Online Continual Learning in Image Classification:
An Empirical Survey

Zheda Mai[a,b,*], Ruiwen Li[a,b], Jihwan Jeong[a,c], David Quispe[a,b],
Hyunwoo Kim[d,e], Scott Sanner[a,c]

[a]*Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, 5 King's College Road, ON M5S3G8, Canada*
[b]*(zheda.mai, ruiwen.li, david.quispe)@mail.utoronto.ca*
[c]*(jhjeong, ssanner)@mie.utoronto.ca*
[d]*LG AI Research, 128, Yeoui-daero, Yeongdeungpo-gu, Seoul, South Korea*
[e]*hwkim@lgresearch.ai*

Abstract

Online continual learning for image classification studies the problem of learning to classify images from an online stream of data and tasks, where tasks may include new classes (class incremental) or data nonstationarity (domain incremental). One of the key challenges of continual learning is to avoid catastrophic forgetting (CF), i.e., forgetting old tasks in the presence of more recent tasks. Over the past few years, a large range of methods and tricks have been introduced to address the continual learning problem, but many have not been fairly and systematically compared under a variety of realistic and practical settings.

To better understand the relative advantages of various approaches and the settings where they work best, this survey aims to (1) compare state-of-the-art methods such as Maximally Interfered Retrieval (MIR), iCARL, and GDumb (a very strong baseline) and determine which works best at different memory and data settings as well as better understand the key source of CF; (2) determine if the best online class incremental methods are also competitive in the domain incremental setting; and (3) evaluate the performance of 7 simple but effective tricks such as the "review" trick and the nearest class mean (NCM) classifier to assess their relative impact. Regarding (1), we observe that iCaRL remains competitive when the memory buffer is small; GDumb outperforms many recently proposed methods in medium-size datasets and MIR performs the best in larger-scale datasets. For (2), we note that GDumb performs quite poorly while MIR — already competitive for (1) — is also strongly competitive in this very different (but important) continual learning setting. Overall, this allows us to conclude that MIR is overall a strong and versatile online continual learning method across a wide variety of settings. Finally for (3), we find that all tricks are beneficial, and when augmented with the "review" trick and NCM classifier, MIR produces performance levels that bring online continual learning much closer to its ultimate goal of matching offline training. Our codes are available at `https://github.com/RaptorMai/online-continual-learning`.

Keywords: Incremental Learning, Continual Learning, Lifelong Learning, Catastrophic Forgetting, Online Learning

1. Introduction

With the ubiquity of personal smart devices and image-related applications, a massive amount of image data is generated daily. The need for updating the deep learning model with new data on personal (i.e., edge) devices to preserve privacy, minimize communication bandwidth and maintain real-time performance necessitates the development of methods that can continually learn from streaming data while minimizing memory storage and computation footprint. However, incrementally updating a neural network with a nonstationary data stream results in *catastrophic forgetting* (CF) [1, 2] — the inability of a network to perform well on previously seen data after updating with recent data. For this reason, conventional deep learning tends to focus on offline training, where each mini-batch is sampled i.i.d from a static dataset with multiple epochs

over the training data. However, to accommodate changes in the data distribution, such a training scheme requires entirely retraining the network on the new dataset, which is inefficient and sometimes infeasible when previous data are not available due to storage limits or privacy issues.

Continual Learning (CL) studies the problem of learning from a non-i.i.d stream of data, with the goal of preserving and extending the acquired knowledge. A more complex and general viewpoint of CL is the stability-plasticity dilemma [3, 4], where stability refers to the ability to preserve past knowledge and plasticity denotes the fast adaptation of new knowledge. Following this viewpoint, CL seeks to strike a balance between learning stability and plasticity. Since CL is often used interchangeably with lifelong learning [5, 6] and incremental learning [7, 8], for simplicity, we will use CL to refer to all concepts mentioned above.

Conventional CL assumes that new data arrive one task at a time, and the data distribution for each task is stationary [9]. Thus, a model can be trained in an *offline* manner, namely mul-

*Corresponding author
    Email address:* `zheda.mai@mail.utoronto.ca` (Zheda Mai)

tiple epochs over the current task with repeat shuffle. However, this setting requires storing all data from the current task for training, which may not be feasible due to privacy issues or resource limitations.

In this survey, we focus on a more realistic but challenging problem, ***Online Continual Learning***, where data arrive one tiny batch at a time and previously seen batches from the current or the previous tasks are not accessible. Therefore, a model is required to efficiently learn from a single pass over the online data stream where the model may experience new classes (*Online Class Incremental, **OCI***) or data nonstationarity, including new background, blur, noise, illumination, and occlusion, etc.(*Online Domain Incremental, **ODI***). Moreover, there are two types of inference configuration in CL, multi-head, and single-head [8]. With the multi-head configuration, a dedicated head (classification layer) is assigned to each task, and the model just needs to classify labels within a task. Although multi-head configuration is adopted to online continual learning in some works [10, 11, 12, 13], the requirement of additional supervisory signals at inference — namely the task-ID — to select the corresponding head obviates its use when the task-ID is unavailable. Thus, in this work, we adopt the single-head configuration where the model has a shared output layer, and it needs to classify all labels without task-IDs. The axes of our setting are based on the practical CL desiderata proposed recently [14, 9, 15] and have received much attention in the past years [16, 17, 18].

