

Class-incremental learning: survey and performance evaluation on image classification

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

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 discriminate 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 class-incremental learning methods for image classification, and in particular we perform an extensive experimental evaluation on thirteen class-incremental methods. We consider several new experimental scenarios, including a comparison of class-incremental methods on multiple large-scale image classification datasets, investigation into small and large domain shifts, and comparison of various network architectures.

I. INTRODUCTION

Incremental 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

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.

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].

We adopt the viewpoint on incremental 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), 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 (see Fig. 1). Scenarios for which the task-ID is typically absent at inference time include those that incrementally increase the granularity of their capacity (e.g. detecting a growing number of object classes in images). In the last few years a wide variety of methods for class-IL have been proposed, and the time is ripe to provide a broad overview and experimental comparison of them.

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 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 thirteen methods on several scenarios

MAJOR FINDINGS OF OUR PERFORMANCE EVALUATION ON CLASS-INCREMENTAL LEARNING

- For exemplar-free class-IL, data regularization methods outperform weight regularization methods (see Table II).
- Finetuning with exemplars (FT-E) yields a good baseline that outperforms more complex methods on several experimental settings (see Figs. 9 and 10).
- Weight regularization combines better with exemplars than data regularization for some scenarios (see Figs. 8 and 9).

- Herding is a better than random exemplar sampling for longer sequences of tasks, but not for short ones (see Table III).
- Methods that explicitly address task-recency bias outperform those that do not.
- Network architecture greatly influences the performance of class-IL methods, in particular the presence or absence of skip connections has a significant impact (see Fig. 12).

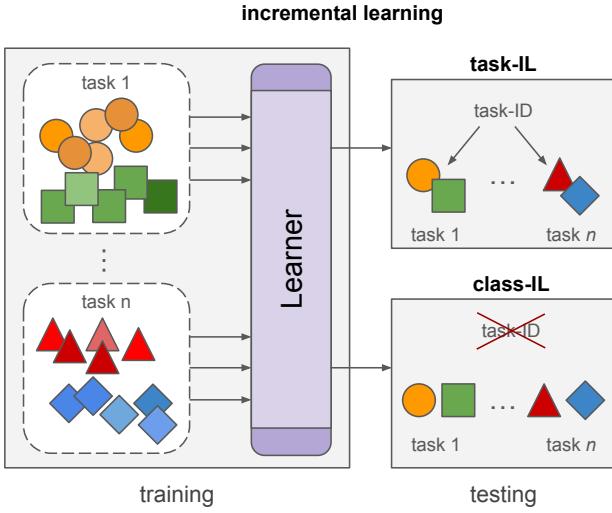


Fig. 1: In incremental learning, disjoint tasks are learned sequentially. Task-IL has access to the task-ID during evaluation, while the more challenging setting of class-IL does not. Class-IL is the subject of this survey.

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 summarize the outcomes of our survey in the “recommendations box” at the top of this page. Our extensible class-IL evaluation framework, including code to reproduce results, is publicly available¹.

This paper is organized as follows. In Sec. II we define class-incremental learning and the main challenges which need to be addressed. In Sec. III 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. IV we describe related work. 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. CLASS-INCREMENTAL LEARNING

Incremental learning is related to several research topics, including continual learning and lifelong learning. Although continual learning and incremental learning are often used interchangeably [8], [9], incremental learning can be thought of as a type of continual learning in which tasks are presented

in chunks of new supervised data. Continual learning is by no means restricted to the supervised scenario. In fact, one work that sparked new interest in continual learning [5] was applied to Deep Reinforcement Learning for video games. Lifelong learning [10], [11] can be thought of as the problem of building intelligent systems capable of learning throughout an extended life-cycle in which new knowledge must be acquired and integrated to accomplish new tasks. Continual learning is one of the characteristics of a lifelong learning system, however even this distinction is often blurred and it is also used interchangeably with incremental and continual learning [12], [13].

A. The practical importance of incremental learning

The notable increase in attention IL has received in the last few years has been fueled by a demand from applications in industry and society. There are several problems for which incremental knowledge assimilation offers a solution.

Memory restrictions: Systems that have physical limitations for the data that they can store cannot resort to joint training strategies because they simply cannot store all seen data. Such systems can only store a limited set of examples for the tasks they perform, and therefore learning must be done incrementally. This scenario is especially common in robotics [14], where robots are faced with different tasks at different times or locations, but should still be able to perform all previously learned tasks.

Data security/privacy restrictions: For systems that learn from data that should not be permanently stored, incremental learning can provide a solution. Government legislation could restrict data storage from clients at a central location (e.g. for applications on mobile devices). Privacy considerations are also common in health applications where legislation prevents the long-term storage of data from patients, and thus incremental learning is key [15].

Sustainable ICT: The cost of training deep learning algorithms can be exorbitant. Examples include GPT-2 which requires 1 week of training on 32 TPUv3 chips [16]. The carbon footprint of retraining such systems for every new data update is considerable, and will likely grow in the coming years [17]. Incremental learning provides algorithms that are much more computationally efficient and only require processing of new data when updating the system.

B. General class-incremental learning setup

Class-IL methods learn from a stream of data drawn from a non-stationary distribution. These methods should scale to

¹<https://github.com/mmasana/FACIL>

a large number of tasks without excessive computational and memory growth. They aim to exploit 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*) [18]. Our investigation focuses on class-incremental learning scenarios in which the algorithm must learn a sequence of tasks (see Section VII for discussion of task-free scenarios). 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 [18] in which each data sample is only seen once.

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

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

where each task t is represented by a set of classes $C^t = \{c_1^t, c_2^t, \dots, c_{n^t}^t\}$ 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. $C^i \cap C^j = \emptyset$ if $i \neq j$).

We consider incremental learners that 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:

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

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:

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

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.

C. 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], [9], [19]. 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) **Task-agnostic in evaluation:** incremental learners able to predict classes from all previously learned tasks without recourse to a task oracle at inference providing a subset of possible classes.
- 2) **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.
- 3) **Fixed network architecture:** methods using a fixed architecture for all tasks, without adding significant amount of parameters to the architecture for new tasks.
- 4) **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.
- 5) **Mature:** methods applicable to complex image classification problems.

Property 1 distinguishes class-incremental from task-incremental learning, while properties 2–5 are characteristics that we use to select methods for our evaluation.

Finally, we consider one additional (optional) property:

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

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.

D. 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

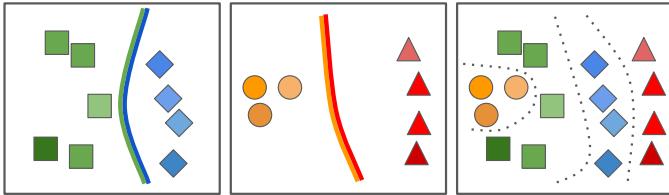


Fig. 2: A network trained continually to discriminate between task 1 (left) and task 2 (middle) is unlikely to have learned features to discriminate between the four classes (right). We call this problem *inter-task confusion*.

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* [20]. 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. 2). 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. 3).

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], [21], [22] or the drift of activations [23], [24].

The last two points are specific to class-IL since they have no access to a task-ID at inference time. Most research has focused on reducing task imbalance [25], [26], [27], which addresses the task-recency bias. To prevent inter-task confusion and learn representations which are optimal to

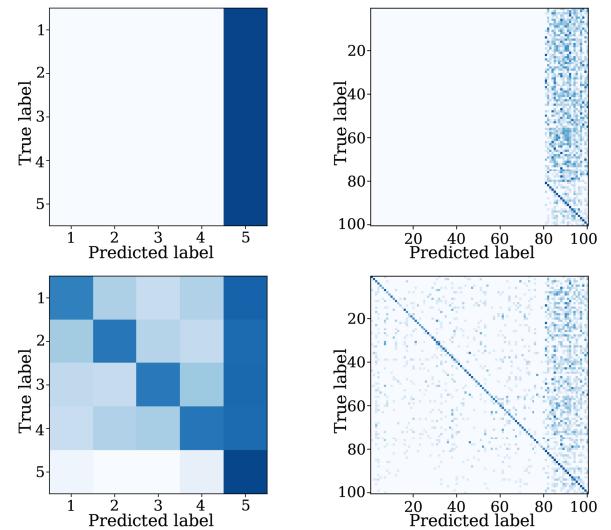


Fig. 3: Examples of task and class confusion matrices for Finetuning (top row) and Finetuning with 2,000 exemplars (bottom row) on CIFAR-100. Note the large bias towards the classes of the last task for Finetuning. By exploiting exemplars, the resulting classifier is clearly less biased.

discriminate between all classes, rehearsal [1], [28] or pseudo-rehearsal [29], [30], [31] are commonly used.

III. APPROACHES

In this section, we describe several approaches to address the above mentioned challenges of class-IL. We divide them into three main categories: regularization-based methods, rehearsal-based methods, and bias-correction methods.

