 was originally proposed for a task-IL setting. However, it has since been a key ingredient of many class-IL methods [1], [69], [70], [71], [76], [79]. Some works have observed that the loss works especially well when the domain shift between tasks is small (as is typically the case for class-IL), however, when domain shifts are large its efficacy drops significantly [23].

Recent developments in regularization. Several new regularization techniques have been proposed in recent work on class-IL. Zagoruyko and Komodakis [80] proposed to use the attention of the teacher network to guide the student network. *Learning without Memorizing* (LwM) [81] applies this technique to class-IL. The main idea is that the attention used by the network trained on previous tasks should not change while training the new task. Features contributing to the decision of a certain class label are expected to remain the same. This is enforced by the attention distillation loss:

$$\mathcal{L}_{AD}\left(\mathbf{x};\theta^t\right) = \left\| \frac{Q^{t-1}(\mathbf{x})}{\|Q^{t-1}(\mathbf{x})\|_2} - \frac{Q^t(\mathbf{x})}{\|Q^t(\mathbf{x})\|_2} \right\|_1, \tag{7}$$

where the attention map Q is given by:

$$Q^t(\mathbf{x}) = \text{Grad-CAM}\left(\mathbf{x}, \theta^t, c\right) \tag{8}$$

$$Q^{t\text{-}1}(\mathbf{x}) = \text{Grad-CAM}\left(\mathbf{x}, \theta^{t\text{-}1}, c\right) \tag{9}$$

and is generated with the Grad-CAM algorithm [82]. Grad-CAM computes the gradient with respect to a target class c to produce a coarse localization map indicating the image regions which contributed to the prediction. Here we cannot use the target class label, because this label did not exist when training the previous model θ^{t-1}. Instead, the authors propose to use the previous class predicted with highest probability to compute the attention maps: $c = \text{argmax h}\left(\mathbf{x};\theta^{t-1}\right)$.

Another recent method building upon LwF is *Domain Model Consolidation* (DMC) [76]. It is based on the observation that there exists an asymmetry between previous and new classes when training: new classes have explicit and strong supervision, whereas supervision for old classes is weaker and communicated by means of knowledge distillation. To remove this asymmetry, they propose to apply a *double distillation* loss on the model θ^{t-1} trained on previous classes and a newly trained model θ^t for the new classes (allowing this model to forget previous tasks):

$$\mathcal{L}_{DD}(\mathbf{u};\theta) = \frac{1}{N^t}\sum_{k=1}^{N^t}\left(\mathbf{o}_k(\mathbf{u}) - \mathring{\mathbf{o}}_k(\mathbf{u})\right)^2, \tag{10}$$

where $\mathring{\mathbf{o}}_k$ are normalized logits:

$$\mathring{\mathbf{o}}_k(\mathbf{u}) = \begin{cases} \mathbf{o}_k^{t-1}(\mathbf{u}) - \dfrac{1}{N^{t-1}}\displaystyle\sum_{l=1}^{N^{t-1}}\mathbf{o}_l^{t-1}(\mathbf{u}) & \text{if } 1 \leqslant k \leqslant N^{t-1} \\ \mathbf{o}_k^{t}(\mathbf{u}) - \dfrac{1}{N^{t}}\displaystyle\sum_{l=1}^{N^{t}}\mathbf{o}_l^{t}(\mathbf{u}) & \text{if } N^{t-1} < k \leqslant N^{t}. \end{cases} \tag{11}$$

Here $\mathbf{o}^{t-1}(\mathbf{u})$ refers to the logits from the network trained on previous tasks, and $\mathbf{o}^t(\mathbf{u})$ the ones trained on the the new classes. Because the algorithm does not have access to data of previous tasks, they propose to use auxiliary data $\mathbf{u}$, which can be any unlabeled data from a similar domain.

Finally, the *less-forget constraint* proposed by Hou et al. [69] in their method is a variant of LwF. Instead of regularizing network predictions, they propose to regularize on the cosine similarity between the L2-normalized logits of the previous and current network:

$$\mathcal{L}_{lf}(\mathbf{x};\theta) = 1 - \frac{\langle \mathbf{o}^{t-1}(\mathbf{x}), \mathbf{o}^t(\mathbf{x})\rangle}{||\mathbf{o}^{t-1}(\mathbf{x})||_2 ||\mathbf{o}^t(\mathbf{x})||_2}, \tag{12}$$

where $\langle\cdot,\cdot\rangle$ is the inner product between vectors. This regularization is less sensitive to task imbalance because the comparison is between normalized vectors. The authors show that this loss reduces bias towards new classes.

C. Rehearsal approaches

Rehearsal methods keep a small number of exemplars [1], [70], [71] (exemplar rehearsal), or generate synthetic images [44], [46] or features [48], [49] (pseudo-rehearsal). By replaying the stored or generated data from previous tasks rehearsal methods aim to prevent the forgetting of previous tasks. Most rehearsal methods combine the usage of exemplars to tackle the inter-task confusion with approaches that deal with other causes of catastrophic forgetting. The usage of exemplar rehearsal for class-IL was first proposed in *Incremental Classifier and Representation Learning* (iCaRL) [1]. This technique has since been applied in the majority of class-IL methods. In this section, we focus on the choices which need to be taken when applying exemplars.

Memory types. Exemplar memory must be extended at the end of a training session after the model has already been adapted to the new task. If the memory has a fixed maximum size across all tasks (fixed memory), some exemplars must first be removed to make space for new ones. This ensures that the memory capacity stays the same and the capacity is fixed. The more tasks and classes that are learned, the less representation each class has for rehearsal. After learning a certain amount of tasks, the memory could be expanded to better accommodate the new distributions. However, previously removed samples will be lost, thus the decision of when to expand is an important one. If the memory is allowed to grow (growing memory), then only new samples from the current task need to be added. This enforces the classes to have a stable representation during rehearsal across all tasks, at the cost of having a linear increase of memory, which might not be suitable in some applications. In both cases, the number of exemplars per class is enforced to be the same to ensure an equal representation of all classes.

Sampling strategies. The simplest way to select exemplars to add to the memory is by randomly sampling from the available data (random), which has been shown to be very effective without much computational cost [1], [7].

Inspired by [83], iCaRL proposes to select exemplars based on their corresponding feature space representations (herding). Representations are extracted for all samples and the mean for each class is calculated. The method iteratively selects exemplars for each of the classes. At each step, an exemplar is selected so that, when added to the exemplars of its class, the resulting exemplar mean is closest to the real class mean. The order in which exemplars are added is important, and taken into account when some need to be removed. Although this iterative selection procedure usually outperforms random, it increases computational cost.

In RWalk [7], two other sampling strategies are proposed. The first one calculates the entropy of the softmax outputs and selects exemplars with higher entropy (entropy). This enforces selection of samples that have more distributed scores across all classes. Similarly, the second one selects exemplars based on how close they are to the decision boundary (distance), assuming that the feature space and the decision boundaries do not change too much. For a given sample $(\mathbf{x}_i, \mathbf{y}_i)$, the pseudo-distance to the decision boundary is calculated by $f(\mathbf{x}_i;\phi)^T V_{\mathbf{y}_i}$, meaning that the smaller the distance, the closer to the decision boundary.

For these sampling strategies (except for random), the order which exemplars are chosen is recorded following a decreasing order of importance. If a fixed memory is used and some memory must be freed to make space for new exemplars, the exemplars with the lower importance are the ones removed first.

Task balancing. When applying rehearsal during the training of a new task, the weight of the new classes compared to the previous ones is defined by the trade-off between the two parts of the loss, as well as the number of samples from each class at each training step. Most approaches sample the training batches from the joint pool between new data and rehearsal exemplars [1], [7], [69], [70]. This means that batches are clearly over-represented by new samples and rely on the trade-off between the cross-entropy loss and the other losses that prevent forgetting. In contrast [71] proposes having a more balanced training where batches are equally distributed between new and previous classes. This seems to have quite beneficial effects in compensating for the task imbalance during training.

Combining rehearsal and data regularization. Several methods [1], [69], [70], [71] use the distillation loss from Learning without Forgetting [67] to deal with the activation

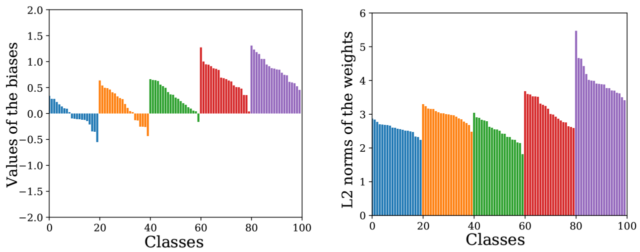


Fig. 3: Bias and weight analysis for iCaRL with 2,000 exemplars on CIFAR-100. We show the ordered biases and norm of the last classification layer of the network for different tasks. Note how the bias and the norm of the weights are higher for the last tasks. This results in a *task-recency bias*.

drift in combination with exemplars. However, Beloudah and Popescu [68] do the important observation that this distillation term actually hurts performance when using exemplars. We will confirm this in our results, however, we will show that in some scenarios a combination of weight regularization and exemplar rehearsal can be beneficial.

D. Bias-correction approaches

Bias-correction methods aim to address the problem of task-recency bias, which refers to the tendency of incrementally learned network to be biased towards classes in the most recently learned task. This is mainly caused by the fact that, at the end of training, the network has seen many examples of the classes in the last task but none (or very few in case of rehearsal) from earlier tasks. One direct consequence of this, as observed by Hou et al. [69], is that the classifier norm is larger for new classes than for the previous ones and that the classifier bias favors the more recent classes. This effect is shown in Fig. 3, where the lower biases and reduced norm of the classifier make less likely for the network to select any of the previous classes. In this section, we discuss several approaches to address this problem.

The earlier mentioned iCaRL method [1] combines exemplars and Learning without Forgetting, using a classifier layer and cross-entropy loss during training. To prevent the task-recency bias, they do not use the classifier at inference. Instead they compute the class mean of the exemplars in the feature representation, and then apply a nearest exemplar-mean for classification. Since this process is independent of the weights and biases of the final layer, the method was shown to be much less prone to the task-recency bias.

One simple yet effective approach to prevent task-recency bias has been proposed by Castro et al. [71] in their method *End-to-End Incremental Learning* (EEIL). They suggest introducing an additional stage, called *balanced training*, at the end of each training session. In this phase an equal number of exemplars from all classes is used for a limited number of iterations. To avoid forgetting the new classes, they introduce a distillation loss on the classification layer only for the classes from the current task. Balanced training could come at the cost of overfitting to the exemplars that have been stored, when these do not completely represent the distribution.

Another simple and effective approach to preventing task-recency bias was proposed by Wu et al. [70], who call their method *Bias Correction* (BiC). They add an additional layer dedicated to correcting task bias to the network. A training session is divided into two stages. During the first stage they train the new task with the cross-entropy loss and the distillation loss (see Eq. 5). Then they use a split of a very small part of the training data to serve as a validation set during the second phase. They propose to learn a linear transformation on top of the logits, $\mathbf{o}_k$, to compensate for the task-recency bias. The transformed logits are given by:

$$\mathbf{q}_k = \alpha_s \mathbf{o}_k + \beta_s, \quad c_k \in C^s \tag{13}$$

where α_s and β_s are the parameters which compensate for the bias in task s. For each task there are only two parameters which are shared for all classes in that task (initialized to $\alpha_1 = 1$ and $\beta_1 = 0$). In the second phase, all the parameters in the network are frozen, except for the parameters of the current task α_t and β_t. These are optimized with a standard softmax on the transformed logits $\mathbf{q}_k$ using the set-aside validation set. Finally, they only apply a weight decay on β parameters and not on the α parameters.