To keep this paper focused, we only consider the supervised classification problem in computer vision. Although CL is also studied in reinforcement learning [19, 20, 21] and more recently in unsupervised learning [22], the image classification problem is still the main focus for many CL researchers.

Over the past few years, a broad range of methods and tricks have been introduced to address CL problems, but many have not been fairly and systematically compared under a variety of settings. To better understand the relative advantages of different approaches and the settings where they work best, this survey aims to do the following:

- We fairly compare state-of-the-art methods in OCI and determine which works best at different memory and data settings. We observe that iCaRL [7] (proposed in 2017) remains competitive when the memory buffer is small; GDumb [23] is a strong baseline that outperforms many recently proposed methods in medium-size datasets, while MIR [16] performs the best in a larger-scale dataset. Also, we experimentally and theoretically confirm that a key cause of CF is due to the recency learning bias towards new classes in the last fully connected layer owing to the imbalance between previous data and new data.

- We determine if the best OCI methods are also competitive in the ODI setting. We note that GDumb performs quite poorly in ODI, whereas MIR — already competitive in OCI — is still strongly competitive in ODI. Overall, these results allow us to conclude that MIR is a strong and versatile online CL method across a wide variety of settings.

- We evaluate the performance of 7 simple but effective tricks to assess their relative impacts. We find that all tricks are beneficial and when augmented with the "review" trick [24] and a nearest class mean (NCM) classifier [7], MIR produces performance levels that bring online CL much closer to its ultimate goal of matching offline training.

The remainder of this paper is organized as follows. Section 2 discusses the existing surveys in the CL community, and Section 3 formally defines the problem, settings and evaluation metrics. In Section 7, we explain the online continual hyperparameter tuning method we use. Then, Section 4 provides an overview of state-of-the-art CL techniques, while Section 5 gives a detailed description of methods that we compared in experiments. We discuss how class imbalance results in catastrophic forgetting and introduce CL tricks that can effectively alleviate the forgetting in Section 6. We outline our experimental setup, comparative evaluation and key findings in Section 8. Finally, Section 9 discusses recent trends and emerging directions in CL, and we conclude in Section 10.

2. Related Work

With the surge in the popularity of CL, there are multiple reviews and surveys covering the advances of CL. The first group of surveys are not empirical. [25] discusses the biological perspective of CL and summarizes how various approaches alleviate catastrophic forgetting. [14] formalizes the CL problem and outlines the existing benchmarks, metrics, approaches and evaluation methods with the emphasis on robotics applications. They also recommend some desiderata and guidelines for future CL research. [26] emphasizes the importance of online CL and discusses recent advances in this setting. Although these three surveys descriptively review the recent development of CL and provide practical guidelines, they do not perform any empirical comparison between methods.

In contrast, the second group of surveys on CL are empirical. For example, [27, 28] evaluate multiple CL methods on three CL scenarios: task incremental, class incremental and domain incremental. [15] empirically analyzes and criticizes some common experimental settings, including the multi-head configuration [8] with an exclusive output layer for each task and the use of permuted-type datasets (e.g., permuted MNIST). However, the analysis in these three works is limited to small datasets such as MNIST and Fashion-MNIST. Another two empirical studies on the performance of CL include [29, 30], but only a small number of CL methods are compared. The first extensive comparative CL survey with empirical analysis is presented in [9], which focuses on the task incremental setting with the multi-head configuration. In contrast, our work addresses a more practical and realistic setting, Online Continual Learning with single-head configuration, which require the model to learn online without access to task-ID at training and inference time. Also, we summarize and evaluate several simple but effective tricks to assess their relative impacts.

| Scenario | Difference between D_{i-1} and D_i | | | Task-ID | Online |
	$P(X_{i-1}) \neq P(X_t)$	$P(Y_{i-1}) \neq P(Y_i)$	$\{Y_{i-1}\} \neq \{Y_i\}$		
Task Incremental	✓	✓	✓	Train & Test	No
Class Incremental	✓	✓		No	Optional
Domain Incremental	✓			No	Optional

Table 1. Three continual learning scenarios based on the difference between D_{i-1} and D_i, following [27]. $P(X)$ is the input data distribution; $P(Y)$ is the target label distribution; $\{Y_{i-1}\} \neq \{Y_i\}$ denotes that output space are from a disjoint space which is separated by task-ID.