A. 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], [21], [22], [32], [33], while others focus on the importance of remembering feature representations [23], [24], [34], [35], [36], [37]. Most of these approaches have been developed within the context of task-IL and have been reviewed by other works [9]. 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:

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

where θ_i^t is weight i of the network currently being trained, θ_i^{t-1} is the value of this parameter at the end of training on task $t-1$, $|\theta^{t-1}|$ 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 a diagonal approximation of the 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 [33], the authors 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 a similar vein of improving the approximation of the Fisher Information Matrix in EWC, the authors of [38] propose an extension of the Kronecker factorization technique for block-diagonal approximation of the Fisher Information Matrix. They additionally demonstrate how such Kronecker factorizations make accommodating batch normalization possible.

In contrast, [22] 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) [21] also proposes to calculate Ω_i online by accumulating the sensitivity of the learned function (the magnitude of the gradient). In [7], the *Riemannian 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 [39], [40] which was originally designed to learn a more compact student network from a larger teacher network. Li et al. [24] 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:

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

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

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

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

π^{t-1} to refer to the predictions of the network after training task $t-1$. The temperature scaling was introduced in [40] to help with the problem of having the probability of the correct class too high.

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], [26], [27], [28], [36], [41]. When the LwF method is combined with exemplars the distillation loss is typically also applied to the exemplars of previous classes [1], [26], [27], [28]. Finally, 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 [12].

A very similar approach, called *less-forgetting learning* (LFL), was proposed by Jung et al. [23]. 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 [34] 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.

Recent developments in regularization. Several new regularization techniques have been proposed in recent work on class-IL. Zagoruyko and Komodakis [42] proposed to use the attention of the teacher network to guide the student network. *Learning without Memorizing* (LwM) [43] 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:

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

where the attention map Q is given by:

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

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

and is generated with the Grad-CAM algorithm [44]. 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: $c = \text{argmax } h(\mathbf{x}; \theta^{t-1})$.

Another recent method building upon LwF is *Deep Model Consolidation* (DMC) [36]. 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):

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

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

$$\hat{\mathbf{o}}_k(\mathbf{u}) = \begin{cases} \mathbf{o}_k^{t-1}(\mathbf{u}) - \frac{1}{N^{t-1}} \sum_{l=1}^{N^{t-1}} \mathbf{o}_l^{t-1}(\mathbf{u}) & \text{if } 1 \leq k \leq N^{t-1} \\ \mathbf{o}_k^t(\mathbf{u}) - \frac{1}{N^t} \sum_{l=1}^{N^t} \mathbf{o}_l^t(\mathbf{u}) & \text{if } N^{t-1} < k \leq N^t. \end{cases} \quad (11)$$

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 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.

Similarly, *Global Distillation* (GD) [37] also proposes to use external data to distill knowledge from previous tasks. They first train a teacher on the current task and calibrate its confidence using the external data and exemplars. Then they triple-distill a new model from the teacher, the previous model, and their ensemble (plus the cross-entropy loss for current task and exemplars). The teacher and the previous task are used with both current task data and external dataset data, while the ensemble is only used with the external data. Finally, they perform finetuning while avoiding task-recency bias by weighting the loss according to the amount of data. The external dataset sampling method is based on the predictions of the model.

In [45], current task data is also not trained directly, but rather used to train an expert teacher first. The method additionally distills an old model with a reserved small fraction of previous task data to preserve the performance on old tasks, similar to LwF but using stored exemplars instead of new data. Based on an analysis of iCaRL, [46] claim that using nearest exemplar mean classifier is biased and propose using a *dynamic threshold moving* algorithm to fix that classifier bias trained with distillation loss by maintaining an up-to-date scaling vector.

Finally, the *less-forget constraint* proposed by Hou et al. [26] 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:

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

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.

B. Rehearsal approaches

Rehearsal methods keep a small number of exemplars [1], [27], [28] (exemplar rehearsal), or generate synthetic images [29], [47] or features [31], [48] (pseudo-rehearsal). By replaying the stored or generated data from previous tasks rehearsals 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 [49], 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], [26], [27]. 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 [28] 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], [26], [27], [28] use the distillation loss from Learning without Forgetting [24] to deal with the activation drift in combination with exemplars. However, Beloudah and Popescu [25] 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.

Recent research on task-IL [50] shows that data regularization (referred to as functional regularization) can provide a natural way to select data for rehearsal by choosing the inducing points of the Gaussian process used to approximate the posterior belief over the underlying task-specific function (network output). This direction was further explored in [51], however the usefulness of these approaches for class-IL is still to be determined.

C. 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. [26], 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. 4, 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

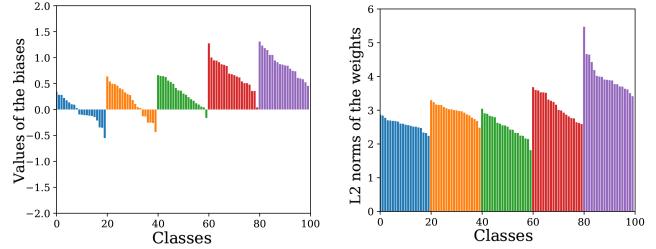


Fig. 4: Bias and weight analysis for iCaRL with 2,000 exemplars on CIFAR-100. We show the ordered biases and norm of the last classification layer of the network for different tasks. Note how the bias and the norm of the weights are higher for the last tasks. This results in a *task-recency bias*.

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. [28] 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. [27], 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:

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

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. [26]. In their method *Learning a Unified*

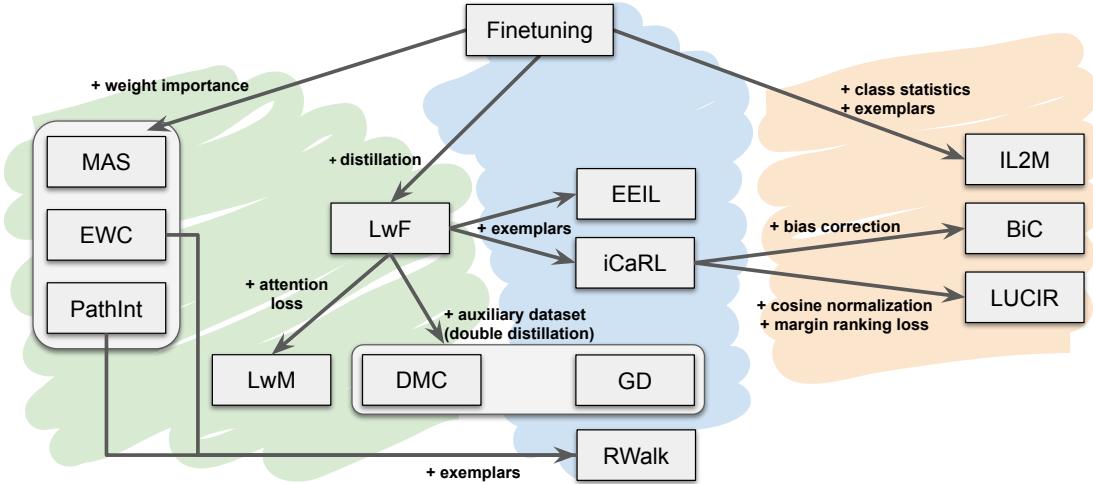


Fig. 5: Relation between class-IL methods. We distinguish three main categories: exemplar-free regularization (green), rehearsal (blue), and rehearsal with explicit bias-correction mechanisms (orange). Methods sharing relation are joined in a rounded box.

Classifier Incrementally via Rebalancing (LUCIR), they propose to replace the standard softmax layer σ with a cosine normalization layer according to:

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

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. [26] 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:

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

where \hat{V}_y refers to the ground truth class embedding of \mathbf{x} , \hat{V}_k is 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 [25] with their method called *Class-IL with dual memory* (IL2M). Their method is similar to BiC [27] 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 $m = \arg \max \hat{y}(\mathbf{x})$, they compute the rectified predictions of the previous classes k as:

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

Here \bar{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 \bar{y}^p is the

mean of the predictions for all classes in that task. Both \bar{y}_k^p and \bar{y}^p are stored directly after their corresponding training session. \bar{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, \bar{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.

D. Relation between class-incremental methods

In previous sections we discussed the main approaches to mitigating catastrophic forgetting by incremental learning methods. We summarize their relations in Fig. 5 starting from the naive finetuning approach. In the diagram we show all methods which we compare in Sec. VI. It 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) [24] upon the whole field of class-IL is clear. However, we expect that with the recent findings of [25], which show that when combined with exemplars finetuning can outperform LwF, could somewhat lessen its continuing influence. Weight regularization methods [5], [21], [22], 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. 5 also shows the influence of iCaRL [1] in the development of more recent methods [26], [27].

IV. RELATED WORK

In this section we broadly review related work, focusing mainly on works not discussed in the previous section.