As mentioned earlier, task-recency bias was also observed by Hou et al. [69]. In their method *Learning a Unified Classifier Incrementally via Rebalancing* (LUCIR), they propose to replace the standard softmax layer σ with a cosine normalization layer according to:

$$\mathcal{L}_{cos}(\mathbf{x}; \theta^t) = \sum_{k=1}^{N^t} y_k \log \frac{\exp(\eta \langle \frac{f(\mathbf{x})}{||f(\mathbf{x})||}, \frac{V_k}{||V_k||} \rangle)}{\sum_{i=1}^{N^t} \exp(\eta \langle \frac{f(\mathbf{x})}{||f(\mathbf{x})||}, \frac{V_i}{||V_i||} \rangle)} \tag{14}$$

where $f(\mathbf{x})$ are the feature extractor outputs, $\langle \cdot, \cdot \rangle$ is the inner product, V_k are the classifier weights (also called class embedding) related to class k, and η is a learnable parameter which controls the peakiness of the probability distribution.

Hou et al. [69] also address the problem of inter-task confusion. To prevent new classes from occupying a similar location as classes from previous tasks, they apply the *margin ranking loss*. This loss pushes the current embedding away from the embeddings of the K most similar classes according to:

$$\mathcal{L}_{mr}(\mathbf{x}) = \sum_{k=1}^{K} \max\left(m - \langle \frac{f(\mathbf{x})}{||f(\mathbf{x})||}, \frac{V_y}{||V_y||} \rangle + \langle \frac{f(\mathbf{x})}{||f(\mathbf{x})||}, \frac{V_k}{||V_k||} \rangle, 0\right) \tag{15}$$

where $\hat{V}_y$ refers to the ground truth class embedding of $\mathbf{x}$, $\hat{V}_k$ refer to the embedding of the closest classes, and m is the margin.

Finally, another approach that addresses task-recency bias was proposed by Belouadah and Popescu [68] with their method called *Class-IL with dual memory* (IL2M). Their method is similar to BiC [70] in the sense that they propose to rectify the network predictions. However, where BiC learns to rectify the predictions by adding an additional layer, IL2M rectifies based on the saved certainty statistics of predictions of classes from previous tasks. Defining $m = \arg\max \hat{\mathbf{y}}(\mathbf{x})$,

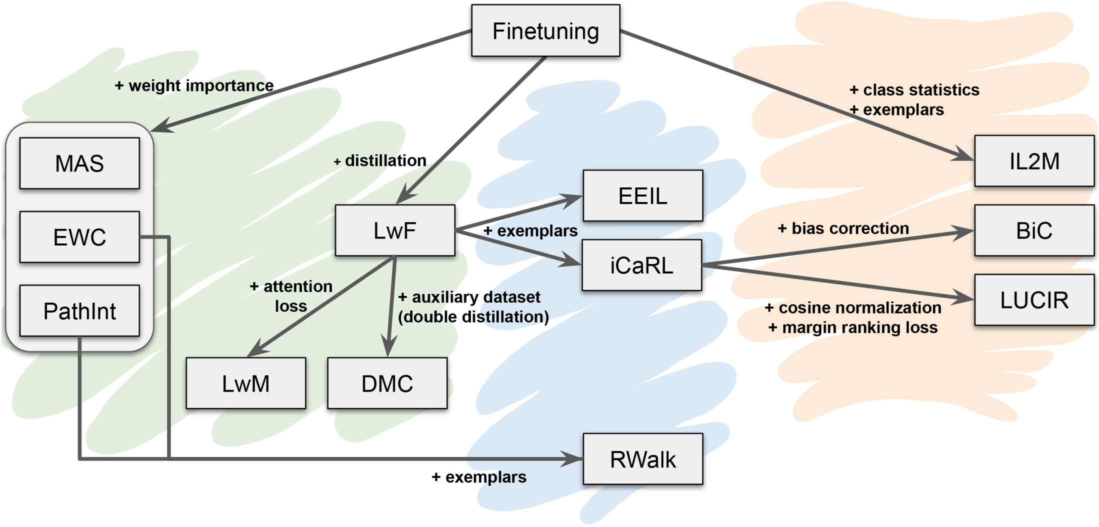


Fig. 4: Relation between class-incremental methods.

they compute the rectified predictions of the previous classes k as:

$$\hat{y}_k^r(\mathbf{x}) = \begin{cases} \hat{y}_k(\mathbf{x}) \times \frac{\overline{y}_k^p}{\overline{y}_k^t} \times \frac{\overline{y}^t}{\overline{y}^p} & \text{if } m \in C^t \\ \hat{y}_k(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{16}$$

Here $\overline{y}_k^p$ (superscript p refers to past) is the mean of the predictions $\hat{y}_k$ for all images of class c_k after training the task in which class c_k is first learned ($c_k \in C^p$), and $\overline{y}^p$ is the mean of the predictions for all classes in that task. Both $\overline{y}_k^p$ and $\overline{y}^p$ are stored directly after their corresponding training session. $\overline{y}_k^t$ is the mean of the predictions $\hat{y}_k$ for all images of class c_k after training the new task (this is computed based on the exemplars). Similarly, $\overline{y}^t$ is the mean of the predictions for all classes in the new task. As can be seen the rectification is only applied when the predicted class is a new class ($m \in C^t$). If the predicted class is an old class, the authors argue that no rectification is required since the prediction does not suffer from task-imbalance.

E. Relation between class-incremental methods

In previous sections we discussed the main approaches to mitigating catastrophic forgetting by incremental learning methods. We summarize the relations between the discussed methods in Fig. 4 starting from the naive finetuning approach. In the diagram we show all methods which we compare in Sec. VI. The diagram distinguishes between methods using exemplars to retain knowledge (blue, orange) and exemplar-free methods (green).

Most notably, the huge impact of Learning without Forgetting (LwF) [67] upon the whole field of class-incremental learning is clear. However, we expect that with the recent findings of [68], which show that when combined with exemplars finetuning can outperform LwF, could somewhat lessen its continuing influence. Weight regularization methods [5], [64], [65], applied frequently in the task-IL setting, are significantly less used for class-IL. They can also be trivially extended with exemplars and we include results of this in our experimental evaluation. Finally, Fig. 4 also shows the influence of iCaRL [1] in the development of more recent methods [69], [70].

V. Experimental setup

In this section, we explain the experimental setup and how we evaluate the approaches. We also introduce the baselines and the experimental scenarios used to gather the results presented in Sec. VI. More details on the implementation of the methods are described in Appendix A.

A. Code framework

In order to make a fair comparison between the different approaches, we implemented a versatile and extensible framework. Datasets are split into the same partitions and data is queued in the same order at the start of each task. All library calls related to randomness are synchronized and set to the same seed so that the initial conditions for all methods are the same. Data from previous tasks (excluding exemplar memory) is not available during training, thus requiring selection of any stability-plasticity-based trade-off before a training session of a task is completed (see also Sec. V-F). All methods were implemented using the original author implementations (when available) which we adapted to work under the different experimental setups proposed below (i.e. different datasets and networks).

The current version of the code includes implementations of several baselines and the following methods: EWC, MAS, PathInt, RWalk, LwM, DMC, LwF, iCaRL, EEIL, BiC, LUCIR, and IL2M. The framework includes extending most exemplar-free methods with the functionality of exemplars. The framework facilitates using these methods with a wide variety of network architectures, and allows to run the various experimental scenarios we perform in this paper. As such, our

TABLE I: Summary of datasets used.

Dataset	#Train	#Eval	#Classes
CIFAR-100 [84]	50,000	10,000	100
Oxford Flowers [85]	2,040	6,149	102
MIT Indoor Scenes [86]	5,360	1,340	67
CUB-200-2011 Birds [87]	5,994	5,794	200
Stanford Cars [88]	8,144	8,041	196
FGVC Aircraft [89]	6,667	3,333	100
Stanford Actions [90]	4,000	5,532	40
VGGFace2 [91]	491,746	50,000	1,000
ImageNet ILSVRC2012 [92]	1,280,861	50,000	1,000

framework contributes to the wider availability and comparability of existing methods, which will facilitate future research and comparisons of class-IL methods[2].

B. Datasets

We study the effects of CL methods for image classification on nine different datasets whose statistics are summarized in Table I. First, we compare the three main categories of approaches described in Sec. IV on the CIFAR-100 dataset [84]. It contains 32 × 32 colour images for 100 classes, with 600 samples for each class divided into 500 for training and 100 for testing. For data augmentation, a padding of 4 is added to each side, and crops of 32 × 32 are randomly selected during training and the center cropped is used during testing. Input normalization and random horizontal flipping are also performed.

Next, we use a variety of fine-grained classification datasets: Oxford Flowers [85], MIT Indoor Scenes [86], CUB-200-2011 Birds [87], Stanford Cars [88], FGVC Aircraft [89], and Stanford Actions [90]. These provide higher resolution and allow studying the effects on larger domain shifts when used as different tasks. To study the effects of the different approaches on smaller domain shifts we use the VGGFace2 dataset [91]. Since the original dataset has no standard splits for our setting, we keep the 1,000 classes that have the most samples and split the data following the setup from [68]. This means that this dataset is not totally balanced, but at least all used classes have a large enough pool of samples.

Finally, the ImageNet dataset [92] is used as a more realistic and large-scale scenario. It consists of 1,000 diverse object classes with different numbers of samples per class. Since this dataset takes time and needs a lot of resources, we also use the reduced ImageNet-Subset, which contains the first 100 classes from ImageNet as in [1].

In order to apply a patience learning rate schedule and an hyperparameter selection framework, an additional separate split of 10% from training is assigned to validation to those datasets that do not provide one. For all datasets except CIFAR-100, images are resized to 256 × 256 with random crops of 224 × 224 for training and center crops for testing. Input normalization and random horizontal flipping are also performed.

C. Network architectures

As suggested in [93], ResNet-32 and ResNet-18 are commonly used in the literature for CIFAR-100 and datasets with larger resolution (input sizes of around $224 \times 224 \times 3$), respectively. However, it is not so common to see other networks being used in class-IL. In this work, we also perform experiments with different networks and evaluate the effects they have on performance. Specifically, we use AlexNet [10], ResNet-18 [93], VGG-11 [94], GoogleNet [95] and MobileNets [96], [97]. We use different networks on different scenarios and make a wider comparison on ImageNet subset in Sec. VI-E). All experiments done on CIFAR-100 are trained on ResNet-32 from scratch.

We have selected the networks to represent a wide variety of network architectures commonly used in deep learning, allowing us to compare them within a continual learning setting. We have chosen AlexNet and VGG-11 as architectures which start with a number of initial convolutional layers followed by several fully connected layers. ResNets have achieved superior performance in many different computer vision tasks, and we therefore consider ResNet-18. We have also included GoogleNet which uses skip-connection and 1×1 convolutions are used as a dimension reduction module to reduce computation. We are also interested to evaluate incremental learning on compact networks. We have therefore selected MobileNet, which, to better trade off latency and accuracy, propose to replace standard convolution layers by depthwise separable convolutions. This makes them suitable to run on mobile devices.

D. Metrics

In incremental learning, $a_{t,k} \in [0, 1]$ denotes the accuracy of task k after learning task t ($k \leqslant t$), which provides fine-grained information about the incremental process. In order to compare the overall incremental learning process, the *average accuracy* is defined as $A_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}$ at task t. This measure is commonly used to compare performances of different methods with a single value. It has to be noted that when tasks have different number of classes, a weighted version needs to be used. Moreover, though *average accuracy* can be easily compared since it is a single value, it can also hide many insights. The more tasks that are learned, the more information that can be hidden. To address this, additional metrics focusing on selected key aspects of IL such as *forgetting* and *intransigence* [7] are needed.