3. Problem Definition and Evaluation Metrics

3.1. Problem definition for Online Continual Learning

We consider the supervised image classification problem with an online (potentially infinite) non-i.i.d stream of data, following the recent CL literature [16, 17, 26, 14]. Formally, we define a data stream of unknown distributions $\mathcal{D} = \{D_1, \ldots, D_N\}$ over $X \times Y$, where X and Y are input and output random variables respectively, and a neural network classifier parameterized by θ, $f : X \mapsto \mathbb{R}^C$ where C is the number of classes observed so far as in [14]. At time t, a CL algorithm A^{CL} receives a mini-batch of samples (x_t^i, y_t^i) from the current distribution D_i, and the algorithm only sees this mini-batch once.

An algorithm A^{CL} is defined with the following signature:

$$A_t^{CL} : \langle f_{t-1}, (x_t, y_t), M_{t-1} \rangle \rightarrow \langle f_t, M_t \rangle \qquad (1)$$

Where:

- f_t is the classifier at time step t.

- (x_t^i, y_t^i) is a mini-batch received at time t from D_i which contains $\{(x_{tj}^i, y_{tj}^i) \mid j \in [1, \ldots, b]\}$ where b is the mini-batch size.

- M_t is an external memory that can be used to store a subset of the training samples or other useful data (e.g., the classifier from the previous time step as in LwF [31]). Note that the online setting does not limit the usage of the samples in M, and therefore, the classifier f_t can use them as many times as it wants.

Note that we assume, for simplicity, a locally i.i.d stream of data where each task distribution D_i is stationary as in [11, 10]; however, this framework can also accommodate the setting in which samples are drawn non-i.i.d from D_i as in [32, 33], where concept drift may occur within D_i.

The goal of A^{CL} is to train the classifier f to continually learn new samples from the data stream without interfering with the performance of previously observed samples. Note that unless the current samples are stored in M_t, A^{CL} will not have access to these sample in the future. Formally, at time step τ, A^{CL} tries to minimize the loss incurred by all the previously seen samples with only access to the current mini-batch and data from $M_{\tau-1}$:

$$\min_\theta \sum_{t=1}^\tau \mathbb{E}_{(x_t, y_t)} \left[\ell \left(f_\tau \left(x_t; \theta \right), y_t \right) \right] \qquad (2)$$

Recently, [27, 28] have categorized the CL problem into three scenarios based on the difference between D_{i-1} and D_i. Table 1 summarizes the differences between the three scenarios,

i.e., task incremental, class incremental and domain incremental. For task incremental, the output spaces are separated by task-IDs and are disjoint between D_{i-1} and D_i. We denote this setting as $\{Y_{i-1}\} \neq \{Y_i\}$, which in turn leads to $P(Y_{i-1}) \neq P(Y_i)$. In this setting, task-IDs are available during both train and test times. For class incremental, mutually exclusive sets of classes comprise each data distribution D_i, meaning that there is no duplicated class among different task distributions. Thus $P(Y_{i-1}) \neq P(Y_i)$, but the output space is the same for all distributions since this setting adopts the single-head configuration where the model needs to classify all labels without a task-ID. Domain incremental represents the setting where input distributions are different, while the output spaces and distribution are the same. Note that task IDs are not available for both class and domain incremental. Table 2 shows examples of these three scenarios. Following this categorization, the settings we focus on in this work are known as Online Class Incremental (OCI) and Online Domain Incremental (ODI) with the single-head configuration. Note that some works consider the relaxed version of OCI with the multi-head configuration [11, 10, 13], especially for methods without using memory buffer [34, 12, 13].

3.2. Evaluation metrics

Besides measuring the final accuracy across tasks, it is also critical to assess how fast a model learns, how much the model forgets and how well the model transfers knowledge from one task to another. To this end, we use five standard metrics in the CL literature to measure performance: (1) the average accuracy for overall performance [10]; (2) the average forgetting to measure how much of the acquired knowledge the model has forgotten [8]; (3) the forward transfer and (4) the backward transfer to assess the ability for knowledge transfer [14, 11]; (5) the total running time, including training and testing times.

Formally, we define $a_{i,j}$ as the accuracy evaluated on the held-out test set of task j after training the network from task 1 through to i, and we assume there are T tasks in total.