Existing surveys. The problem of catastrophic forgetting has been acknowledged for many years. Already in the eighties, McCloskey and Cohen [6] showed that algorithms trained

with backpropagation suffered from catastrophic forgetting. Radcliff [52] 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 [53] 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. [54] 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 [14] 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. Pfubl and Gepperth [55] propose a training and evaluation paradigm for task-IL methods, limited to two tasks. De Lange et al. [9] 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.

Mask-based methods. This type of parameter isolation 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 effective for task-aware evaluation, but impractical for task-agnostic settings [9], [56]. Piggyback learns masks on network weights while training a backbone [57]. PackNet learns weights and then prunes them to generate masks [58]. HAT [59] applies attention masks on layer activations to penalize modifications to those that are important for a specific task. DAN [60] combines existing filters to learn filters for new tasks. Finally, PathNet [61] learns selective routing through the weights using evolutionary strategies.

Dynamic architectures. The other type of parameter isolation method, called architecture growing, dynamically increases the capacity of the network to reduce catastrophic forgetting. These methods 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 [12] duplicates the model for each new task in order to completely eliminate forgetting. PNN [62] extends 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, complexity increases with the number of tasks. To address this issue, P&C [19] proposes duplicating the network only once to keep the number of parameters fixed, and use EWC [5] to mitigate forgetting. In [63], similar classes are grouped together expanding hierarchically, at the cost of an expensive training procedure and a rigid architecture. ACL [64] fuses a dynamic architecture with exemplars, explicitly disentangling shared and task-specific features with an adversarial loss. This allows to learn shared features that are more robust to forgetting.

Finally, Random Path Selection (RPS) [65] provides better performance with a customized architecture by combining distillation and rehearsal-based replay. Contrary to some of the previously mentioned approaches, RPS does not need a task-ID at inference time. However, in order to learn the different paths for each task, the proposed architecture is much larger than other class-IL approaches. Since this approach needs to use their particular RPSNet architecture and the capacity is not comparable to the other approaches compared in this survey, we provide results in Appendix B (see Sec. B.2 and Table S1), for an analysis on different numbers of paths and memory required.

Online Incremental learning. Within the field of incremental 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.

Lopez-Paz [18] 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 [13] improves on GEM in efficiency by constraining based on the average of gradients from previous class exemplars. However, the authors of [66] show that simply training on the memorized exemplars, similar to the well-established technique in reinforcement learning [67], outperforms previous results. GSS [68] performs gradient-based exemplar selection based on the GEM and A-GEM procedure to allow training without knowing the task boundaries. MIR [69] trains on the exemplar memory by selecting exemplars that will have a larger loss increase after each training step. In [70] the memory is used to store discrete latent embeddings from a Variational Autoencoder that allows generation of previous task data for training. MER [71] combines experience replay with a modification of the meta-learning method Reptile [72] to select replay samples which minimize forgetting.

Variational continual learning. Variational continual learning is based on the Bayesian inference framework. VCL [73] 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 [74] introduces Variational Generative Replay, a variational inference generalization of Deep Generative Replay (DGR), which is complementary to VCL. UCL [75] 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 [76] proposes to use Natural Gradients and Stein Gradients to better estimate posterior distributions over the parameters and to construct coresets using approximated posteriors. IUVCL [77] proposes a new best-practice approach to mean-field variational Bayesian neural networks. CLAW [78] extends VCL by applying an attention mechanism on the whole network which allows automation of the architecture adaptation process that assigns parameters to be fixed or not after each task. UCB [79] defines uncertainty for each weight to control the change in the parameters of a Bayesian Neural Network, identifying which are the weights that should stay fixed or change. They further extend their method by using a pruning strategy together with binary masks for each task to retain performance from previous tasks. These methods normally consider only evaluation on task-IL. BGD [80] 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 [29] 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 [30], where a label-conditional GAN and replay alignment are used. DGM [47] 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 [81] extends image generation without catastrophic forgetting from label-conditional to image-conditional GANs. As an alternative to exemplar rehearsal, some methods perform *feature replay* [31], [48], which need a fixed backbone network to provide good representations.

Incremental Learning beyond image classification. Shmelkov et al. [82] propose to learn object detectors incrementally. They use Fast-RCNN [83] 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. [84] extend Faster-RCNN [85] with knowledge distillation. Similarly, Michieli et al. [86] propose to distill both on the output logits and on intermediate features for incremental semantic segmentation. Recently, Cermelli et al. [87] 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 [88]

and medical data [89].

Catastrophic forgetting has been mainly studied in feed-forward neural networks. Only recently the impact of catastrophic forgetting in recurrent LSTM networks was studied [90]. 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 [91]. A Net2Net technique [92] was combined with gradient episodic memory in [93]. 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 [94].

The paper that introduced EWC [5] also considered training Deep Reinforcement Learning (DRL) agents to play multiple Atari games [67] over their lifetimes. Reinforcement learning is an application area of deep learning in which *task specification* is usually implicit in the definition of the reward function to be optimized, and as such is another example where laboratory practice often does not completely reflect the real world since the agent’s goals must evolve with the changing environment around them. Incremental task acquisition enjoys a long tradition in the reinforcement learning community [95], and more recently the Continual Learning with Experience And Replay (CLEAR) approach mixes on-policy learning for plasticity with off-policy learning from replayed experiences to encourage stability with respect to tasks acquired in the past [96].

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-E).

The current version of the code includes implementations of several baselines and the following methods: EWC, MAS, PathInt, RWalk, LwM, DMC, GD, 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

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

B. Datasets

We study the effects of CL methods for image classification on nine different datasets whose statistics are summarized in Appendix A. First, we compare the three main categories of approaches described in Sec. III on the CIFAR-100 dataset [97]. Next, we use several fine-grained classification datasets: Oxford Flowers [98], MIT Indoor Scenes [99], CUB-200-2011 Birds [100], Stanford Cars [101], FGVC Aircraft [102], and Stanford Actions [103]. These provide higher resolution and allow studying the effects on larger domain shifts when used as different tasks. To study the effects on smaller domain shifts we use the VGGFace2 dataset [104]. 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 [25]. 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 [105] 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 class-balanced split of 10% from training is assigned to validation for those datasets. In the case of Flowers and Aircrafts, we fuse the official train and validation splits and then randomly extract a class-balanced 10% validation split.

C. Metrics

In incremental learning, $a_{t,k} \in [0, 1]$ denotes the accuracy of task k after learning task t ($k \leq t$), which provides precise information about the incremental process. In order to compare the overall 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 used to compare performances of different methods with a single value. When tasks have different number of classes, a class frequency weighted version is used.

Additional metrics focusing on several aspects of IL such as *forgetting* and *intransigence* [7] have also been proposed. *Forgetting* estimates how much the model forgot about previous tasks. Another measure, *intransigence* quantifies a model's inability to learn a new task. Both can be considered complementary measures that help understand the stability-plasticity dilemma. These measures were originally proposed for task-IL. However, their use in class-IL was not found equally useful. When adding new tasks the performance of previous tasks drops because the learner has to perform the more complex task of classifying data in all seen classes. This effect will incorrectly contribute to the forgetting measure. Therefore, in this survey we use *average accuracy* as the main metric.

TABLE I: 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 80.7 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+	38.2	31.0	22.6	18.6	15.7	14.4	13.1	11.6	10.1
FT-E (fixed)	65.7	61.7	55.0	51.7	48.3	46.2	41.1	38.7	37.9
FT-E (grow)	49.9	46.0	36.9	38.9	37.1	37.0	34.5	34.5	34.6
FZ	24.1	18.4	12.8	12.7	9.2	8.2	7.8	6.3	5.3
FZ+	42.2	31.3	24.5	23.1	20.5	18.3	17.0	15.6	14.4
FZ-E (fixed)	50.0	37.1	26.1	24.2	19.5	19.4	15.3	14.3	11.3
FZ-E (grow)	40.5	31.2	22.0	20.9	16.6	17.6	13.7	13.8	11.3

All reported CIFAR-100 results are averages over 10 runs, while the domain shift and different architecture results are averages over 5 runs. Each run uses a different random seed, but these are fixed across the different approaches so that the comparison is on identical splits generated from the same set of seeds.

D. 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 I, 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 I, 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.

E. 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.

In order to provide this more robust evaluation, we use the Continual Hyperparameter Framework proposed in [9]. 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 [9] (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.

F. Network architectures

As suggested in [106], 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. Therefore, we use those architectures trained from scratch for most of the experiments, but we also include an analysis on different architectures in Sec. VI-F and Appendix B-E.

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 of 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 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. In our evaluation, we do not consider the case where a task consists of only a single class. This is because several methods cannot be straightforwardly applied to this scenario, mainly because they train a cross-entropy loss on only the last task (e.g. BiC, DMC). Adding tasks with multiple classes is the most common scenario considered in class-IL literature.

VI. EXPERIMENTAL RESULTS

In this section we evaluate a large number of incremental learning methods in terms of many aspects of incremental learning on a broad variety of datasets.