Forgetting estimates how much the model forgot about previous task k at current task t and is defined as $f_{t,k} = \max_{i\in\{1,\dots,t-1\}} a_{i,k} - a_{t,k}$. As with accuracy, this measure can be averaged over all tasks learned so far: $F_t = \frac{1}{t-1}\sum_{i=1}^{t-1} f_i^t$. The lower the F_t is, the less forgetting is happening during incremental learning. In a similar way, *intransigence* quantifies a model's inability to learn a new task. Both can be considered complementary measures that help understand the stability-plasticity dilemma. The borderline case of a model that is never trained after first task will have no forgetting at all, but will be unable to learn new tasks. *Intransigence* for the t-th task is calculated as $I_t = a_t^* - a_{t,t}$,

[2] Code will be made available upon acceptance of this manuscript.

where a_t^* is the accuracy of a reference model for task t trained jointly on all data. The lower the $I_t \in [-1, 1]$, the better the model for that task.

To gain more insight about the classifier performance, a *confusion matrix* can be used, which gives information of miss-classification between each pair of classes. Despite the fact that it is not a single-value metric, it is often used to summarize the behavior of a classifier across many incremental tasks.

E. Baselines

Training with only a cross-entropy loss (see Eq. 2) is the default Finetuning (FT) baseline common in most IL works. This learns each task incrementally while not using any data or knowledge from previous tasks and is often used to illustrate the severity of catastrophic forgetting. However, when moving to a class-IL scenario where all previous and new classes are evaluated, other finetuning variants can be considered. We might not update the weights corresponding to the outputs of previous classes (FT+), which avoids the slow forgetting due to not having samples for those classes (see Eq. 3). As seen in Table II, this simple modification has an impact on the baseline performance. Since previous classes will not be seen, freezing the weights associated with them avoids biased modifications based only on new data. Furthermore, in the proposed scenarios approaches usually make use of an exemplar memory, which helps improve overall performance and avoid catastrophic forgetting by replaying previously seen classes. Therefore, as an additional baseline we also consider extending FT with the same exemplar memory as exemplar-based approaches (FT-E). The result of this is quite clearly more beneficial than the other FT baselines, and makes the baseline more comparable with approaches using the same memory.

In the case of Freezing (FZ), the baseline is also simple: we freeze all layers except the last one (corresponding to the classification layer or head of the network) after the first task is learned. Similarly to FT, we can also make the simple modification of not updating the weights directly responsible for the previous classes outputs (FZ+). This extends freezing to that specific group of weights which we know will not receive a gradient from previous class samples. As seen in Table II, this leads to a more robust baseline. However, if we add exemplars (FZ-E) the performance decreases. We have also observed that, when starting from a larger or more diverse first task, freezing can achieve much better performance since the learned representations before freezing are more robust.

Finally, we also use as an upper bound the joint training over all seen data (Joint). In order to have this baseline comparable over all learned tasks, we perform incremental joint training which uses all seen data at each task, starting from the model learned for the previous one. This baseline gives us an upper bound reference for all learned tasks.

F. Hyperparameter selection

For a fair comparison of IL methods, two main issues with non-IL evaluation need to be addressed. The first is that choosing the best hyperparameters for the sequence of tasks after those are learned is not a realistic scenario in that information from future tasks is used. A better comparison under an IL setting is to search for the best hyperparameters as the tasks are learned with the information at hand for each of them. Second, it makes the comparison very specific to the scenario, and in particular to the end of the specific sequence of tasks. It provides a less robust evaluation of the results over the rest of tasks, which means that other task sequence lengths are not taken into account. We feel that a broader evaluation of CL methods should include results over all tasks as if each of them were the last one for hyperparameter selection purposes.

TABLE II: Average accuracy for different baseline variants on CIFAR-100 (10/10). E denotes using 2,000 exemplars (fixed memory) or 20 exemplars per class (grow) selected with herding. All baselines start with 75.3 accuracy after task 1.

	T2	T3	T4	T5	T6	T7	T8	T9	T10
FT	33.9	27.9	19.1	17.7	12.2	11.6	10.2	9.0	7.9
FT+	39.7	32.4	25.5	20.7	16.4	14.3	12.7	11.0	9.7
FT-E (fixed)	59.4	55.2	46.6	49.1	45.9	42.3	38.2	39.5	36.5
FT-E (grow)	48.0	42.5	33.1	36.5	35.8	31.8	33.5	31.6	32.0
FZ	24.1	18.4	12.8	12.7	9.2	8.2	7.8	6.3	5.3
FZ+	40.5	31.1	24.4	24.0	21.1	18.9	17.2	16.1	14.8
FZ-E (fixed)	44.9	36.3	22.9	21.7	18.0	17.4	13.6	13.2	9.3
FZ-E (grow)	37.1	29.7	19.5	19.1	14.7	15.5	12.3	12.5	9.4

In order to provide this more robust evaluation, we use the Continual Hyperparameter Framework proposed in [16]. This framework assumes that at each task, only the data for that task is available, as in a real scenario. For each task, a *Maximal Plasticity Search* phase is used with Finetuning, and a *Stability Decay* phase is used with the corresponding method. This allows to establish a reference performance first and find the best stability-plasticity trade-off second [16] (see also Appendix A).

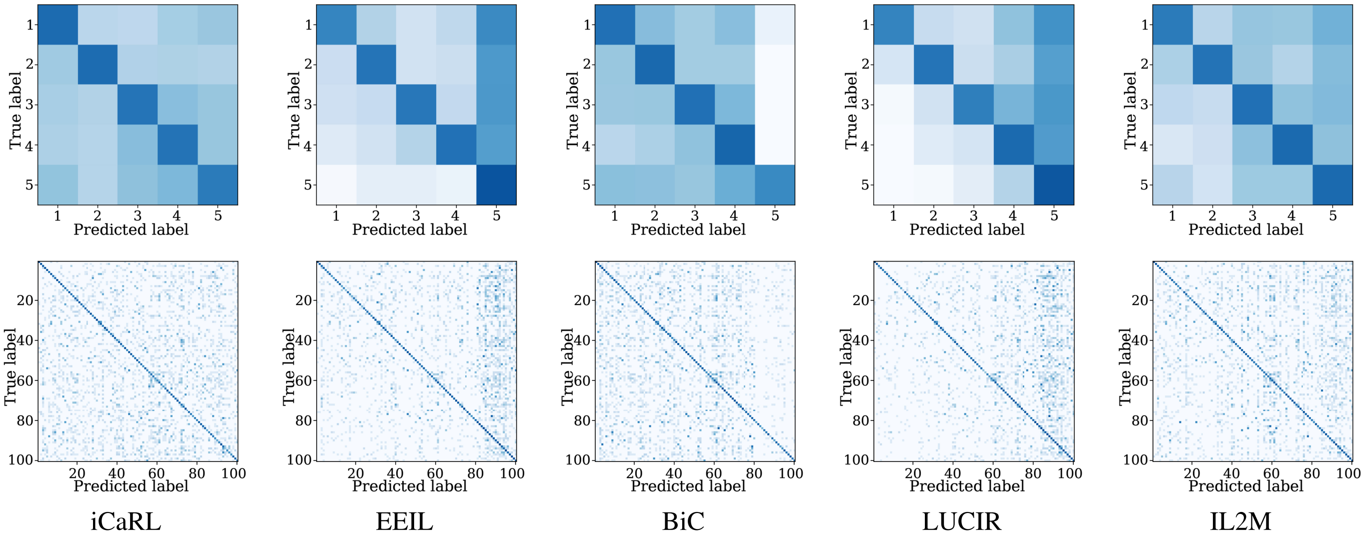
The hyperparameters that have no direct correspondence with the intransigence-forgetting duality are set to the recommended values for each of the methods. A list of those, together with the values can be found in Appendix A.

G. Experimental scenarios

To make the following results section easier to read, we define a few experimental scenarios here. We denote a dataset with B tasks and A classes on the first task as (A/B). For example, a CIFAR-100 (10/10) experiment refers to splitting the dataset into 10 tasks with the first task having 10 classes. This corresponds to an equal split among tasks and classes, making for the total amount of 100 total classes. Another setting that we use is CIFAR-100 (50/11), which means that the first task has 50 classes, and the remaining 50 classes are divided into 10 tasks of 5 classes each. Among others, these are the two main proposed settings for evaluating the different approaches and their characteristics on simpler scenarios, before moving into larger and more realistic ones.

TABLE III: Average accuracy for regularization-based methods on CIFAR-100 (10/10) on ResNet-32 trained from scratch.

	avg. acc. after	FT	LwF	EWC	PathInt	MAS	RWalk	DMC	LwM
No exemplars (task-IL)	task 2	55.9	73.2	61.6	62.0	64.8	63.2	71.5	73.6
	task 5	47.3	73.8	59.8	60.4	62.8	57.4	72.7	75.2
	task 10	37.3	65.8	56.4	56.6	53.9	46.6	67.0	69.1
No exemplars (Class-IL)	task 2	30.8	56.1	39.5	36.2	42.3	45.4	57.9	53.9
	task 5	13.1	40.9	24.0	21.8	24.4	23.9	42.2	37.3
	task 10	7.8	29.7	12.3	12.2	11.5	12.9	26.7	20.4
2,000 exemplars fixed memory (Class-IL)	task 2	59.4	59.9	56.8	56.0	56.2	56.5	-	64.7
	task 5	49.1	44.8	39.3	30.7	31.2	46.4	-	52.9
	task 10	36.5	31.8	25.8	10.5	19.1	31.4	-	38.1
20 exemplars per class growing memory (Class-IL)	task 2	48.0	51.4	43.2	42.0	41.4	41.3	-	52.1
	task 5	36.5	33.1	30.6	25.2	22.9	27.2	-	39.7
	task 10	32.0	27.4	23.9	15.8	16.8	18.9	-	33.5



iCaRL EEIL BiC LUCIR IL2M

Fig. 5: Task (top) and class (bottom) confusion matrices. CIFAR-100 (20/5) with 2,000 exemplars selected with herding.

VI. Experimental results

In this section, we discuss different aspects of continual learning approaches on CIFAR-100 being the default dataset for comparisons and we extend the experiments to larger datasets, such as VGGFace2, different fine-grained datasets and ImageNet. It includes analysis on different regularization methods with and without exemplars. Then we demonstrate how bias-correction methods perform through both task and class confusion matrices. We analyze how different aspects of exemplar usage affect performance, such as memory properties and sampling strategies. Additionally, we evaluate different methods on two popular scenarios whether the first task starts with half of classes. We also demonstrate how domain shift can result in different performance for continual learning. Next, we compare the most prominent methods on a wide range of network architectures. Then we experiment different methods on the large scale dataset ImageNet.

A. *On regularization methods*

Most of the regularization approaches have been proposed for a task-IL setting where the task-ID is known at inference time [5], [66], [67], [72], [74]. Since regularization is applied to weights or representations, they can be easily extended to a class-IL setting without much or any modification. This makes for a more challenging problem, and several more recent regularization methods already show results for class-IL [7], [76], [81]. Similarly to the baselines in Sec. V-E, when not using exemplars, methods can freeze the weights of the final layer associated with previous classes to improve performance based on the assumption that only data from new classes is used during a training session. This helps the problem of vanishing weights from learned classes and the task-recency bias, especially when using weight decay.

In Table III we compare regularization-based methods for both task-IL and class-IL. Three methods that apply data regularization (LwF, DMC, LwM) and three weight regularization methods (EWC, PathInt, MAS) are compared on CIFAR-100 (10/10). The ten tasks are learned sequentially, and each method and setting shows average accuracy at the second, fifth and final tasks to illustrate different sequence lengths. We start by comparing the regularization methods without using exemplars. Results clearly show a significant drop in performance due to the lack of the task-ID, especially after 5 and 10 tasks.

LwF obtains better results than weight-based regularization methods, which might explain why distillation has been the dominant approach for most rehearsal methods [1], [69], [70], [71].