Average Accuracy can be defined as Eq. (3). When $i = T$, A_T represents the average accuracy by the end of training with the whole data sequence (see example in Table 3).

$$\text{Average Accuracy}(A_i) = \frac{1}{i} \sum_{j=1}^i a_{i,j} \qquad (3)$$

Average Forgetting at task i is defined as Eq. (4). $f_{i,j}$ represents how much the model has forgot about task j after being trained on task i. Specifically, $\max_{l \in \{1, \cdots, k-1\}} (a_{l,j})$ denotes the best test accuracy the model has ever achieved on task j before learning task k, and $a_{k,j}$ is the test accuracy on task j after learning task k.
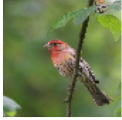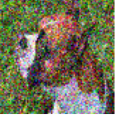
Task	Task Incremental		Class Incremental		Domain Incremental	
D_{i-1}	x:		x:		x:	
	y: Bird	Dog	y: Bird	Dog	y: Bird	Dog
task-ID(test)	**i-1**		**Unknown**		**Unknown**	
D_i	x:		x:		x:	
	y: Ship	Guitar	y: Ship	Guitar	y: Bird	Dog
task-ID(test)	**i**		**Unknown**		**Unknown**	

Table 2. Examples of the three CL scenarios. (x, y, task-ID) represents (input images, target label and task identity). The main distinction between task incremental and class incremental is the availability of task-ID. The main difference between class incremental and domain incremental is that, in class incremental, a new task contains completely new classes, whereas domain incremental, a new task consists of new instances with nonstationarity (e.g., noise) of all the seen classes.

a	te_1	te_2	$\ldots$	te_{T-1}	te_T
tr_1	$a_{1,1}$	$a_{1,2}$	$\ldots$	$a_{1,T-1}$	$a_{1,T}$
tr_2	$a_{2,1}$	$a_{2,2}$	$\ldots$	$a_{2,T-1}$	$a_{2,T}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
tr_{T-1}	$a_{T-1,1}$	$a_{T-1,2}$	$\ldots$	$a_{T-1,T-1}$	$a_{T-1,T}$
tr_T	$a_{T,1}$	$a_{T,2}$	$\ldots$	$a_{T,T-1}$	$a_{T,T}$

Table 3. Accuracy matrix example following the notations in [14]. tr_i and te_i denote training and test set of task i. A_T is the average of accuracies in the box. BWT^+ is the average of accuracies in purple and FWT is the average of accuracies in green.

$$\text{Average Forgetting}(F_i) = \frac{1}{i-1} \sum_{j=1}^{i-1} f_{i,j}$$

$$\text{where } f_{k,j} = \max_{l \in \{1, \cdots, k-1\}} (a_{l,j}) - a_{k,j}, \forall j < k \quad (4)$$

Positive Backward Transfer(BWT^+) measures the positive influence of learning a new task on preceding tasks' performance (see example in Table 3).

$$BWT^+ = max\left(\frac{\sum_{i=2}^{T} \sum_{j=1}^{i-1} \left(a_{i,j} - a_{j,j}\right)}{\frac{T(T-1)}{2}}, 0\right) \quad (5)$$

Forward Transfer(FWT) measures the positive influence of learning a task on future tasks' performance (see example in Table 3).

$$FWT = \frac{\sum_{i=1}^{j-1} \sum_{j=1}^{T} a_{i,j}}{\frac{T(T-1)}{2}} \quad (6)$$

4. Overview of Continual Learning Techniques

Although a broad range of methods have been introduced in the past few years to address the CL problem, the assumptions that each method makes are not consistent due to the plethora of settings in CL. In particular, some methods have a better ability to generalize to different CL settings because they require less

supervisory signals during both training and inference times. For example, ER [35] was designed for the task incremental setting, but the method can easily be used in all the other CL settings since it does not need any additional supervisory signals. In this sense, a systematic summary of supervisory signals that different methods demand will help understand the generalization capacity and limitations of the methods. Furthermore, the summary will facilitate fair comparison in the literature. On the other hand, CL methods have typically been taxonomized into three major categories based on the techniques they use: regularization-based, memory-based and parameter-isolation-based [9, 25]. A clear trend in recent works, however, is to simultaneously apply multiple techniques in order to tackle the CL problem. In this section, we comprehensively summarize recently proposed methods based on techniques they use and supervisory signals required at training and inference times (see Table 4).

Supervisory Signals. The most important supervisory signal is the task-ID. When a task-ID is available, a training or testing sample is given as (x, y, t) instead of (x, y) where t is the task-ID. For the task incremental setting, task-IDs are available at both training and inference times. In regards to the class incremental setting, a task-ID is not available at the inference time but can be inferred during training as each task has disjoint class labels. On the other hand, in the domain incremental setting, a task-ID is not available at all times. Other supervisory signals include a natural language description of a task or a matrix specifying the attribute values of the objects to be recognized in the task [10].

Moreover, a method is *online-able* if it does not need to revisit samples it has processed before. Hence, to be online-able, the model needs to learn efficiently from one pass of the data. For example, the herding-based memory update strategy proposed in iCaRL [7] needs all the samples from a class to select a representative set; therefore, methods using this strategy are not online-able.

Techniques. **Regularization** techniques impose constraints on the update of network parameters to mitigate catastrophic forgetting. This is done by either incorporating additional penalty

4

terms into the loss function [36, 37, 38, 39] or modifying the gradient of parameters during optimization [11, 10, 40].