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], [23], [24], [32], [34]. 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], [36], [43]. Similarly to the baselines in Sec. V-D, 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 II 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], [26], [27], [28].

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 II 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, even having a better performance than RWalk for fixed memory.

TABLE II: 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	59.8	72.0	61.5	63.8	63.4	63.4	71.0	74.2
	task 5	49.8	76.7	60.2	57.3	61.8	56.3	72.3	76.2
	task 10	38.3	76.6	56.7	53.1	58.6	49.3	66.7	70.4
No exemplars (Class-IL)	task 2	38.2	55.4	39.8	41.2	39.9	40.3	58.2	57.8
	task 5	18.6	41.6	21.9	23.5	22.1	22.9	41.5	37.4
	task 10	10.1	30.2	13.1	13.6	13.9	14.0	25.9	21.9
2,000 exemplars fixed memory (Class-IL)	task 2	65.7	63.4	61.5	56.8	57.6	56.9	-	65.5
	task 5	51.7	46.2	42.7	34.5	29.3	36.5	-	52.7
	task 10	37.9	30.8	28.1	18.5	18.9	22.7	-	37.4
20 exemplars per class growing memory (Class-IL)	task 2	49.9	51.0	47.9	45.1	45.3	44.1	-	53.7
	task 5	38.9	32.6	32.1	26.3	22.9	27.0	-	39.4
	task 10	34.6	27.2	25.4	17.3	15.9	20.3	-	32.3

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 second best results in all cases when combined with exemplars, even though the method was originally not proposed with exemplars. Furthermore, FT-E performs the best in this scenarios, in front of LwF, as also noticed in [25]. 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 [36]), 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. 3, 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. 3 and Fig. 6. 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.

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 create new approaches that can both have better overall performance while simultaneously addressing the bias-correction issue.

C. On exemplar usage

Here we study the effects of different characteristics related to exemplars. 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 7 we compare several rehearsal methods with different numbers of exemplars per class in a 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

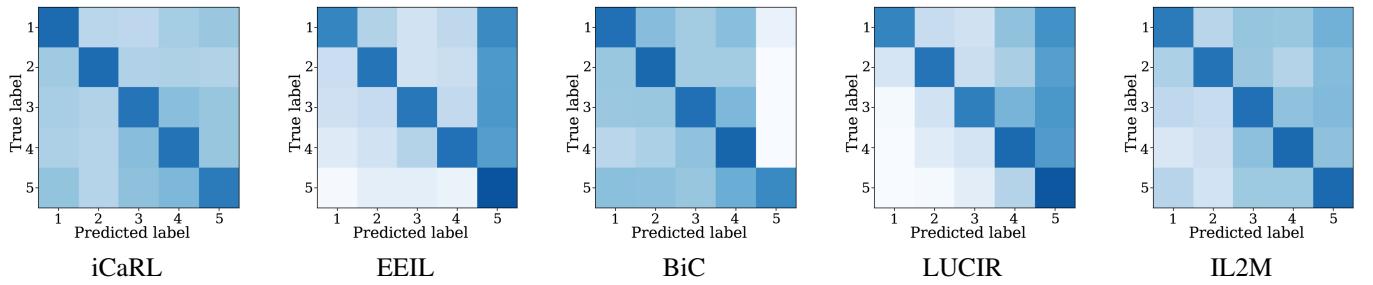


Fig. 6: Task (top) and class (bottom) confusion matrices. CIFAR-100 (20/5) with 2,000 exemplars selected with herding.

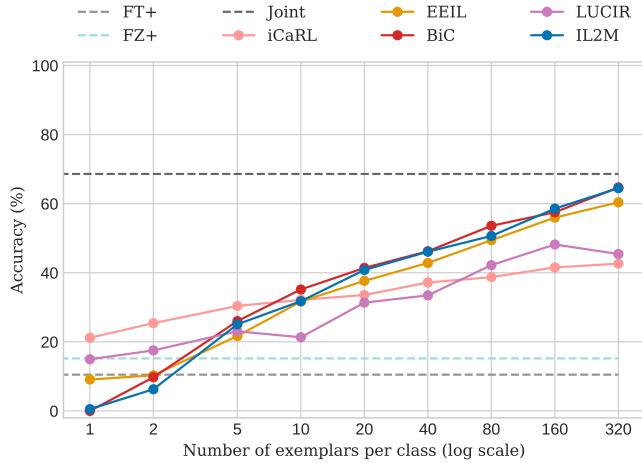


Fig. 7: Results for CIFAR-100 (10/10) on ResNet-32 trained from scratch with different exemplar memory sizes.

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. III-B, for rehearsal approaches there are different strategies to select which

TABLE III: 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

exemplars to keep. In Table III 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. III-B: 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-C.

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,

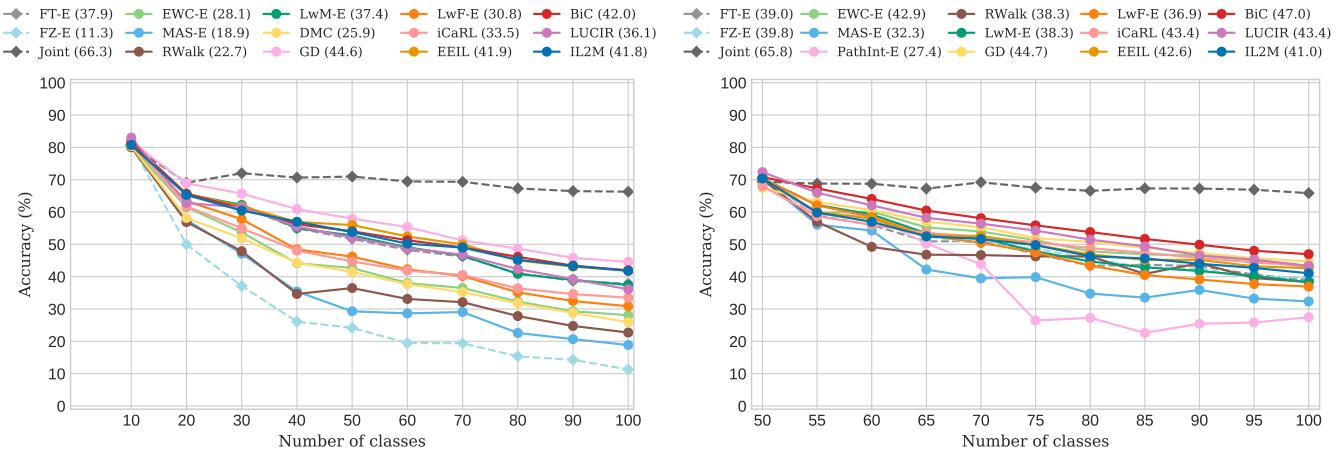


Fig. 8: CIFAR-100 (10/10) with 2,000 exemplar fixed memory (left), and CIFAR-100 (50/11) with 2,000 exemplar fixed memory (right). Results with 20 exemplars per class growing memory are available in Appendix B-A.

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 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. The scenario is evaluated with fixed memory (2,000 total exemplars) with herding as the sampling strategy (results for the growing scenario are provided in Appendix B-A). DMC and GD use an external dataset (reduced ImageNet-32 [36]) that is much larger than the memories. Except DMC, 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 8 (left), GD, BiC, EEIL and IL2M achieve the best results after learning 10 tasks. They are followed by iCaRL, LwM-E, and then LUCIR. Some methods have different starting points on task 1 since they do not have the same initial conditions as the other approaches (e.g. LUCIR uses cosine linear layers, while BiC uses fewer data during training because it stores some for bias-correction parameter training). 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 [25], FT-E achieves competitive performance similar to the lowest performance of that family.

Figure 8 (right) shows that, in general, all methods improve when starting from a larger number of classes, probably because anchoring to the first task already yields more diverse features. This is especially noticeable in the case of FZ-E. The results show the importance of comparing to this baseline when doing experiments with pretrained models or

a very strong first task. Both LUCIR and EWC-E also seem to perform much better in this scenario. Notably, GD does not perform as well in this setting since starting with a larger amount of classes and data slightly removes the advantage of having the extra unlabelled data.

D. On domain shift effects

Up to this point experiments were 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 [104]. 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 discriminative enough to generalize to new classes.