We also expand the regularization methods with exemplars to see how it affects their performance. Note that these methods are originally proposed without exemplars, except for RWalk. In Table III we include results with a fixed memory of 2,000 exemplars, and with a growing memory of 20 exemplars per class. When using a fixed memory of exemplars, all methods improve after each task. However, that is not true in all cases for the growing memory. The reduced number of exemplars available when learning the first tasks in comparison to a fixed memory has some impact on the results. In this case, LwF outperforms EWC, PathInt and MAS, while having a similar performance than RWalk for fixed memory. Note how RWalk without exemplars does not show much improvement over other weight-based regularization methods, but that changes when a fixed memory is used. One of the most interesting results of this experiment is that LwM obtains the best results in all cases when combined with exemplars, even though the method was originally not proposed with exemplars. Furthermore, FT-E performs the second best in this scenarios, in front of LwF, as also noticed in [68]. It should be noted that in some of the next experiments we find that weight regularization and exemplars can actually achieve good results.

Finally, DMC uses a large memory based on an auxiliary dataset (300 classes from ImageNet-32, as described in [76]), we provide task-IL and class-IL results while using said extra memory, and no exemplars from the learned classes are stored. The method provides privacy-preserving properties at the cost of some performance. However, we found that in these experiments the gain obtained by distillation from an additional dataset is rather small.

Given these results, in the following experiments we will mainly compare to the best performing regularization methods, namely LwF, LwM and EWC.

B. *On bias-correction*

As seen in Fig. 2, there exists a clear bias towards recent tasks. Here we evaluate the success of class-IL methods to address the task-recency bias. To allow for a better visualization, we use a CIFAR-100 (20/5) split with ResNet-32 trained from scratch and a fixed memory of 2,000 exemplars. In the text, we will also give in brackets the average accuracy after the last task for all methods we considered.

We show the task and class confusion matrices for different bias-correction approaches in Fig. 2 and Fig. 5. The FT-E baseline, despite having improved performance due to the use of rehearsal strategies (40.9), still has a clear task-recency bias. iCaRL clearly benefits from using the NME classifier, removing most task-recency bias, although at the cost of having slightly worse performance (43.5) than the other approaches. EEIL ignores the task-recency bias during training of new tasks, however at the end of each training session it performs balanced training based only on the exemplars. This method obtains good performance (47.6), as balanced training calibrates all outputs from previous classes and thus removes a large part of the task-recency bias. BiC does a very good job at avoiding the bias while maintaining a good performance (45.7). It is clear that the newer tasks have less inter-task classification errors. However, it seems like the small pool of samples used for learning the α and β parameters (see Eq. 13) leads to having the opposite effect, and BiC appears to over-compensate toward previous tasks. LUCIR shows a more gradual task-recency bias while maintaining good performance (47.3). This could be related to the change in experimental scenario. LUCIR was shown to work better when having a larger first task followed by some smaller ones. In the more challenging setup used here their bias-correction struggles to obtain good results. Finally, IL2M clearly overcomes task-recency bias while improving on iCaRL (45.6). The class confusion matrix looks similar or better than iCaRL, but the task confusion matrix seems to point towards more inter-task miss-classifications.

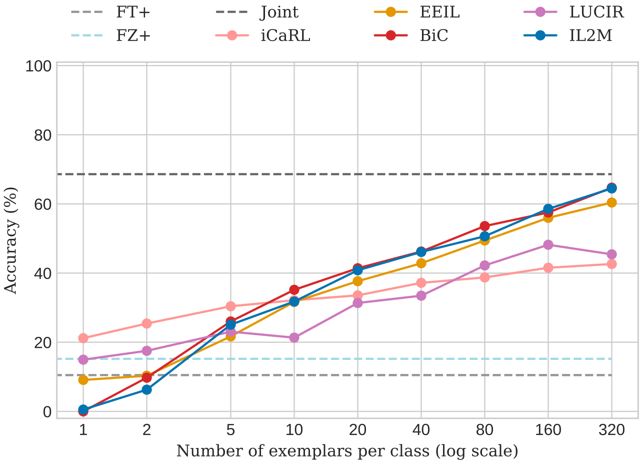


Fig. 6: Results for CIFAR-100 (10/10) on ResNet-32 trained from scratch with different exemplar memory sizes.

These results show that the two methods that have better performance (EEIL, LUCIR) still suffer from task-recency bias, while approaches that have a better solution for it (iCaRL, BiC, IL2M) still have a margin for performance improvement. This leaves room for future work to better combine or create new approaches that can both have better overall performance while simultaneously addressing the bias-correction issue.

C. *On exemplar usage*

After establishing that most methods can benefit from exemplars or use them as their main tool to avoid catastrophic forgetting, we study the effects of different characteristics related to them. The number of exemplars to store is limited by the type and amount of memory available, and exemplars are selected at the end of each training session following a sampling strategy.

On memory size: We first analyze how the number of exemplars per class affects performance as we expand the exemplar memory. In Figure 6 we compare several rehearsal methods with different numbers of exemplars per class in a

TABLE IV: CIFAR-100 (10/10) for different sampling strategies with fixed memory of 2,000 exemplars on ResNet-32.

avg. acc. after	sampling strategy	FT-E	LwF-E	EWC-E	EEIL	BiC
task 2	random	**67.3**	60.8	55.2	62.9	62.2
	herding	59.4	**61.9**	56.8	**67.2**	**62.4**
	entropy	57.9	57.8	56.4	57.7	64.2
	distance	55.5	56.5	51.0	56.0	61.9
	inv-entropy	57.6	57.6	56.4	62.6	61.7
	inv-distance	55.8	54.6	**57.4**	61.7	59.9
task 5	random	**51.3**	**48.6**	39.6	**54.7**	53.4
	herding	49.1	47.8	**41.7**	53.4	**54.9**
	entropy	38.7	36.3	33.3	45.3	43.6
	distance	41.1	38.1	27.7	45.5	42.7
	inv-entropy	40.6	41.0	36.6	47.4	45.8
	inv-distance	38.9	39.5	36.6	45.5	44.4
task 10	random	**37.1**	30.5	26.1	40.8	39.9
	herding	36.5	**30.9**	**26.8**	**42.1**	**42.9**
	entropy	21.8	18.8	14.4	28.7	29.3
	distance	20.4	16.9	10.6	27.4	25.3
	inv-entropy	29.0	25.1	22.9	31.3	34.7
	inv-distance	27.1	24.2	23.1	31.3	35.8

growing memory. As expected, in almost all cases performance increases as more exemplars are added. LUCIR and iCaRL always perform equal to or better than FT+ and FZ+. When using few exemplars per class, the weights of the last layer can be modified by large gradients coming from new classes while very little to no variability of gradients comes from previous ones. We found that the freezing of the last layer weights as used in FT+ provides a larger advantage than is obtained with only a few exemplars (see results with fewer than five exemplars for EEIL, BiC, and IL2M).

Adding more samples becomes more costly after 20 exemplars per class in comparison to the gain in performance obtained. As an example, expanding the memory from 10 to 20 samples per class on BiC yields a 6.2 point gain in average accuracy. Expanding from 20 to 40 yields a 4.8 point gain at the cost of doubling the memory size. For the other methods, these gains are similar or worse. Although starting with better performance spot with fewer exemplars per class, iCaRL has a slight slope which makes the cost of expanding the memory less beneficial. LUCIR follows with a similar curve, and both seem to be further away from Joint training (upper bound), probably due to the differences in how the classification layer is defined (NME and cosine normalization, respectively). Finally, BiC, IL2M and EEIL are quite close to Joint training when using a third of the data as memory (160 out of 500 maximum samples per class). To maintain a realistic memory budget, and given the lower performance gains from increasing said memory, we fix growing memories to use 20 exemplars per class.

On sampling strategies: As introduced in Sec. IV-C, for rehearsal approaches there are different strategies to select which exemplars to keep. In Table IV we compare the FT-E baseline, the two most common regularization-based methods (LwF-E, EWC-E), and two of the latest bias-correction methods (EEIL, BiC). We use the four different sampling strategies introduced in Sec. IV-C: random, herding (mean of features), entropy-based, and plane distance-based. We also add a variation of the last two which chooses the samples furthest away from the task boundaries to observe the effect of choosing the least confusing samples instead. We denote these as *inv-entropy* and *inv-distance*. These methods and strategies are evaluated under our two main proposed scenarios: CIFAR-100 (10/10) and (50/11)—the second one available in Appendix B.1.

Results show a clear preference across all approaches for the herding sampling strategy, except for FT-E which prefers random. The second best strategy in some cases, and generally close to herding, is random. Both these strategies clearly outperform the others when evaluating after 5 and 10 tasks in both scenarios. When only evaluating after two tasks for the (10/10) scenario, the gap between them is much smaller, probably due to the large number of exemplars available at that point (2,000). It is notable that for shorter task sequences, entropy- and distance-based perform similar to the proposed inverse versions. However, for larger sequences of tasks, the inverse versions perform better. This could be due to samples further away from the boundaries (and closer to the class centers) becoming more relevant when the number of exemplars per class becomes smaller.

On different starting scenarios: We explore two scenarios with different numbers of classes in the starting task. The first one compares a large number methods on CIFAR-100 (10/10), with classes equally split across all tasks. For the second scenario, we compare the same methods on CIFAR-100 (50/11) which is similar to having the first task being a pretrained starting point with more classes and a richer feature representation before the subsequent 10 smaller tasks are learned. Both those scenarios are further extended and presented under the two described types of memory: fixed (2,000 total exemplars) and growing (20 exemplars per class), with herding as the sampling strategy. DMC uses an external dataset (reduced ImageNet-32 [76]) that is already larger than the memories, so no exemplars are stored. All other methods which were not originally proposed with exemplars have been adapted to use them and show better performance overall than their original versions.

In Figure 7, BiC, EEIL and IL2M achieve the best results after learning 10 tasks with both fixed and growing memories. They are followed by iCaRL, LwM-E, and then LUCIR. LUCIR and BiC have different starting points on task 1 since they do not have the same initial conditions as the other approaches (LUCIR uses cosine linear layers, while BiC uses fewer data during training because it stores some for the later training of the bias-correction parameters). It is quite clear that the approaches that tackle task-recency bias have an overall better performance than the others. Furthermore, as already noted by [68], FT-E achieves competitive performance similar to the lowest performance of that family. In general, most methods seem to suffer less catastrophic forgetting when using a fixed memory that allows storing more exemplars during early tasks. For some approaches, the difference is quite considerable after learning 5 tasks and slightly better after the full 10-task sequence.

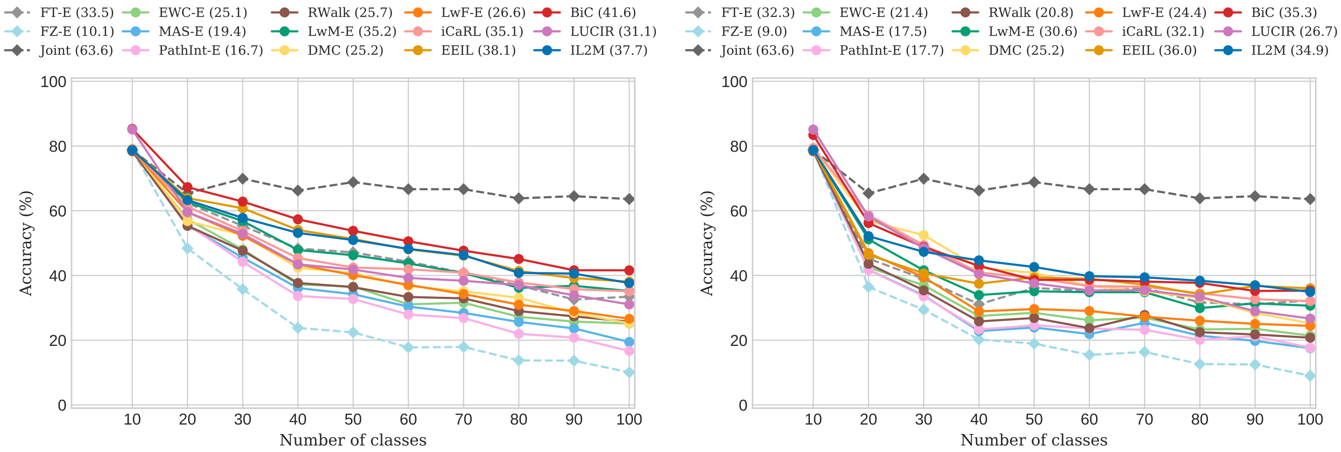


Fig. 7: CIFAR-100 (10/10) with 2,000 exemplar fixed memory (left), and 20 exemplars per class growing memory (right).