Knowledge Distillation (KD) [41] is an effective way for knowledge transfer between networks. KD has been widely adopted in CL methods [42, 31, 43, 44], and it is often considered as one of the regularization techniques. Due to the prevalence of KD in CL methods, we list it as a separate technique in Table 4. One shortcoming of regularization-based techniques including KD is that it is difficult to strike a balance between the regularization and the current learning when learning from a long stream of data.

Memory-based techniques store a subset of samples from previous tasks for either replay while training on new task [35, 16, 17] or regularization purpose [45, 11, 46]. These methods become infeasible when storing raw samples is not possible due to privacy or storage concerns.

Instead of saving raw samples, an alternative is **Generative Replay** which trains a deep generative model such as GAN [47] to generate pseudo-data that mimic past data for replay [48, 49, 50]. The main disadvantages of generative replay are that it takes long time to train such generative models and that it is not a viable option for more complex datasets given the current state of deep generative models [51, 16].

Parameter-isolation (PI)-based techniques bypass interference by allocating different parameters to each task. PI can be subdivided into Fixed Architecture (FA) that only activates relevant parameters for each task without modifying the architecture [52, 53, 54], and Dynamic Architecture (DA) that adds new parameters for new tasks while keeping old parameters unchanged [55, 56, 57]. Most previous works require task-IDs at inference time, and a few recent methods have been introduced to predict without a task-ID [18, 58].

For a more detailed discussion of these techniques, we refer readers to the recent CL surveys [9, 25, 14].

5. Compared Methods

5.1. Regularization-based methods

Elastic Weight Consolidation (EWC). EWC [84] incorporates a quadratic penalty to regularize the update of model parameters that were important to past tasks. The importance of parameters is approximated by the diagonal of the Fisher Information Matrix F. Assuming a model sees two tasks A and B in sequence, the loss function of EWC is:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_j \frac{\lambda}{2} F_j \left(\theta_j - \theta_{A,j}^*\right)^2 \tag{7}$$

where $\mathcal{L}_B(\theta)$ is the loss for task B, $\theta_{A,j}^*$ is the optimal value of jth parameter after learning task A and λ controls the regularization strength. There are three major limitations of EWC: (1) It requires storing the Fisher Information Matrix for each task, which makes it impractical for a long sequence of tasks or models with millions of parameters. (2) It needs an extra pass over each task at the end of training, leading to its infeasibility for the online CL setting. (3) Assuming the Fisher to be diagonal may

not be accurate enough in practice. Several variants of EWC are proposed lately to address these limitations [85, 8, 86]. As we use the online CL setting, we compare EWC++, an efficient and online version of EWC that keeps a single Fisher Information Matrix calculated by moving average. Specifically, given F^{t-1} at $t-1$, the Fisher Information Matrix at t is updated as:

$$F^t = \alpha F_{tmp}^t + (1 - \alpha) F^{t-1} \tag{8}$$

where F_{tmp}^t is the Fisher Information Matrix calculated with the current batch of data and $\alpha \in [0, 1]$ is a hyperparameter controlling the strength of favouring the current F^t.

Learning without Forgetting (LwF). LwF [31] utilizes knowledge distillation [41] to preserve knowledge from past tasks in the multi-head setting [8]. In LwF, the teacher model is the model after learning the last task, and the student model is the model trained with the current task. Concretely, when the model receives a new task (X_n, Y_n), LwF computes Y_o, the output of old tasks for the new data X_n. During training, LwF optimizes the following loss:

$$\mathcal{L}(\theta) = \left(\lambda_o \mathcal{L}_{KD}\left(Y_o, \hat{Y}_o\right) + \mathcal{L}_{CE}\left(Y_n, \hat{Y}_n\right) + \mathcal{R}(\theta)\right) \tag{9}$$

where $\hat{Y}_o$ and $\hat{Y}_n$ are the predicted values of the old task and new task using the same X_n. $\mathcal{L}_{KD}$ is the knowledge distillation loss incorporated to impose output stability of old tasks with new data and $\mathcal{L}_{CE}$ is the cross-entropy loss for the new task. $\mathcal{R}$ is a regularization term, and λ_o is a hyperparameter controlling the strength of favouring the old tasks over the new task. A known shortcoming of LwF is its heavy reliance on the relatedness between the new and old tasks. Thus, LwF may not perform well when the distributions of the new and old tasks are different [31, 7, 57]. To apply LwF in the single-head setting where all tasks share the same output head, a variant of LwF (LwF.MC) is proposed in [7], and we evaluate LwF.MC in this work.

5.2. Memory-based methods

A generic online memory-based method is presented in Algorithm 1. For every incoming mini-batch, the algorithm retrieves another mini-batch from a memory buffer, updates the model using both the incoming and memory mini-batches and then updates the memory buffer with the incoming mini-batch. What differentiate various memory-based methods are the memory retrieval strategy (line 3) [16, 87], model update (line 4) [10, 11] and memory update strategy (line 5) [62, 64, 17].