—♦— FT-E (68.8) —●— EWC-E (70.0) —○— iCaRL (55.4) —●— LUCIR (73.0)
 —△— FZ+ (34.3) —●— LwM-E (55.0) —○— EEIL (58.0) —●— IL2M (70.9)
 —◆— Joint (98.7) —●— LwF-E (43.5) —●— BiC (72.2)

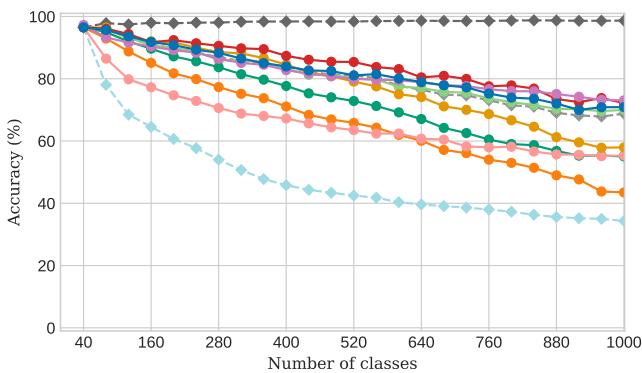


Fig. 9: Small domain shifts on VGGFace2 (40/25) on ResNet-18 trained from scratch and 5,000 exemplar fixed memory.

—♦— FT-E (41.0) —●— EWC-E (34.2) —○— LwF-E (37.3) —●— BiC (41.7)
 —△— FZ+ (7.6) —●— MAS-E (36.3) —○— iCaRL (38.5) —●— LUCIR (41.7)
 —◆— Joint (62.3) —●— LwM-E (40.6) —○— EEIL (41.1) —●— IL2M (39.9)

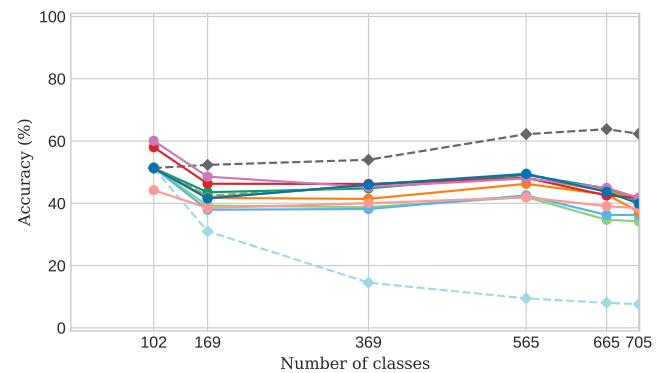


Fig. 10: Large domain shifts with multiple fine-grained datasets (Flowers, Scenes, Birds, Cars, Aircraft, Actions).

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, 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. 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.

E. 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. 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.

F. On network architectures

We compare the four most competitive methods over a range of different network architectures in Fig. 12. Specifically, we use AlexNet [53], ResNet-18 [106], VGG-11 [107], GoogleNet [108] and MobileNet [109]. An interesting observation is that for different networks, the performance rank-

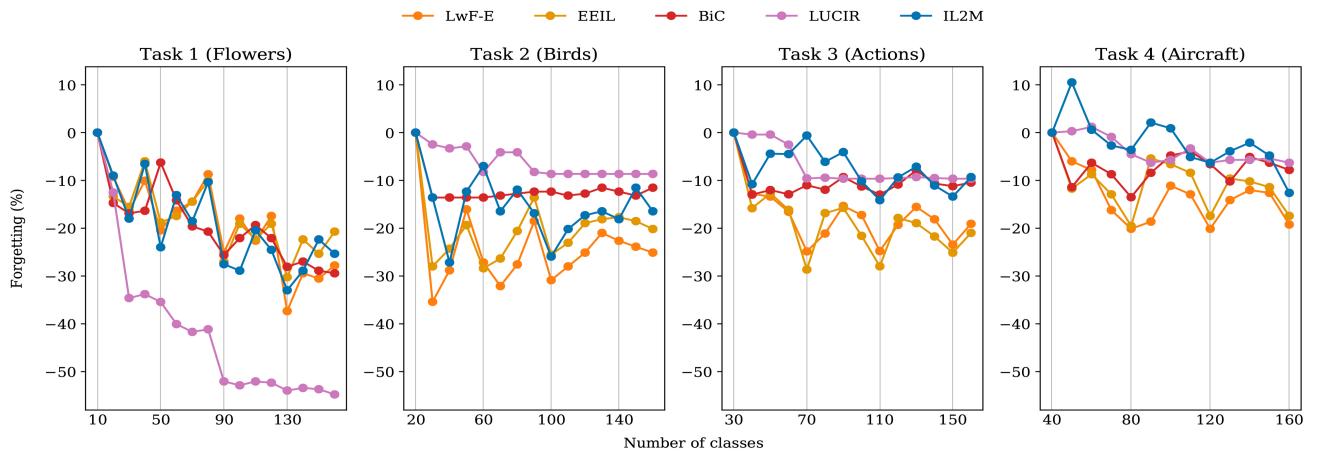


Fig. 11: Forgetting when revisiting old domains with new classes from different fine-grained datasets on AlexNet.

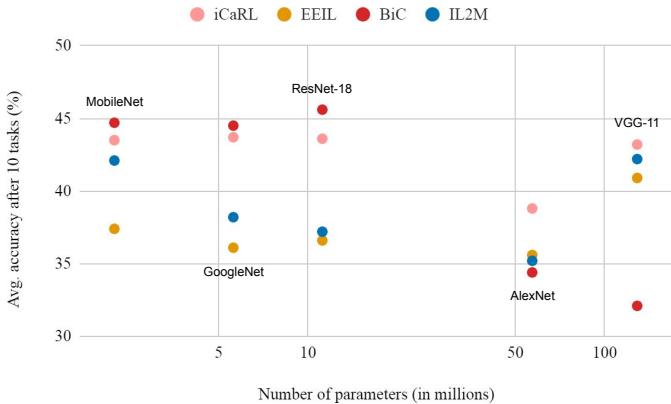


Fig. 12: Average accuracy after 10 tasks on ImageNet-Subset-100 (10/10) with different networks trained from scratch. From left to right: MobileNet (2017), GoogleNet (2014), ResNet-18 (2015), AlexNet (2012) and VGG-11 (2014).

ings of the methods can change completely. For instance, in architectures which do not use skip connections (AlexNet, VGG-11), iCaRL performs the best. On the other hand, BiC performs worse without skip connections, but performs the best with architectures that have them (ResNet-18, MobileNet and GoogleNet). IL2M is more consistent compared to other methods, 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 other methods 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 parameters/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.

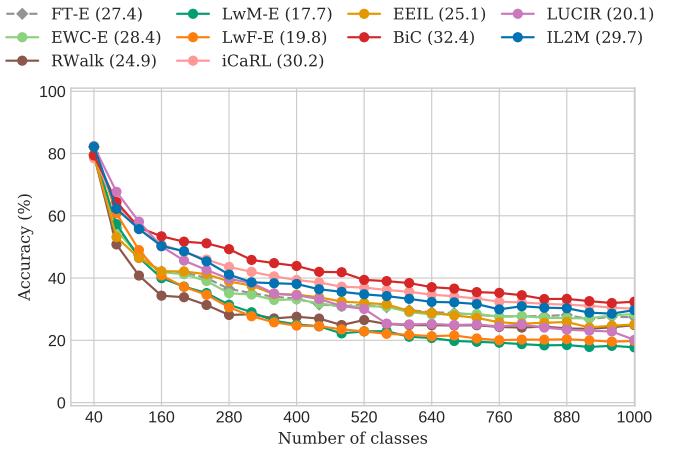


Fig. 13: ImageNet (40/25) on ResNet-18 with growing memory of 20 exemplars per class and herding sampling.

G. 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 13 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.

VII. EMERGING TRENDS IN CLASS-IL

Here we discuss some recent developments in class-IL that 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 [41], [110]. This enables much more efficient use of available storage. Liu et al. [41] propose Mnemonics Training, a method that trains the parametrized exemplars. The exemplars are optimized to prevent the forgetting when evaluated on the current task data. Chaudry et al. [110] generalize the theory to a streaming setting, where the learning of exemplars does not require multiple loops over the data for every task. Optimizing the available storage by computing more efficient exemplars is expected to attract more research in the coming years.

Feature rehearsals. Pseudo-rehearsal is a good alternative to storing exemplars [29], [30], [47]. It learns a separate network that generates images of previous tasks. However, current state-of-the-art image generation methods struggle to realistically generate complex image data, and therefore this approach has been applied to simple datasets and is known to obtain unsatisfying results on complex ones. To address this problem, some works have proposed to perform *feature* replay instead of image replay [31], [111], [112], 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 [113]. Moving away from image replay towards different variants of feature replay is expected to gain traction.

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. [114] 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 [115] is only little explored within the context of IL, and is expected to gain interest.

Beyond cross-entropy loss. Several recent works show that the cross-entropy loss might be responsible for high levels of catastrophic forgetting [116], [117]. Less forgetting has been reported by replacing the cross-entropy loss with a metric learning loss [116] or by using a energy-based method [117]. Combining these methods with the other class-IL categories, such as bias-correction and rehearsal, is expected to result in highly competitive methods.

Meta-learning. Meta-learning aims to learn new tasks leveraging information accrued while solving related tasks [118]. Riemer et al. [71] show that such a method can learn parameters that reduce interference of future gradients and improves transfer based on future gradients. Javed and White [119] 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.