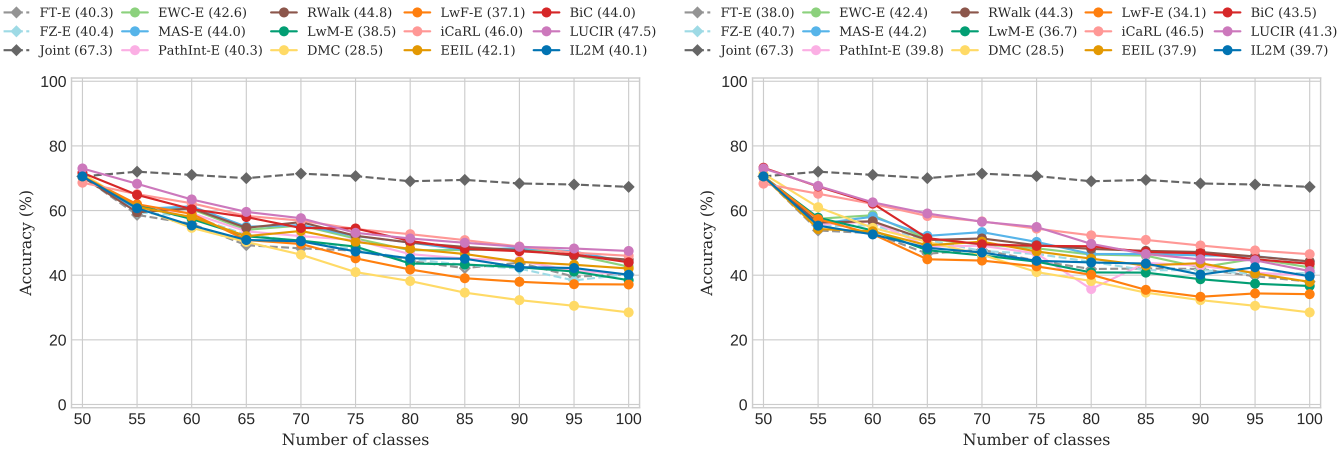


Fig. 8: CIFAR-100 (50/11) with 2,000 exemplar fixed memory (left), and 20 exemplars per class growing memory (right).

In Figure 8 the results are quite different. In general, starting from a larger number of classes makes all methods perform better, probably due to anchoring to the first task making the features already more diverse. This is specially noticeable in the case of FZ-E, which benefits significantly from freezing after a much more representative first task. This shows the importance of comparing to this baseline when doing experiments with pretrained models or a very strong first task. In this scenario, LUCIR and iCaRL have a much better performance with a fixed memory, followed by MAS-E, RWalk and BiC.

D. On domain shift effects

Up to this point all experiments were performed on a dataset with a small input size and a wide variety of classes from a similar distribution. In this experiment, we study the effects of using tasks which have different degrees of domain shifts between them and whose images also have higher resolution.

Smaller domain shift: We first conduct experiments on very small domain shifts between different classes and tasks, as is the case for VGGFace2 [91]. We divide the 1,000 classes equally into 25 tasks of 40 classes, store 5,000 exemplars in a fixed memory and train ResNet-18 from scratch. In Fig. 9 we see that LUCIR, BiC and IL2M perform the best among all methods. In particular, LUCIR achieves 73.0 average accuracy after 25 tasks, which is relatively high compared to previous experiments on CIFAR-100, which indicates that this approach might be more indicated for smaller domain shifts. Surprisingly, FT-E performs only 4.2 points lower than LUCIR and above all the other remaining approaches except for EWC-E, which also performs well with small domain shifts between tasks. EEIL shows competitive performance on the first 13 tasks, but starts to decline for the remaining ones. On the other hand, iCaRL has a larger drop in performance during early tasks, maintains the performance quite well afterwards, and ends up with similar results as EEIL and LwM-E. Of the regularization-based methods, EWC-E is superior to both LwF-E and LwM-E. FZ+ has better performance when starting from a larger first task (due to more robust feature representations), which we assumed would translate into a good performance when having small domain shifts between tasks and classes. However, the initial frozen representations are not

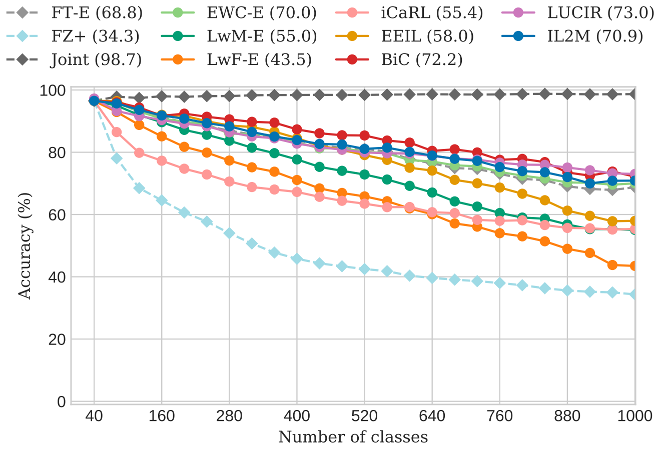


Fig. 9: Small domain shifts on VGGFace2 (40/25) on ResNet-18 trained from scratch and 5,000 exemplar fixed memory.

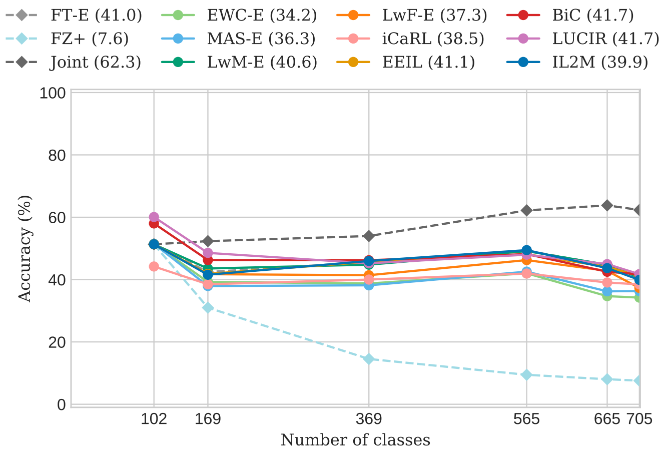


Fig. 10: Large domain shifts with multiple fine-grained datasets (Flowers, Scenes, Birds, Cars, Aircraft, Actions).

discriminative enough to generalize to new classes.

Larger domain shift: We are the first to compare class-IL methods to incrementally learn classes from various datasets. As a consequence tasks have large domain shifts and different number of classes. We use six fine-grained datasets (Flowers, Scenes, Birds, Cars, Aircraft and Actions) learned sequentially on ResNet-18 from scratch with a growing memory of 5 exemplars per class. The number of classes varies among the tasks (see Table I), but the classes inside each of them are closely related. In Fig. 10 we see that most approaches have a similar performance, much unlike previous experiments. It is noticeable that bias-correction methods do not have a clear advantage compared to other approaches. It seems that when the domain shift between tasks is large, inter-task confusion becomes the major cause for catastrophic forgetting. Solving the task-recency bias provides a lower performance advantage than in other scenarios and only improves the outputs of the corresponding task. The forgetting which is caused by the large weight and activation drift originated from the large domain shifts seems to dominate in this scenario. The fact that no method clearly outperforms the FT-E baseline shows that scenarios with large domain shifts, where catastrophic forgetting is caused by inter-task confusion, are still an important direction of study since most proposed methods focus on weight drift (EWC-E, MAS-E), activation drift (LwF-E, LwM-E, iCaRL, EEIL, BiC, LUCIR) or task-recency bias (iCaRL, BiC, LUCIR, IL2M).

Another interesting effect we visualize in Fig. 10 is the behaviour when learning Actions. The other datasets have a very clear focus on color and shape features to discriminate between their classes. However, for Actions, context is very relevant to identify which action the human is portraying in the image. Some features from the Scenes dataset can be helpful to identify an indoor or outdoor action, but in general this dataset is less related to the others. And we see that already Joint training lowers a bit the average accuracy when learning this task, as do most of the methods. Only EWC-E and MAS-E maintain or improve when learning that task, raising the question whether weight regularization-based methods have an advantage in these scenarios.

On “interspersed” domains: We propose another scenario that is not explored in class-IL: revisiting learned distributions to learn new classes. We propose to learn four fine-grained datasets split into four tasks of ten classes each for a total of 16 tasks. A group consists of four tasks, one from each dataset in this order: Flowers, Birds, Actions, Aircraft. The experiment consists of four group repetitions, where each group contains different classes (for a total of 160). This allows us to analyze how class-IL methods perform when similar tasks re-appear after learning different tasks. We refer to this scenario as “interspersed” domains since classes from each domain are distributed across tasks.

Results of forgetting on the first group during the whole sequence are presented in Fig. 11. We clearly see a difference between LUCIR and other methods. LUCIR suffers quite a large loss on the first task at the beginning of the sequence and after the second group is learned, never recovering any performance for that task. However, LUCIR shows very little forgetting for the remaining tasks in the sequence. This seems to be related to the preference of LUCIR to have a larger first task with more diverse feature representations, as also observed in earlier experiments. For the remaining methods, the first task has a lot of variation with a general decaying trend. BiC has an initial drop right after learning each of the other tasks, but manages to prevent further forgetting, though with some variability on the first Aircraft task. LwF-E and EEIL have a more cyclic pattern of forgetting and recovering. Forgetting is more pronounced when the task being learned is of the same dataset as the current one, and seems to slightly recover when learning less similar tasks. Finally, the forgetting of IL2M shows a lot of variation, which might be related to the lack of a distillation loss keeping new representations closer to previous ones.

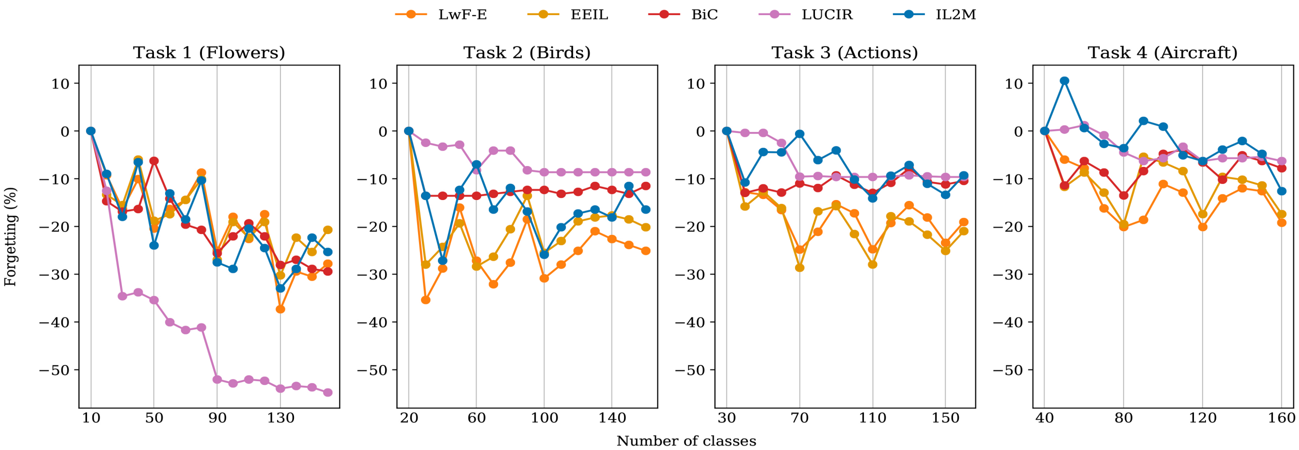


Fig. 11: Forgetting when revisiting old domains with new classes from different fine-grained datasets on AlexNet.