Averaged GEM (A-GEM). A-GEM [10] is a more efficient version of GEM [11]. Both methods prevent forgetting by constraining the parameter update with the samples in the memory buffer. At every training step, GEM ensures that the loss of the memory samples for each individual preceding task does not increase, while A-GEM ensures that the average loss for all past tasks does not increase. Specifically, let g be the gradient computed with the incoming mini-batch and g_{ref} be the gradient computed with the same size mini-batch randomly selected

5

Methods	Settings			Techniques					
	t-ID free(test)	t-ID free(train)	Online-able	Reg	Mem	KD	PI(FA)	PI(DA)	Generative
MIR[16], GSS[17], ER[35], CBO[59]									
GDumb[23], DER[60], MER[61]					✓				
CBRS[62], GMED[63], PRS[64]									
La-MAML[65], MEFA[66]									
MERLIN[67]					✓		✓		
CN-DPM[18], TreeCNN[68]								✓	
A-GEM[10], GEM[11], VCL [45]	✓	✓	✓	✓	✓				
WA[69], BiC[43], LUCIR[70]					✓	✓			
IL2M[71], ILO[72]									
LwF-MC[7], LwM[73], DMC[74]						✓			
SRM[75], AQM[76]					✓				✓
EWC++[8]				✓					
AR1[77]				✓			✓		
EEIL[42], iCaRL[7], MCIL[78]					✓	✓			
SDC[79]	✓	✓	✗	✓					
DGR[48]									✓
DGM[80]							✓	✓	✓
ICGAN[81], RtF[50]						✓			✓
iTAML[82],CCG[83]	✓	✗	✓	✓			✓		

Table 4. Summary of recently proposed CL methods based on supervisory signals required and techniques they use. t-ID free(test/train) means task-ID is not required at test/train time.

Algorithm 1: Generic online Memory-based method

Input : Batch size b, Learning rate α
Initialize: Memory $\mathcal{M} \leftarrow \{\} * M$; Parameters θ; Counter $n \leftarrow 0$
1 **for** $t \in \{1, \ldots, T\}$ **do**
2   **for** $B_n \sim D_t$ **do**
3          $B_{\mathcal{M}} \leftarrow MemoryRetrieval(B_n, \mathcal{M})$
4          $\theta \leftarrow ModelUpdate(B_n \cup B_{\mathcal{M}}, \theta, \alpha)$
5          $\mathcal{M} \leftarrow MemoryUpdate(B_n, \mathcal{M})$
6          $n \leftarrow n + b$

7 **return** θ

from the memory buffer. In A-GEM, if $g^T g_{ref} \geq 0$, g is used for gradient update but when $g^T g_{ref} < 0$, g is projected such that $g^T g_{ref} = 0$. The gradient after projection is:

$$\tilde{g} = g - \frac{g^\top g_{ref}}{g_{ref}^\top g_{ref}} g_{ref} \qquad (10)$$

As we can see, A-GEM focuses on *ModelUpdate* in Algorithm 1, and we apply reservoir sampling [88] in *MemoryUpdate* and random sampling in *MemoryRetrieval* for A-GEM.

Incremental Classifier and Representation Learning (iCaRL). iCaRL [7] is a replay-based method that decouples the representation learning and classification. For representation learning, the training set is constructed by mixing all the samples in the memory buffer and the current task samples.

The loss function includes a classification loss to encourage the model to predict the correct labels for new classes and a KD loss to prompt the model to reproduce the outputs from the previous model for old classes. Note that the training set is imbalanced since the number of new-class samples in the current task is larger than that of the old-class samples in the memory buffer. iCaRL applies the binary cross entropy (BCE) for each class to handle the imbalance, but BCE may not be effective in addressing the relationship between classes. For the classifier, iCaRL uses a nearest-class-mean classifier [89] with the memory buffer to predict labels for test images. Moreover, it proposes a *MemoryUpdate* method based on the distance in the latent feature space with the inspiration from [90]. For each class, it looks for a subset of samples whose mean of latent features are closest (in the Euclidean distance) to the mean of all the samples in this class. However, this method requires all samples from every class, and therefore it cannot be applied in the online setting. As such, we modify iCaRL to use reservoir sampling [88], which has been shown effective for *MemoryUpdate* [35].

Experience Replay (ER). ER refers to a simple but effective replay-based method that has been discussed in [35, 91]. It applies reservoir sampling [88] in *MemoryUpdate* and random sampling in *MemoryRetrieval*. Reservoir sampling ensures every streaming data point has the same probability, mem_{sz}/n, to be stored in the memory buffer, where mem_{sz} is the size of the

buffer and n is the number of data points observed up to now. We summarize the detail in Algorithm A1 in Appendix A. For *ModelUpdate*, ER simply trains the model with the incoming and memory mini-batches together using the cross-entropy loss. Despite its simplicity, recent research has shown that ER outperforms many specifically designed CL approaches with and without a memory buffer [35].