Task-free settings. Many practical applications do not fit well into the experimental setup with non-overlapping tasks. A more realistic scenario is one where there are no clear task boundaries and the distribution over classes changes gradually. This scenario is expected to receive increased attention in near future. This setting was studied in several early task-aware continual learning works, including EWC [5] and P&C [19]. The transition to the task-free setting is not straightforward, since many methods have inherent operations that are performed on the task boundaries: replacing the old model, updating of importance weights, etc. Recently, several works for class-IL started addressing this setting [117], [120], [121].

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 thirteen 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 II). 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 [25] showing that the added regularization does not improve results and the baseline method of finetuning with exemplars performs better (see Table II). However, using LwM for data regularization does perform better than the baseline.
- Using additional data to distill knowledge from previous tasks to new networks can greatly improve results [37] (see Section VI-C).
- We found that in several scenarios weight regularization outperforms the baseline LwF-E significantly (see Figs. 8 and 9), showing that the IL community choice of data regularization with LwF (see Fig. 5) 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 III).

- Methods that explicitly address the task-recency bias obtain better performance for class-IL (see Figs. 8, 9, 10, 13): 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.

ACKNOWLEDGMENTS

We acknowledge the support from Huawei Kirin Solution. Marc Masana acknowledges grant 2019-FI_B2-00189 from Generalitat de Catalunya. Joost van de Weijer acknowledges the Spanish project PID2019-104174GB-I00. Andrew D. Bagdanov acknowledges the project “PON IDEHA - Innovazioni per l’elaborazione dei dati nel settore del Patrimonio Culturale” of the Italian Ministry of Education.

REFERENCES

- [1] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.
- [2] S. Thrun, “Is learning the n-th thing any easier than learning the first?” in *Proc. Adv. Neural Inf. Process. Syst.*, 1996.
- [3] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in cognitive sciences*, 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 *Proc. Int. Conf. Learn. Repres.*, 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*, 2017.
- [6] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in *Psychology of learning and motivation*, 1989.
- [7] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, “Riemannian walk for incremental learning: understanding forgetting and intransigence,” in *Proc. Eur. Conf. Comput. Vis.*, 2018.
- [8] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” in *NeurIPS Continual Learning Workshop*, 2018.
- [9] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardi, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.
- [10] S. Thrun, “A lifelong learning perspective for mobile robot control,” in *Intelligent Robots and Systems*, 1995.
- [11] Z. Chen and B. Liu, “Lifelong machine learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2018.
- [12] R. Aljundi, P. Chakravarty, and T. Tuytelaars, “Expert gate: Lifelong learning with a network of experts,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.
- [13] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, “Efficient lifelong learning with a-gem,” in *Proc. Int. Conf. Learn. Repres.*, 2019.
- [14] 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*, 2020.
- [15] P. McClure, C. Y. Zheng, J. R. Kaczmarzyk, J. A. Lee, S. S. Ghosh, D. Nielson, P. Bandettini, and F. Pereira, “Distributed weight consolidation: a brain segmentation case study,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.
- [16] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for deep learning in nlp,” in *Assoc. for Comput. Linguistics*, 2019.
- [17] O. Sharir, B. Peleg, and Y. Shoham, “The cost of training nlp models: A concise overview,” *arXiv*, 2020.
- [18] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
- [19] 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 *Proc. Int. Conf. Mach. Learn.*, 2018.
- [20] M. Mermilliod, A. Bugaiska, and P. Bonin, “The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects,” *Frontiers in psychology*, 2013.
- [21] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Proc. Eur. Conf. Comput. Vis.*, 2018.
- [22] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in *Proc. Int. Conf. Mach. Learn.*, 2017.
- [23] H. Jung, J. Ju, M. Jung, and J. Kim, “Less-forgetting learning in deep neural networks,” *arXiv*, 2016.
- [24] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.
- [25] E. Belouadah and A. Popescu, “Il2m: class incremental learning with dual memory,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2019.
- [26] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, “Learning a unified classifier incrementally via rebalancing,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2019.
- [27] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, “Large scale incremental learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
- [28] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, “End-to-end incremental learning,” in *Proc. Eur. Conf. Comput. Vis.*, 2018.
- [29] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
- [30] 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 *Proc. Adv. Neural Inf. Process. Syst.*, 2018.
- [31] Y. Xiang, Y. Fu, P. Ji, and H. Huang, “Incremental learning using conditional adversarial networks,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2019.
- [32] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, “Overcoming catastrophic forgetting by incremental moment matching,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
- [33] 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 *Int. Conf. on Pattern Recognit.*, 2018.
- [34] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, “Encoder based lifelong learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.
- [35] D. L. Silver and R. E. Mercer, “The task rehearsal method of lifelong learning: Overcoming impoverished data,” in *Conference of the Canadian Society for Computational Studies of Intelligence*, 2002.
- [36] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Taseli, L. Heck, H. Zhang, and C.-C. J. Kuo, “Class-incremental learning via deep model consolidation,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020.

- [37] K. Lee, K. Lee, J. Shin, and H. Lee, "Overcoming catastrophic forgetting with unlabeled data in the wild," in *Proc. IEEE Int. Conf. Comput. Vision*, 2019.
- [38] J. Lee, H. G. Hong, D. Joo, and J. Kim, "Continual learning with extended kronecker-factored approximate curvature," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
- [39] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Int. Conf. on Knowledge Discovery and Data Mining*, 2006.
- [40] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning Workshop*, 2014.
- [41] Y. Liu, A.-A. Liu, Y. Su, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
- [42] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. Int. Conf. Learn. Repres.*, 2017.
- [43] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
- [44] 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 *Proc. IEEE Int. Conf. Comput. Vision*, 2017.
- [45] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Lifelong learning via progressive distillation and retrospection," in *Proc. Eur. Conf. Comput. Vis.*, 2018.
- [46] K. Javed and F. Shafait, "Revisiting distillation and incremental classifier learning," in *Asian Conference on Computer Vision*, 2018.
- [47] O. Ostapenko, M. Puscas, T. Klein, P. Jähnichen, and M. Nabi, "Learning to remember: A synaptic plasticity driven framework for continual learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
- [48] R. Kemker and C. Kanan, "Fearnnet: Brain-inspired model for incremental learning," in *Proc. Int. Conf. Learn. Repres.*, 2018.
- [49] M. Welling, "Herding dynamical weights to learn," in *Proc. Int. Conf. Mach. Learn.*, 2009.
- [50] M. K. Titsias, J. Schwarz, A. G. d. G. Matthews, R. Pascanu, and Y. W. Teh, "Functional regularisation for continual learning with gaussian processes," in *Proc. Int. Conf. Learn. Repres.*, 2020.
- [51] P. Pan, S. Swaroop, A. Immer, R. Eschenhagen, R. E. Turner, and M. E. Khan, "Continual deep learning by functional regularisation of memorable past," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.
- [52] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, 1990.
- [53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012.
- [54] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.
- [55] B. Pfülb and A. Gepperth, "A comprehensive, application-oriented study of catastrophic forgetting in dnns," in *Proc. Int. Conf. Learn. Repres.*, 2019.
- [56] M. Masana, T. Tuytelaars, and J. van de Weijer, "Ternary feature masks: continual learning without any forgetting," *arXiv*, 2020.
- [57] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proc. Eur. Conf. Comput. Vis.*, 2018.
- [58] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
- [59] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc. Int. Conf. Mach. Learn.*, 2018.
- [60] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.
- [61] 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*, 2017.
- [62] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv*, 2016.
- [63] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification," in *ACM Int. Conf. Multim.*, 2014.
- [64] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach, "Adversarial continual learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020.
- [65] J. Rajasegaran, M. Hayat, S. Khan, F. S. Khan, and L. Shao, "Random path selection for incremental learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [66] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "Continual learning with tiny episodic memories," in *Proc. Int. Conf. Mach. Learn.*, 2019.
- [67] 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.
- [68] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [69] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [70] M. Riemer, T. Klinger, D. Bouneffouf, and M. Franceschini, "Scalable recollections for continual lifelong learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019.
- [71] 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 *Proc. Int. Conf. Learn. Repres.*, 2019.
- [72] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv*, 2018.
- [73] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *Proc. Int. Conf. Learn. Repres.*, 2018.
- [74] S. Farquhar and Y. Gal, "A unifying bayesian view of continual learning," in *NeurIPS Deep Learning Workshop*, 2019.
- [75] H. Ahn, S. Cha, D. Lee, and T. Moon, "Uncertainty-based continual learning with adaptive regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [76] Y. Chen, T. Diethe, and N. Lawrence, "Facilitating bayesian continual learning by natural gradients and stein gradients," in *NeurIPS Continual Learning Workshop*, 2018.
- [77] S. Swaroop, C. V. Nguyen, T. D. Bui, and R. E. Turner, "Improving and understanding variational continual learning," in *NeurIPS Continual Learning Workshop*, 2018.
- [78] T. Adel, H. Zhao, and R. E. Turner, "Continual learning with adaptive weights (claw)," in *Proc. Int. Conf. Learn. Repres.*, 2020.
- [79] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach, "Uncertainty-guided continual learning with bayesian neural networks," in *Proc. Int. Conf. Learn. Repres.*, 2020.
- [80] C. Zeno, I. Golan, E. Hoffner, and D. Soudry, "Task agnostic continual learning using online variational bayes," in *NeurIPS Bayesian Deep Learning Workshop*, 2018.
- [81] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, "Lifelong gan: Continual learning for conditional image generation," in *Proc. IEEE Int. Conf. Comput. Vision*, 2019.
- [82] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017.
- [83] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015.
- [84] Y. Hao, Y. Fu, Y.-G. Jiang, and Q. Tian, "An end-to-end architecture for class-incremental object detection with knowledge distillation," in *Int. Conf. on Multimedia and Expo*, 2019.
- [85] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015.
- [86] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vision Workshops*, 2019.
- [87] F. Cermelli, M. Mancini, S. R. Bulo, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
- [88] O. Tasar, Y. Tarabalka, and P. Alliez, "Incremental learning for semantic segmentation of large-scale remote sensing data," *IEEE Applied Earth Observations and Remote Sensing*, 2019.
- [89] F. Ozdemir, P. Fuernstahl, and O. Goksel, "Learn the new, keep the old: Extending pretrained models with new anatomy and images," in *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, 2018.
- [90] M. Schak and A. Gepperth, "A study on catastrophic forgetting in deep lstm networks," in *Int. Conf. on Artif. Neural Networks*, 2019.