E. *On network architectures*

We compare the four most competitive methods over a range of different network architectures in Table V. An interesting observation is that for different networks, the performance rankings of the methods can change completely. For instance, in architectures which do not use skip connections, iCaRL performs the best when using AlexNet and VGG-11. On the other hand, BiC performs worse without skip connections, but performs the best with architectures that have them (ResNet-18, MobileNet and GoogleNet). BiC exhibits the least forgetting among all methods, even having positive forgetting which indicates that performance improves on some tasks after learning them. However, this result comes at the expense of having slightly lower performance for each task just after learning them. IL2M is more consistent compared to other methods using different networks, never having the best nor the worst performance. Networks without skip connections seem to reduce forgetting for iCaRL and IL2M. EEIL suffers more forgetting compared to others across different networks.

ResNet-18 obtains the best result among all networks with BiC. Note that in most of the literature, ResNet-18 is used as the default network for this scenario and similar ones. However, as shown above, it seems that methods benefit from architectures differently. Another interesting observation is that MobileNet, which has the lowest number of operations and can run on devices with limited capacity, has very competitive results compared to the other networks. These results show that existing IL approaches can be applied to different architectures with comparable results to the scenarios presented in the literature.

F. *On large-scale scenarios*

Finally, we compare different methods using ResNet-18 on ImageNet (40/25) with a growing memory of 20 exemplars per class. Figure 12 shows that BiC and iCaRL achieve the best performance with 32.4% and 30.2% average accuracy after 25 tasks, respectively. Surprisingly, EWC-E and FT-E outperform some methods, such as IL2M and LUCIR, in this setting. Note that in other settings, IL2M and LUCIR often perform better than EWC-E and FT-E. LwF-E and LwM-E obtain worst results compared to how they previously performed. We note that BiC, iCaRL, IL2M and LUCIR avoid a larger initial drop in performance during the first four tasks compared to other methods and continue learning without major drops in performance except for LUCIR. Of the rest of the methods, EWC-E, FT-E and EEIL seem to stabilize after the initial drop and show less forgetting as new tasks are added. RWalk, LwF-E and LwM-E continue having a larger drop in performance after task four, which only RWalk slightly recovers from. In this scenarios with a larger number of classes and more variability, methods which can easily handle early tasks will perform better afterwards. On the second half of the sequence, most approaches have the same stable behaviour since the network has learned a robust representation from the initial tasks.

TABLE V: ImageNet-Subset-100 (10/10) with different networks trained from scratch. Task accuracy when the task was learned and forgetting after learning all classes (between brackets). Final column reports the average accuracy after 10 tasks.

		task 2	task 5	task 9	A_{10}
AlexNet 60m params 2012	iCaRL	39.6 (-23.2)	30.0 (-8.4)	33.0 (-5.2)	38.8
	EEIL	27.4 (-55.0)	25.2 (-49.0)	22.6 (-49.4)	35.6
	BiC	30.6 (-31.8)	26.4 (+14.0)	21.2 (+16.8)	34.4
	IL2M	27.4 (-52.4)	21.6 (-41.2)	44.0 (-25.2)	35.2
VGG-11 133m params 2014	iCaRL	32.4 (-30.0)	34.0 (-24.8)	42.6 (-8.2)	43.2
	EEIL	29.6 (-56.0)	29.0 (-50.4)	32.8 (-45.6)	40.9
	BiC	32.4 (-33.8)	19.6 (+3.4)	31.0 (-3.2)	32.1
	IL2M	27.8 (-58.2)	31.0 (-19.6)	54.0 (-17.4)	42.2
GoogleNet 6.8m params 2014	iCaRL	35.0 (-30.0)	29.2 (-24.0)	43.6 (-12.2)	43.7
	EEIL	18.2 (-68.4)	26.0 (-49.2)	31.8 (-45.0)	36.1
	BiC	27.2 (-51.2)	39.8 (-14.2)	49.0 (-4.4)	44.5
	IL2M	23.6 (-59.0)	23.0 (-36.6)	40.0 (-36.0)	38.2
ResNet-18 11m params 2015	iCaRL	38.4 (-31.8)	29.6 (-21.8)	43.8 (-9.8)	43.6
	EEIL	26.0 (-59.4)	26.8 (-52.8)	28.2 (-48.8)	36.6
	BiC	31.2 (-48.6)	41.0 (+0.4)	49.4 (+4.4)	45.6
	IL2M	26.2 (-60.8)	24.0 (-47.8)	35.0 (-44.4)	37.2
MobileNet 4.2m params 2017	iCaRL	38.4 (-33.4)	33.6 (-21.6)	40.2 (-23.8)	43.5
	EEIL	21.2 (-68.4)	29.0 (-52.4)	25.4 (-54.8)	37.4
	BiC	39.4 (-44.2)	41.2 (-11.4)	45.2 (-14.0)	44.7
	IL2M	35.0 (-46.6)	24.2 (-24.2)	42.6 (-30.0)	42.1

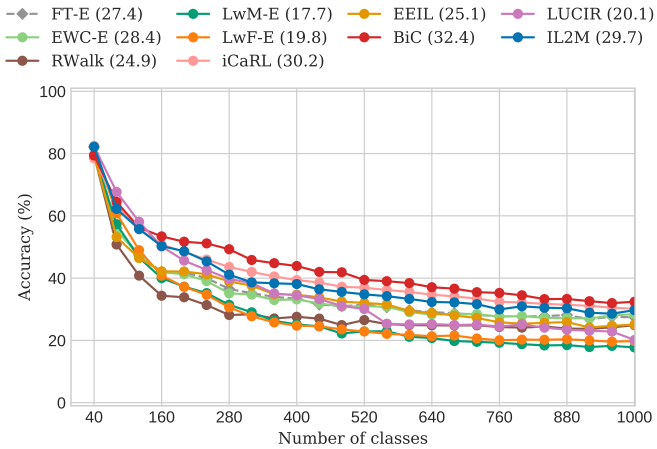


Fig. 12: ImageNet (40/25) on ResNet-18 with growing memory of 20 exemplars per class and herding sampling.

VII. Emerging trends in class-IL learning

Here we briefly discuss some recent developments in class-IL we think will play an important role in the coming years.

Exemplar learning. Recently, an exciting new direction has emerged that parametrizes exemplars and optimizes them to prevent forgetting [79], [98]. This enables much more efficient use of available storage. Liu et al. [79] propose Mnemonics Training, a method that trains the parametrized exemplars in a two-phase procedure. During the first phase, the model is optimized on the new data and exemplars of previous tasks while preventing forgetting with a distillation loss. In the second phase, the exemplars are optimized to prevent the forgetting when evaluated on the current task data (i.e. when finetuning on these exemplars the increase of the cross-entropy loss on the current data is minimal). The proposed method is combined with weight transfer [99] to reduce the number of parameters that are learned for each task. Chaudry et al. [98] generalize the theory to a streaming setting, where the learning of the exemplars does not require multiple loops over the data for every task. They propose learning a single anchor exemplar that prevents forgetting on future tasks. Because there is no access to future data, preventing forgetting on future tasks is approximated by preventing it on previous tasks (represented by exemplars).

Both methods [79], [98] show that they can outperform random and herding strategies. Optimizing the available storage by computing more efficient exemplars is expected to be one of the main future research directions, and more efficient use of limited storage in IL systems in general is expected to attract more research in the coming years.

Feature rehearsal. Pseudo-rehearsal is a good alternative to storing exemplars [44], [45], [46]. It learns a separate network that generates images of previous tasks. However, current state-of-the-art image generation methods still struggle to realistically generate complex image data, and therefore this approach has been mainly applied to relatively simple datasets and is known to obtain unsatisfying results on complex ones. To address this problem, some recent works have proposed to perform *feature* replay instead of image replay [49], [100], [101], where instead a generator is trained to generate features at some hidden layer of the network. In this way rehearsal can also be applied to complex datasets. Another closely related line of research is based on the observation that storing feature exemplars is much more compact than storing images [102]. Moving away from image replay towards different variants of feature replay is expected to gain traction.

Explicit task classification. The majority of class-IL methods incrementally learn a classifier over all classes up until those in the current task. Another approach is to learn one classifier head per task that only distinguishes between the classes within the task, and another classifier that predicts the task label. This would allow to extend any task-IL method to a class-IL method. An early version of this idea was proposed by Aljundi et al. [23] where gating autoencoders are used to predict the task label. A recent work extends a mask-based method (similar to [20]) with an explicit task classifier [103]. It is yet unclear why and when explicit task classification is expected to outperform learning one joined classifier. Further study and comparison between these two fundamentally different strategies to the problem of class-IL is needed. A recent work of Rajasegaran et al. [104] convincingly shows that for the plausible scenario where images at inference time are processed in batches with the same task-ID, this knowledge can significantly improve the quality of the explicit task classifier.

Self- and unsupervised incremental learning. Being able to incrementally learn representations from an unsupervised data stream is a desirable feature in any learning system. This direction applied to class-IL has received relatively little attention to date. Rao et al. [105] propose a method that performs explicit task classification and fits a mixture of Gaussians on the learned representations. They also explore scenarios with smooth transitions from one task to another. Still in its infancy, more research on unsupervised incremental learning is expected in coming years. In addition, leveraging the power of self-supervised representation learning [106] is only little explored within the context of IL, and is expected to gain interest.

Meta-learning. Meta-learning aims to learn new tasks leveraging information accrued while solving related tasks [107]. Riemer et al. [36] show that such a method can learn parameters that reduce interference of future gradients and improves transfer based on future gradients. Javed and White [108] explicitly learn a representation for continual learning that avoids interference and promotes future learning. These initial works have shown the potential of meta-learning on small datasets. However, we expect these techniques to be further developed in the coming years, and will start to obtain results on more complex datasets like the ones considered in our evaluation.