Maximally Interfered Retrieval (MIR). MIR [16] is a recently proposed replay-based method aiming to improve the *MemoryRetrieval* strategy. MIR chooses replay samples according to the loss increases given the estimated parameter update based on the incoming mini-batch. Concretely, when receiving a mini-batch B_n, MIR performs a virtual parameter update $\theta^v \leftarrow \text{SGD}(B_n, \theta)$. Then it retrieves the top-k samples from the memory buffer with the criterion $s(x) = l(f_{\theta^v}(x), y) - l(f_\theta(x), y)$, where $x \in M$ and M is the memory buffer. Intuitively, MIR selects memory samples that are maximally interfered (the largest loss increases) by the parameter update with the incoming mini-batch. MIR applies reservoir sampling in *MemoryUpdate* and replays the selected memory samples with new samples in *ModelUpdate*.

Gradient based Sample Selection (GSS). GSS [17] is another replay-based method that focuses on the *MemoryUpdate* strategy[1]. Specifically, it tries to diversify the gradient directions of the samples in the memory buffer. To this end, GSS maintains a score for each sample in the buffer, and the score is calculated by the maximal cosine similarity in the gradient space between the sample and a random subset from the buffer. When a new sample arrives and the memory buffer is full, a randomly selected subset is used as the candidate set for replacement. The score of a sample in the candidate set is compared to the score of the new sample, and the sample with a lower score is more likely to be stored in the memory buffer. Algorithm A2 in Appendix A shows the main steps of this update method. Same as ER, GSS uses random sampling in *MemoryRetrieval*.

Greedy Sampler and Dumb Learner (GDumb). GDumb [23] is not specifically designed for CL problems but shows very competitive performance. Specifically, it greedily updates the memory buffer from the data stream with the constraint to keep a balanced class distribution (Algorithm A3 in Appendix A). At inference, it trains a model from scratch using the balanced memory buffer only.

5.3. Parameter-isolation-based methods

Continual Neural Dirichlet Process Mixture (CN-DPM). CN-DPM [18] is one of the first dynamic architecture methods that does not require a task-ID. The intuition behind this method is that if we train a new model for a new task and leave the existing models intact, we can retain the knowledge of the past tasks. Specifically, CN-DPM is comprised of a group of experts, where each expert is responsible for a subset of the

data and the group is expanded based on the Dirichlet Process Mixture [92] with Sequential Variational Approximation [93]. Each expert consists of a discriminative model (classifier) and a generative model (VAE [94] is used in this work). The goal of CN-DPM is to model the overall conditional distribution as a mixture of task-wise conditional distributions as the following, where K is the number of experts in the current model:

$$p(y \mid x) = \sum_{k=1}^{K} \underbrace{p(y \mid x, z = k)}_{\text{discriminative}} \frac{\overbrace{p(x \mid z = k)}^{\text{generative}} p(z = k)}{\sum_{k'=1}^{K} p(x \mid z = k') \, p(z = k')} \tag{11}$$

6. Tricks for Memory-Based Methods in the OCI Setting

In class incremental learning, old class samples are generally not available while training on new class samples. Although keeping a portion of old class samples in a memory buffer has been proven effective [35, 7], the *class imbalance* is still a serious problem given a limited buffer size. Moreover, multiple recent works have revealed that class imbalance is one of the most crucial causes of catastrophic forgetting [42, 43, 70]. To alleviate the class imbalance, many simple but effective tricks have been proposed as the building blocks for CL methods by modifying the loss function, post-processing, or using different types of classifiers.

We will disclose how class imbalance results in recency bias, which has been identified as the main source of catastrophic forgetting in Section 6.1. Section 6.2 will introduce the compared tricks in detail and explain how they alleviate recency bias.

6.1. Biased FC layer

To better understand how class imbalance affects the learning performance, we define the following notations. The convolutional neural network (CNN) classification model can be split into a feature extractor $\phi(\cdot) : \mathbf{x} \mapsto \mathbb{R}^d$ where d is the dimension of the feature vector of image $\mathbf{x}$, and a fully-connected (FC) layer with Softmax output. The output of the FC layer is obtained with:

$$logits(\mathbf{x}) = \mathbf{W}^T \phi(\mathbf{x}) \tag{12}$$
$$\mathbf{o}(\mathbf{x}) = Softmax(logits(\mathbf{x}))) \tag{13}$$

where $\mathbf{W} \in \mathbb{R}^{d \times (|C_{old}| + |C_{new}|)}$, and C_{old} and C_{new} are the sets of old and new classes respectively, with $|\cdot|$ denoting the number of classes in each set. $\mathbf{W_i}$ represents the weight vector for class i. For notational brevity, $\mathbf{W}$ also contains the bias terms.