- [91] 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*, 2013.
- [92] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," in *Proc. Int. Conf. Learn. Repres.*, 2016.
- [93] S. Sodhani, S. Chandar, and Y. Bengio, "Toward training recurrent neural networks for lifelong learning," *Neural Computation*, 2019.
- [94] 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.
- [95] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, 1993.
- [96] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [97] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
- [98] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Indian Conf. on Comput. Vis., Graphics & Image Processing*, 2008.
- [99] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009.
- [100] 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.
- [101] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proc. IEEE Int. Conf. Comput. Vision Workshops*, 2013.
- [102] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Tech. Rep., 2013.
- [103] 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 *Proc. IEEE Int. Conf. Comput. Vision*, 2011.
- [104] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *Int. Conf. on Automatic Face & Gesture Recognit.*, 2018.
- [105] 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," *Int. J. Comput. Vis.*, 2015.
- [106] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.
- [107] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Repres.*, 2015.
- [108] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015.
- [109] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
- [110] A. Chaudhry, A. Gordo, P. K. Dokania, P. Torr, and D. Lopez-Paz, "Using hindsight to anchor past knowledge in continual learning," *Proc. AAAI Conf. Artif. Intell.*, 2021.
- [111] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020.
- [112] A. Iscen, J. Zhang, S. Lazebnik, and C. Schmid, "Memory-efficient incremental learning through feature adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2020.
- [113] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *Proc. Eur. Conf. Comput. Vis.*, 2020.
- [114] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [115] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.
- [116] L. Yu, B. Twardowski, X. Liu, L. Herranz, K. Wang, Y. Cheng, S. Jui, and J. v. d. Weijer, "Semantic drift compensation for class-incremental learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
- [117] S. Li, Y. Du, G. M. van de Ven, A. Torralba, and I. Mordatch, "Energy-based models for continual learning," *arXiv*, 2020.
- [118] J. Schmidhuber, "Evolutionary principles in self-referential learning," *Diploma thesis, Tech. Univ. Munich*, 1987.
- [119] K. Javed and M. White, "Meta-learning representations for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [120] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
- [121] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, and M. Shah, "ITAML: An incremental task-agnostic meta-learning approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
- [122] M. De Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," *arXiv:2009.00919*, 2020.
- [123] F. Kunstner, L. Balles, and P. Hennig, "Limitations of the empirical fisher approximation for natural gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
- [124] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2008.
- [125] Stanford. (CS231N) Tiny imagenet challenge, cs231n course. [Online]. Available: <https://tiny-imagenet.herokuapp.com/>
- [126] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
- [127] M. Masana, B. Twardowski, and J. Van de Weijer, "On class orderings for incremental learning," in *ICML Workshop on Continual Learning*, 2020.
- [128] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. Int. Conf. Mach. Learn.*, 2009.
- [129] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference*, 2016.

APPENDIX A IMPLEMENTATION AND HYPERPARAMETERS

We study the effects of CL methods for image classification on nine different datasets whose statistics are summarized in Table S4. CIFAR-100 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. 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 performed for all datasets.

As described in Section 5.5, the Continual Hyperparameter Framework (CHF) [122] 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). When combining LwF with

TABLE S4: Summary of datasets used. We use a random 10% from the train set for validation.

Dataset	#Train	#Eval	#Classes
CIFAR-100 [97]	50,000	10,000	100
Oxford Flowers [98]	2,040	6,149	102
MIT Indoor Scenes [99]	5,360	1,340	67
CUB-200-2011 Birds [100]	5,994	5,794	200
Stanford Cars [101]	8,144	8,041	196
FGVC Aircraft [102]	6,667	3,333	100
Stanford Actions [103]	4,000	5,532	40
VGGFace2 [104]	491,746	50,000	1,000
ImageNet ILSVRC2012 [105]	1,281,167	50,000	1,000

exemplars the distillation loss is also applied to the exemplars of previous classes [1], [26], [27], [28]. 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 $\alpha = 0.5$ (chosen empirically) to avoid the storage of the importance weights for each task. The Fisher Information Matrix (FIM) is calculated by using all samples from the current task and is based on the predicted class. Referring to the definitions from [123], we have implementations of the empirical and real Fisher Information Matrix (FIM) in our experimental framework, the difference being using either a fixed label or sampling from the model’s predictive distribution when computing the FIM. In the manuscript we report results for EWC using the FIM estimated using the maximum probability class output, which is a variant described in [8]. 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 use a 32×32 resized version of Imagenet as auxiliary dataset and set its batch size to 128. The student is neither initialized from the previous

- tasks or new task models but random, as proposed in the original work.
- **GD:** we implement both the training and the sampling algorithms as described in [37]. Due to the withdrawal of Tiny Images [124] (the auxiliary dataset used in the original implementation), we use a 32×32 resized Imagenet as auxiliary dataset and we set its batch size to 128.
 - **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. More on CIFAR-100

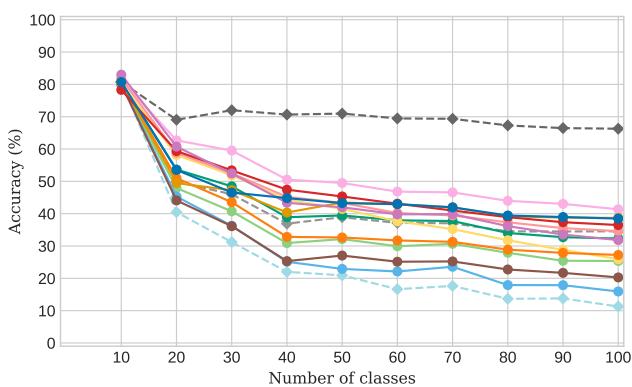
Experiments on CIFAR-100 are evaluated on 10 fixed random seeds that are the same for all approaches, to make the comparison fair. In Table S5, we show mean and standard deviation for average accuracy after learning the last task on the CIFAR-100 scenarios from Sec. 6. For most approaches and scenarios, the standard deviation seems to be below 2.5. However, some regularization-based methods (MAS-E, PathInt-E, RWalk) and iCaRL, seem to have much more variation when used in the initial larger task scenario. In the case of regularization-based methods, some runs struggle to learn new tasks properly after the initial ones, obtaining quite low performance and therefore resulting in high variability in the results. In the case of iCaRL, the variability seems to be related on how well the output features do on the initial task, since performance stays quite stable on the remaining ones. It is also notable that among the bias-correction methods, IL2M is the more stable one.

Next to the fixed memory scenario evaluated in the main paper, we here also provide results for the growing memory scenario (20 exemplars per class) with herding as the sampling strategy. In Figure S14, GD, BiC, EEIL and IL2M achieve the best results after learning 10 tasks with a growing memory, just as it was for a fixed memory. In general, most methods seem to suffer less catastrophic forgetting when using a fixed memory that allows storing more exemplars during early tasks. That is the case for BiC, GD and LUCIR, which have much better performance with a fixed memory. For some approaches, the difference is quite considerable after learning 5 tasks and slightly better after the full 10-task sequence.