VIII. Conclusions

We performed an extensive survey of class-incremental learning. We organized the proposed approaches along three main lines: regularization, rehearsal, and bias-correction. In addition, we provided extensive experiments in which we compare twelve methods on a wide range of incremental learning scenarios. Here we briefly enumerate the main conclusions from these experiments:

- When comparing exemplar-free methods, LwF obtains the best results (see Table III). Among the other regularization methods, data regularization (LwM) obtains superior results compared to weight regularization (EWC and MAS). Exemplar-free methods can currently not compete with exemplar rehearsal methods, and given the more restrictive setting in which they operate, we advocate comparing them separately.
- When combining LwF with exemplars, we confirm the results in [68] showing that the added regularization does not improve results and the baseline method of finetuning with exemplars performs better (see Table III). However, using LwM for data regularization does perform better than the baseline.
- We found that in several scenarios weight regularization outperforms the baseline FT-E significantly (see Figs. 8 and 9), showing that the IL community choice of data regularization with LwF (see Fig. 4) instead of weight regularization should be reconsidered.
- Herding is a more robust exemplar sampling method than random for larger sequences of tasks, but is not better than others for short sequences (see Table IV).
- Methods that explicitly address the task-recency bias obtain better performance for class-IL (see Figs. 7, 8, 9, 10, 12): we found that BiC obtains state-of-the-art on several experiments (notably on ImageNet). IL2M obtains consistent good performance on most datasets. Also, iCaRL and EEIL obtain good performance on several datasets, but fail to outperform the baseline FT-E on others. Methods like LUCIR require a good starting representation – for example in the scenario with the larger first task or smaller domain shifts, LUCIR can be state-of-the-art.
- Current methods have mainly presented results on datasets with small domain shifts (typically random class orderings from a single dataset). When considering large domain shifts none of the methods significantly outperform the baseline FT-E (see Fig. 10). Large domain shift scenarios have been considered for task-IL, but our results show that they require new techniques to obtain satisfactory results in class-IL settings.
- We are the first to compare class-IL methods on a wide range of network architectures, showing that current class-IL works on variety of networks. Results show that most are sensitive to architecture and rankings change depending on the network used. It is quite clear that using a network with skip connections favors some methods, while their absence favors others.

Acknowledgment

We acknowledge the support from Huawei Kirin Solution. Marc Masana acknowledges grant 2019-FI_B2-00189 from Generalitat de Catalunya.

References

[1] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[2] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Advances in Neural Information Processing Systems*, 1996.

[3] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[4] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," in *International Conference on Learning Representations*, 2014.

[5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *National Academy of Sciences*, vol. 114, no. 13, 2017.

[6] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24.

[7] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *European Conference on Computer Vision*, 2018.

[8] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," in *NeurIPS Continual Learning Workshop*, 2018.

[9] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[12] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[13] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information Fusion*, vol. 58, pp. 52–68, 2020.

[14] B. Pfülb and A. Gepperth, "A comprehensive, application-oriented study of catastrophic forgetting in dnns," in *International Conference on Learning Representations*, 2019.

[15] V. Lomonaco and D. Maltoni, "Comparing incremental learning strategies for convolutional neural networks," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer, 2016, pp. 175–184.

[16] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "Continual learning: A comparative study on how to defy forgetting in classification tasks," *arXiv:1909.08383*, 2019.

[17] M. Masana, T. Tuytelaars, and J. van de Weijer, "Ternary feature masks: continual learning without any forgetting," *arXiv:2001.08714*, 2020.

[18] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *European Conference on Computer Vision*, 2018.

[19] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[20] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*, 2018.

[21] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

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

[23] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Conference on Computer Vision and Pattern Recognition*, 2017.

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

[25] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A

scalable framework for continual learning," in *International Conference on Machine Learning*, 2018.

[26] J. Xu and Z. Zhu, "Reinforced continual learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 899–908.

[27] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in *International Conference on Machine Learning*, 2019.

[28] G. I. Parisi and V. Lomonaco, "Online continual learning on sequences," in *Recent Trends in Learning From Data*. Springer, 2020, pp. 197–221.

[29] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[30] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *International Conference on Learning Representations*, 2019.

[31] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "Continual learning with tiny episodic memories," in *International Conference on Machine Learning*, 2019.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[33] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 816–11 825.

[34] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems*, 2019.

[35] M. Riemer, T. Klinger, D. Bouneffouf, and M. Franceschini, "Scalable recollections for continual lifelong learning," in *AAAI Conference on Artificial Intelligence*, 2019.

[36] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *International Conference on Learning Representations*, 2019.

[37] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv:1803.02999*, 2018.

[38] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *International Conference on Learning Representations*, 2018.

[39] S. Farquhar and Y. Gal, "A unifying bayesian view of continual learning," in *NeurIPS Deep Learning Workshop*, 2019.

[40] H. Ahn, S. Cha, D. Lee, and T. Moon, "Uncertainty-based continual learning with adaptive regularization," in *Advances in Neural Information Processing Systems*, 2019.

[41] Y. Chen, T. Diethe, and N. Lawrence, "Facilitating bayesian continual learning by natural gradients and stein gradients," *arXiv:1904.10644*, 2019.

[42] S. Swaroop, C. V. Nguyen, T. D. Bui, and R. E. Turner, "Improving and understanding variational continual learning," *arXiv:1905.02099*, 2019.

[43] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task agnostic continual learning using online variational bayes," *arXiv:1803.10123*, 2018.

[44] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[45] C. Wu, L. Herranz, X. Liu, Y. Wang, J. van de Weijer, and B. Raducanu, "Memory replay GANs: learning to generate images from new categories without forgetting," in *Advances in Neural Information Processing Systems*, 2018.

[46] O. Ostapenko, M. Puscas, T. Klein, P. Jähnichen, and M. Nabi, "Learning to remember: A synaptic plasticity driven framework for continual learning," in *Conference on Computer Vision and Pattern Recognition*, 2019.

[47] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, "Lifelong gan: Continual learning for conditional image generation," in *International Conference on Computer Vision*, 2019.

[48] R. Kemker and C. Kanan, "Fearnet: Brain-inspired model for incremental learning," in *International Conference on Learning Representations*, 2018.

[49] Y. Xiang, Y. Fu, P. Ji, and H. Huang, "Incremental learning using conditional adversarial networks," in *International Conference on Computer Vision*, 2019.

[50] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *International Conference on Computer Vision*, 2017.

[51] R. Girshick, "Fast r-cnn," in *International Conference on Computer Vision*, 2015.

[52] Y. Hao, Y. Fu, Y.-G. Jiang, and Q. Tian, "An end-to-end architecture for class-incremental object detection with knowledge distillation," in *International Conference on Multimedia and Expo*, 2019.

[53] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[54] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *International Conference on Computer Vision Workshops*, 2019.

[55] F. Cermelli, M. Mancini, S. R. Bulo, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," in *Conference on Computer Vision and Pattern Recognition*, 2020.

[56] O. Tasar, Y. Tarabalka, and P. Alliez, "Incremental learning for semantic segmentation of large-scale remote sensing data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 9, pp. 3524–3537, 2019.

[57] F. Ozdemir, P. Fuernstahl, and O. Goksel, "Learn the new, keep the old: Extending pretrained models with new anatomy and images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018.

[58] M. Schak and A. Gepperth, "A study on catastrophic forgetting in deep lstm networks," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 714–728.

[59] R. Coop and I. Arel, "Mitigation of catastrophic forgetting in recurrent neural networks using a fixed expansion layer," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–7.

[60] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," in *International Conference on Learning Representations*, 2016.

[61] S. Sodhani, S. Chandar, and Y. Bengio, "Toward training recurrent neural networks for lifelong learning," *Neural computation*, vol. 32, no. 1, pp. 1–35, 2020.

[62] R. Del Chiaro, B. Twardowski, A. D. Bagdanov, and J. Van de Weijer, "Ratt: Recurrent attention to transient tasks for continual image captioning," in *ICML Workshop LifelongML*, 2020.

[63] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in psychology*, 2013.

[64] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *European Conference on Computer Vision*, 2018.

[65] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*, 2017.

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

[67] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, 2017.

[68] E. Belouadah and A. Popescu, "Il2m: Class incremental learning with dual memory," in *International Conference on Computer Vision*, 2019.

[69] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *International Conference on Computer Vision*, 2019.

[70] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *International Conference on Computer Vision*, 2019.

[71] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *European Conference on Computer Vision*, 2018.

[72] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in Neural Information Processing Systems*, 2017.

[73] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov, "Rotate your networks: Better weight consolidation and less catastrophic forgetting," in *International Conference on Pattern Recognition*, 2018.

[74] A. Rannen, R. Aljundi, and M. B. B. T. Tuytelaars, "Encoder based lifelong learning," in *Conference on Computer Vision and Pattern Recognition*, 2017.

[75] D. L. Silver and R. E. Mercer, "The task rehearsal method of life-long learning: Overcoming impoverished data," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2002, pp. 90–101.

[76] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," in *Winter Conference on Applications of Computer Vision*, 2020.

[77] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *International Conference on Knowledge Discovery and Data Mining*, 2006.

[78] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[79] Y. Liu, A.-A. Liu, Y. Su, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *Conference on Computer Vision and Pattern Recognition*, 2020.

[80] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *International Conference on Learning Representations*, 2017.

[81] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Conference on Computer Vision and Pattern Recognition*, 2019.

[82] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *International Conference on Computer Vision*, 2017.

[83] M. Welling, "Herding dynamical weights to learn," in *International Conference on Machine Learning*, 2009.

[84] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[85] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[86] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[87] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[88] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *International Conference on Computer Vision Workshops*, 2013.

[89] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Tech. Rep., 2013.

[90] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, "Human action recognition by learning bases of action attributes and parts," in *International Conference on Computer Vision*, 2011.

[91] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *International Conference on Automatic Face & Gesture Recognition*, 2018.

[92] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[93] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition*, 2016.

[94] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[95] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Conference on Computer Vision and Pattern Recognition*, 2015.

[96] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[97] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[98] A. Chaudhry, A. Gordo, P. K. Dokania, P. Torr, and D. Lopez-Paz, "Using hindsight to anchor past knowledge in continual learning," *arXiv:2002.08165*, 2020.

[99] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[100] X. Liu, C. Wu, M. Menta, L. Herranz, B. Raducanu, A. D. Bagdanov, S. Jui, and J. van de Weijer, "Generative feature replay for class-incremental learning," in *Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[101] A. Iscen, J. Zhang, S. Lazebnik, and C. Schmid, "Memory-efficient incremental learning through feature adaptation," in *European Conference on Computer Vision*, 2020.

[102] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *European Conference on Computer Vision*, 2019.

[103] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, "Conditional channel gated networks for task-aware continual learning," in *Conference on Computer Vision and Pattern Recognition*, 2020.

[104] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, and M. Shah, "iTAML: An incremental task-agnostic meta-learning approach," in *Conference on Computer Vision and Pattern Recognition*, 2020.

[105] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," in *Advances in Neural Information Processing Systems*, 2019.

[106] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[107] J. Schmidhuber, "Evolutionary principles in self-referential learning," *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, vol. 1, no. 2, 1987.

[108] K. Javed and M. White, "Meta-learning representations for continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 1820–1830.

[109] Stanford. (CS231N) Tiny imagenet challenge, cs231n course. [Online]. Available: https://tiny-imagenet.herokuapp.com/

[110] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[111] M. Masana, B. Twardowski, and J. Van de Weijer, "On class orderings for incremental learning," in *ICML Workshop on Continual Learning*, 2020.

[112] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International Conference on Machine Learning*, 2009.

APPENDIX A
IMPLEMENTATION AND HYPERPARAMETERS

As described in Section 5.6, the Continual Hyperparameter Framework (CHF) [16] is used for the stability-plasticity trade-off hyperparameters that are associated to intransigence and forgetting when learning a new task. The CHF first performs a learning rate (LR) search with Finetuning on the new task. This corresponds to the *Maximal Plasticity Search* phase.

The LR search is limited to {5e-1, 1e-1, 5e-2} on the first task since all experiments are trained from scratch. For the remaining tasks, the LR search is limited to the three values immediately lower than the one chosen for the first task from this set: {1e-1, 5e-2, 1e-2, 5e-3, 1e-3}. We use a patience scheme as a LR scheduler where the patience is fixed to 10, the LR factor to 3 (LR is divided by it each time the patience is exhausted), and the stopping criteria is either having a LR below 1e-4 or if 200 epochs have passed (100 for VGGFace2 and ImageNet). We also do gradient clipping at 10,000, which is mostly negligible for most training sessions except the first one. We use SGD with momentum set to 0.9 and weight decay fixed to 0.0002. Batch size is 128 for most experiments except 32 for fine-grained datasets and 256 for ImageNet and VGGFace2. All code is implemented using Pytorch.

Once the shared hyperparameters are searched, the best ones are fixed and the accuracy for the first phase is stored as a reference. The hyperparameter directly related to the stability-plasticity trade-off is set to a high value which represents a heavy intransigence to learn the new task, close to freezing the network so that knowledge is preserved. At each search step, the performance is evaluated on the current task and compared to the reference accuracy from the *Maximal Plasticity Search* phase. If the method accuracy is above the 80% of the reference accuracy, we keep the model and trade-off as the ones for that task. If the accuracy is below the threshold, the trade-off is reduced in half and the search continues. As the trade-off advances through the search, it becomes less intransigence and slowly converges towards higher forgetting, which ultimately would correspond to the Finetuning of the previous phase. This corresponds to the *Stability Decay* phase.

The methods have the following implementations:

- **LwF**: we implement the $\mathcal{L}_{dis}$ distillation loss following Eqs. 5-6, and fix the temperature scaling parameter to $T = 2$ as proposed in the original work (and used in most of the literature). This loss is combined with the $\mathcal{L}_c$ cross-entropy loss from Eqs. 2-3 with a trade-off that is chosen using the CHF and starts with a value of 10. In our implementation we choose to duplicate the older model for training to evaluate the representations (instead of saving them at the end of the previous session) to benefit from the data augmentation. That older model can be removed after the training session to avoid overhead storage.
- **EWC**: the fusion of the old and new importance weights is done with $\alpha = 0.5$ to avoid the storage of the importance weights for each task. The Fisher Information Matrix is calculated by using all samples from the current task based on the predicted class. The loss introduced in Eq. 4 is combined with the $\mathcal{L}_c$ cross-entropy loss with a trade-off chosen using the CHF and with a starting value of 10,000.
- **PathInt**: we fix the damping parameter to 0.1 as proposed in the original work. As in LwF and EWC, the trade-off between the quadratic surrogate loss and the cross-entropy loss is chosen using the CHF with a starting value of 1.
- **MAS**: we implement MAS in the same way as EWC, with $\alpha = 0.5$ and the same Fisher Information Matrix setting. The trade-off between the importance weights penalty and the cross-entropy loss is chosen using the CHF and a starting value of 400.
- **RWalk**: since it is a fusion of EWC and PathInt, the same parameters $\alpha = 0.5$, Fisher Information Matrix setting and damping $= 0.1$ are fixed.The starting value for the CHF on the trade-off between their proposed objective loss and the cross-entropy loss is 10.
- **DMC**: we implement the $\mathcal{L}_{DD}$ double distillation loss from Eqs. 10-11. We set the auxiliary dataset batch size to 128, and the student is neither initialized from the previous tasks or new task models but random, as proposed in the original work.
- **LwM**: We combine the cross-entropy loss with the distillation loss and $\mathcal{L}_{AD}$ attention distillation using the β and γ trade-offs respectively. The β trade-off is the one that balances the stability-plasticity dilemma and we chose it using the CHF with a starting value of 2. The γ trade-off is fixed to 1 since it does not directly affect the stability-plasticity dilemma. Since there is no mention in the original work on which are the better values to balance the three losses, that last value was chosen after a separate test with values $\gamma \in (0, 2]$ and fixed for all scenarios in Section 6.
- **iCaRL**: we implement the five algorithms that comprise iCaRL. The distillation loss is combined with the cross-entropy loss during the training sessions and chosen using the CHF with a starting value of 4. However, during evaluation, the NME is used instead of the softmax outputs.
- **EEIL**: we implement EEIl with the balanced and unbalanced training phases. The unbalanced phase uses the hyperparameters shared across all methods. However, for the balanced phase the LR is reduced by 10 and the number of training epochs to 40. As with LwF, $T = 2$ and the trade-off is chosen using the CHF starting at 10. However, we apply a slight modification to the original work by not using the addition of noise to the gradients. Our preliminary results with this method showed that it was consistently detrimental to performance, which provided a worse representation of the capabilities of the method.
- **BiC**: the distillation stage is implemented the same as LwF, as in the original paper, with $T = 2$. However, the trade-off between distillation and cross-entropy losses is not chosen using the CHF. The authors already propose to set it to $\frac{n}{n+m}$, where n is the number of previous classes,

and m is the number of new classes, and we keep that decision. On the bias correction stage, also following the original work, we fix the percentage of validation split used from the total amount of exemplar memory to be 10%.

- **LUCIR**: for this method we make two changes on the architecture of the model. First, we replace the classifier layer by a cosine normalization layer following Eq. 14; and second we remove the ReLU from the penultimate layer to allow features to take both positive and negative values. However, since this procedure is only presented in the original work for ResNet models, we do not extend it to other architectures. The original code used a technique called imprint weights during the initialization of the classifier. However, since it was not mentioned in the original paper, and preliminary experiments showed no significant difference, we decided to not include it in our implementation.
  The cross-entropy loss is combined with the $\mathcal{L}_{lf}$ less-forget constraint from Eq. 12 and the $\mathcal{L}_{mr}$ margin ranking loss from Eq. 15. The number of new class embeddings chosen as hard negatives and the margin threshold are fixed to $K = 2$ and $m = 0.5$ as in the original work. The margin ranking loss is combined with the cross-entropy loss in a one-to-one ratio, while the less-forget constraint is chosen using the CHF with a starting value of 10, as is the trade-off related to the stability-plasticity dilemma.
- **IL2M**: since it only stores some statistics on the classes and applies them after the training is done in the same way as Finetuning, there is no hyperparameter to tune for this method.

Finally, the Finetuning, Freezing and Joint training baselines have no hyperparameters associated to them, reducing the Continual Hyperparameter Framework to only performing the learning rate search for each task before doing the final training.

APPENDIX B
SUPPLEMENTAL RESULTS

A. *On sampling strategies*

The better performance achieved by using herding in comparison to other sampling strategies is also very clear in the CIFAR-100 (50/11) scenario. As seen in Table S1, for longer task sequences herding has a clear benefit over the other sampling strategies when using class-incremental learning methods. In the case of shorter sequences, similar to transfer learning, performance does not seem to specifically favour any sampling strategy.

B. *On semantic tasks*

The popularity of iCaRL and the interest in comparing with it makes it quite common to utilize the random class ordering for experiments based on CIFAR-100 [84]. The authors of iCaRL use a random order of classes which is fixed in the iCaRL code by setting the random seed to 1993 just before shuffling the classes. However, this gives very little insight on class orderings which make use of the coarse labels from that dataset to group classes into sharing similar semantic concepts. This was explored for the tinyImageNet (Stanford, CS231N [109]) dataset in [16], [17], where the authors show that some methods report different results based on different semantics-based class orderings. In [16], the iNaturalist [110] dataset is split into tasks according to supercategories and are ordered using a relatedness measure. Having tasks with different semantic distributions and learning tasks in different orders is interesting for real-world applications where subsequent tasks are based on correlated data instead of fully random. Recently, [111] also brings attention to the learning variability between using different class orderings when learning a sequence of tasks incrementally.

TABLE S1: CIFAR-100 (50/11) with different sampling strategies and fixed memory of 20 exemplars per class on ResNet-32 trained from scratch.

acc. after	sampling strategy	FT-E	LwF-E	EWC-E	EEIL	BiC
task 2	random	42.4	49.0	**47.2**	44.5	55.5
	herding	**48.0**	**51.7**	45.1	**47.9**	53.5
	entropy	39.6	43.6	38.6	38.4	46.1
	distance	36.0	44.0	33.3	37.4	43.6
	inv-entropy	41.4	44.5	45.5	43.3	**55.6**
	inv-distance	44.3	48.2	43.9	40.3	47.9
task 5	random	**38.5**	34.2	30.4	**41.3**	43.2
	herding	36.5	**36.6**	**34.1**	40.8	**44.6**
	entropy	27.3	24.4	20.2	28.2	31.4
	distance	25.1	25.2	20.0	27.6	31.2
	inv-entropy	34.5	32.4	30.0	35.9	41.6
	inv-distance	33.1	32.5	30.0	37.0	38.3
task 10	random	**32.5**	26.0	22.7	37.3	36.1
	herding	32.0	**26.3**	**23.6**	**38.8**	**39.1**
	entropy	16.1	14.8	10.7	23.0	25.9
	distance	17.1	13.5	8.5	23.0	22.7
	inv-entropy	28.7	22.2	21.8	30.1	32.8
	inv-distance	29.2	23.3	20.6	27.1	35.4

In joint training, specific features in the network can be learned that focus on differentiating two classes that are otherwise easily confused. However, in an IL setting those discriminative features become more difficult to learn or can be modified afterwards, especially when the classes belong to different tasks. Thus, the difficulty of the task can be perceived differently in each scenario. Depending on the method, this issue may be handled differently and therefore lead to more catastrophic forgetting. This setting is different from the one proposed in Curriculum Learning [112], since the objective here is not to find the best order to learn tasks efficiently, but rather to analyze incremental learning settings (in which the order is not known in advance) and analyze the robustness of methods under different task orderings.

In order to investigate robustness to class orderings, we use the 20 coarse-grained labels provided in the CIFAR-100 dataset to arrive at semantically similar groups of classes. Then, we order these groups based on their classification difficulty. To assess performance we trained a dedicated model with all CIFAR-100 data in a single training session and use this model accuracy as a proxy value for classification difficulty. Finally, we order them from easier to harder (Dec.

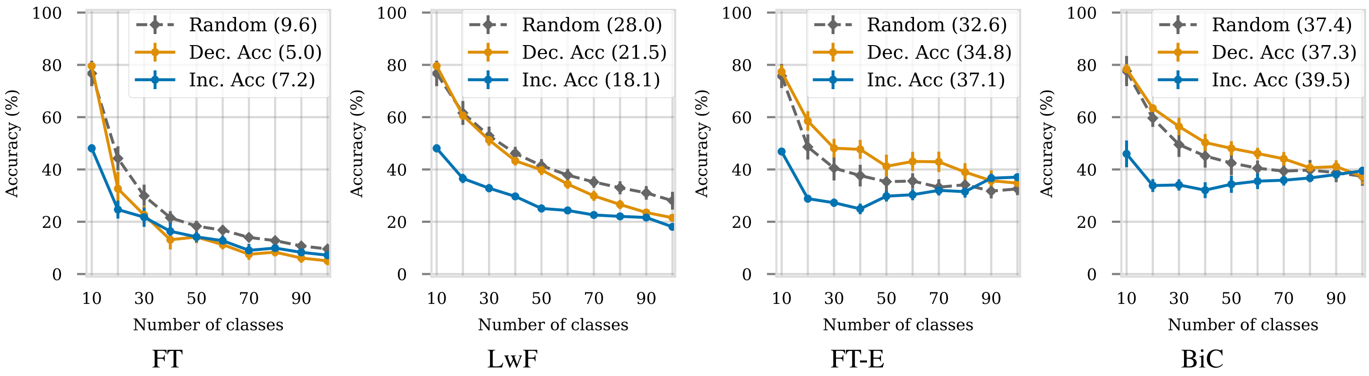


Fig. S1: Class ordering results for CIFAR-100 on ResNet-32 trained from scratch. For FT-E and BiC, 20 exemplars per class are sampled using herding. Error bars indicate standard deviation over six runs.

Acc.) and the other way around (Inc. Acc.). Results are presented in Fig. S1 for two methods without exemplars (FT+, LwF), and two methods with exemplars (FT-E, BiC). Performance can be significantly lower when using a semantics-based ordering compared to random one. In the examplar-free cases, special care of the used task ordering should be taken as the final performance after learning all classes can have quite some variability as seen in the LwF case. However, the variation with respect to the orderings is mitigated by the use of exemplars. Therefore, evaluating methods which use exemplars with randomized task orderings often suffices.