B. On Random Path Selection

Although Random Path Selection (RPS) [65] is not a fixed network architecture approach, it is one of the better

◆ FT-E (34.6) ● EWC-E (25.4) ● LwM-E (32.3) ● LwF-E (27.2) ● BiC (36.5)
 ◆ FZ-E (11.3) ● MAS-E (15.9) ● DMC (25.9) ● iCaRL (34.6) ● LUCIR (31.8)
 ◆ Joint (66.3) ● RWalk (20.3) ● GD (41.3) ● EEIL (38.7) ● IL2M (38.5)



◆ FT-E (37.5) ● EWC-E (41.7) ● RWalk (35.2) ● LwF-E (34.0) ● BiC (45.1)
 ◆ FZ-E (38.9) ● MAS-E (33.5) ● LwM-E (35.9) ● iCaRL (42.4) ● LUCIR (41.7)
 ◆ Joint (65.8) ● PathInt-E (41.0) ● GD (43.5) ● EEIL (40.8) ● IL2M (40.0)

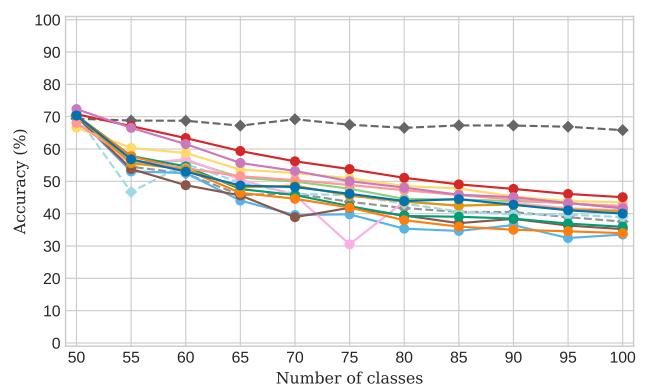


Fig. S14: CIFAR-100 (10/10) with 20 exemplars per class growing memory (left), and CIFAR-100 (50/11) with 20 exemplars per class growing memory (right).

TABLE S5: Mean and standard deviation of average accuracy over 10 runs for different CIFAR-100 scenarios.

Approach	CIFAR-100 (10/10)		CIFAR-100 (50/11)	
	fixd mem.	grow mem.	fixd mem.	grow mem.
FT-E	37.9 ± 2.1	34.6 ± 2.3	39.0 ± 1.7	37.5 ± 3.2
FZ-E	11.3 ± 0.6	11.3 ± 0.6	39.8 ± 1.3	38.9 ± 2.1
Joint	66.3 ± 2.2	66.3 ± 2.2	65.8 ± 0.0	65.8 ± 0.0
EWC-E	28.1 ± 2.2	25.4 ± 2.1	42.9 ± 1.5	41.7 ± 1.4
MAS-E	18.9 ± 2.2	15.9 ± 2.2	32.3 ± 7.8	33.5 ± 5.9
PathInt-E	18.5 ± 1.5	17.3 ± 2.2	27.4 ± 6.8	41.0 ± 0.0
RWalk	22.7 ± 1.3	20.3 ± 3.0	38.3 ± 8.5	35.2 ± 8.5
LwM-E	37.4 ± 1.7	32.3 ± 2.4	38.3 ± 1.3	35.9 ± 2.0
DMC	25.9 ± 1.3	25.9 ± 1.3	-	-
GD	44.6 ± 1.0	41.3 ± 0.7	44.7 ± 0.6	43.5 ± 1.4
LwF-E	30.8 ± 2.1	27.2 ± 2.0	36.9 ± 1.3	34.0 ± 1.2
iCaRL	33.5 ± 1.7	34.6 ± 1.3	43.4 ± 4.7	42.4 ± 5.1
EEIL	41.9 ± 3.0	38.7 ± 2.7	42.6 ± 1.0	40.8 ± 1.6
BiC	42.0 ± 2.6	36.5 ± 3.5	47.0 ± 1.1	45.1 ± 1.6
LUCIR	36.1 ± 3.5	31.8 ± 3.5	43.4 ± 3.0	41.7 ± 2.9
IL2M	41.8 ± 1.8	38.5 ± 2.2	41.0 ± 1.6	40.0 ± 1.5

performing methods from the dynamic architectures family. In Table S6 we provide a comparison with different number of paths and a range of baselines. The original CIFAR-100 (10/10) experiment was proposed with a variation of ResNet-18, however, to make it comparable with the experiments in Sec. 6 we compare it using a customized ResNet-32, which is even more memory efficient than the original ResNet-18 with 3.72M instead of 89.56M parameters. As expected, performance decreases when reducing the number of paths, making this approach very competitive if memory restrictions for the network are not an issue. With a comparable network size, it becomes less competitive in comparison to other approaches such as finetuning with exemplars (FT-E). We also report the average time per epoch for all tasks, and it is clear that the original RPS with ResNet-18 computational cost is much larger than other methods. When we change the network to ResNet-32 (with significantly fewer parameters), both performance and running time reduce dramatically, but

TABLE S6: Comparison of Random Path Selection (RPS) on CIFAR-100 (10/10) with fixed 2,000-exemplar memory.

	#paths	#params	avg. acc.	time / epoch
RPS (ResNet-18)	8	89.56M	57.0	36.4s
RPS (ResNet-32)	8	3.72M	42.1	21.1s
	4	1.86M	41.3	18.0s
	2	0.93M	37.8	13.5s
	1	0.47M	33.0	12.4s
FT-E	1	0.47M	36.5	12.1s
FZ-E	1	0.47M	10.7	11.4s
LwF-E	1	0.47M	31.8	12.8s

the running time is still much more than other methods due to the execution of different paths in parallel.

C. More 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 S7, 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.

D. 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 [97]. 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 [125]) dataset in [56], [122], where the authors show that some methods report different results based

TABLE S7: 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

on different semantics-based class orderings. In [122], the iNaturalist [126] dataset is split into tasks according to super-categories 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, [127] also brings attention to the learning variability between using different class orderings when learning a sequence of tasks incrementally.

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 [128], 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. Acc.) and the other way around (Inc. Acc.). Results are presented in Fig. S15 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

TABLE S8: 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	
WideResNet-50 66.8m params 2016	iCaRL	34.2 (-32.4)	33.4 (-22.2)	41.2 (-19.8)	42.7	
	EEIL	25.6 (-61.4)	25.8 (-55.6)	23.0 (-55.6)	37.0	
	BiC	40.8 (-41.8)	34.4 (-21.4)	54.0 (-7.4)	45.0	
	IL2M	27.4 (-53.0)	29.8 (-24.0)	41.6 (-33.2)	40.0	
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	

compared to random one. In the exemplar-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.

E. More on network architectures

We have selected the networks in the experiment from Sec. 6.6 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.

We provide more detailed results on the experiment with different architectures in Table S8. Each network architecture is evaluated on the same 10 random seeds and the accuracy and forgetting presented is an average of those runs. We also include WideResNet-50 [129] together with the ones presented in Fig. 11. BiC exhibits the least forgetting among all methods,

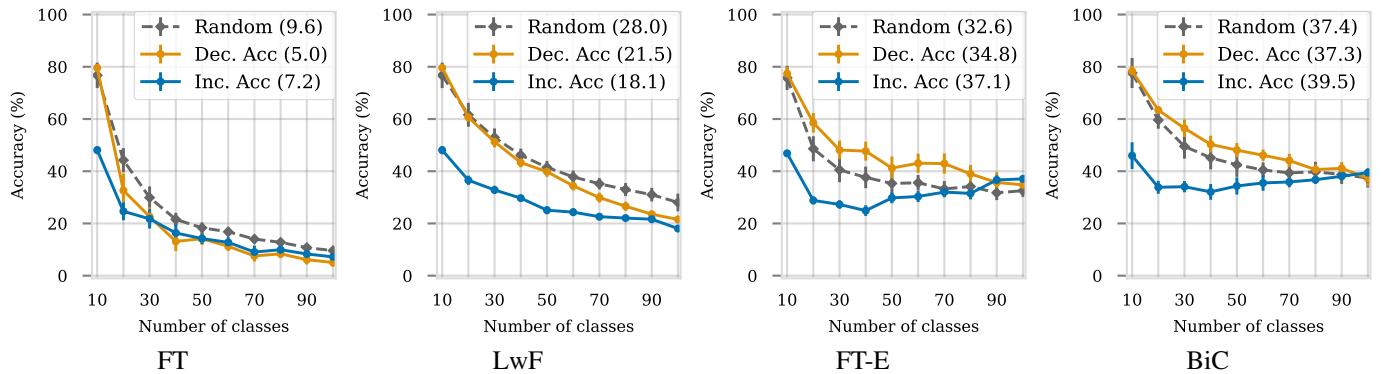


Fig. S15: Class ordering results for CIFAR-100 on ResNet-32 trained from scratch. For FT-E and BiC, 20 exemplars per class are sampled using herding. Error bars indicate standard deviation over six runs.

even having positive forgetting which indicates that performance improves on some tasks after learning subsequent ones. However, this result comes at the expense of having slightly lower performance for each task right after learning them.