A recent work reveals how does class imbalance result in a biased $\mathbf{W}$ [95]. We denote $s_i = \mathbf{W_i}^T \phi(\mathbf{x})$ as the logit of sample $\mathbf{x}$ for class i. The gradient of the cross-entropy loss L_{CE} with respect to $\mathbf{W_i}$ for the Softmax output is:

$$\frac{\partial \mathcal{L}_{CE}}{\partial \mathbf{W_i}} = \left(\frac{e^{s_i}}{\sum_{j \in C_{old} + C_{new}} e^{s_j}} - \mathbb{1}_{\{i=y\}} \right) \phi(\mathbf{x}) \tag{14}$$

where y is the ground-truth class and $\mathbb{1}_{\{i=y\}}$ is the indicator for $i = y$. Since ReLU is often used as the activation function for

---

[1] [17] proposes two gradient-based methods, and we select the more efficient one with better performance, dubbed GSS-Greedy

the embedding networks [96], $\phi(\mathbf{x})$ is always positive. Therefore, the gradient is always positive for $i \neq y$. If i belongs to old classes, $i \neq y$ will hold most of the time as the new class samples significantly outnumber the old class samples during training. Thus, the logit for class i will keep being penalized during the gradient descent. As a result, the logits for the old classes are prone to be smaller than those for the new classes, and the model is consequently biased towards new classes.

We will perform quantitative error analysis in Section 8.3 and empirically show that the logits for old classes are significantly smaller than the logits for new classes.

Other than the bias related to the Softmax classifier mentioned above, another problem induced by class imbalance is under-representation of the minority [97, 98], where minority classes do not show a discernible pattern in the latent feature space [64]. The under-representation introduces additional difficulty for other classifiers apart from the Softmax classifier, such as nearest-class-mean [89] and cosine-similarity-based classifier [99].

6.2. Compared tricks

To mitigate the strong bias towards new classes due to class imbalance, multiple methods have been proposed lately [100, 101, 69, 71, 102]. For example, LUCIR [70] proposes three tricks: cosine normalization to balance class magnitudes, a margin ranking loss for inter-class separation, and a less-forgetting constraint to preserve the orientation of features. BiC [43] trains an additional linear layer to remove bias with a separate validation set. We compare seven simple (effortless integration without additional resources) but effective (decent improvement) tricks in this work.

Labels Trick (LB). [103] proposes to consider only the outputs for the classes in the current mini-batch when calculating the cross-entropy loss, in contrast to the common practice of considering outputs for all the classes. To achieve this, the outputs that do not correspond to the classes of the current mini-batch are masked out when calculating the loss.

Although the author did not demonstrate the motivation of this trick, we can easily find the rationale based on the analysis in Section 6.1. Masking out all the outputs that don't match the classes in the current mini-batch is equivalent to changing the loss function to:

$$\mathcal{L}_{\text{CE}}(x_i, y_i) = -\log\left(\frac{e^{s_{y_i}}}{\sum_{j \in C_{cur}} e^{s_j}}\right) \tag{15}$$

where C_{cur} denotes the classes in the current mini-batch. We can see that $\frac{\partial \mathcal{L}_{\text{CE}}}{\partial s_j} = 0$ for $j \notin C_{cur}$, and therefore training with the current mini-batch will not overly penalize the logits for classes that are not in the mini-batch.

Knowledge Distillation with Classification Loss (KDC). KD [41] is an effective way for knowledge transfer between networks. Multiple recent works have proposed different ways to combine the KD loss with the classification loss [7, 42, 104].

In this part, we compare the methods from [43]. Specifically, the loss function is given as:

$$\mathcal{L}(\mathbf{x}, y) = \lambda \mathcal{L}_{CE}(\mathbf{x}, y) + (1 - \lambda)\mathcal{L}_{KD}(\mathbf{x}) \tag{16}$$

where λ is set to $\frac{|C_{new}|}{|C_{old}| + |C_{new}|}$. Note that $(\mathbf{x}, y)$ is from both new class data from the current task and old class data from the memory buffer.

As shown in Table 9, however, this method does not perform well in our experiment setting, especially with a large memory buffer. We identify λ as the key issue. We find that the accuracy for new class samples becomes almost zero around the end of training because λ is very small. In other words, $\mathcal{L}_{KD}$ dominates the loss, and the model cannot learn any new knowledge. Hence, we suggest setting λ to $\sqrt{\frac{|C_{new}|}{|C_{old}| + |C_{new}|}}$. We denote the trick with this modification as KDC*.

Multiple Iterations (MI). Most of the previous works only perform a single gradient update on the incoming mini-batch in the online setup. [16] suggests performing multiple gradient updates to maximally utilize the current mini-batch. Particularly for replay methods, additional updates with different replay samples can improve performance. We run 5 iterations per incoming mini-batch and retrieve different memory samples for each iteration in this work.

Nearest Class Mean (NCM) Classifier. To tackle the biased FC layer, one can replace the FC layer and Softmax classifier with another type of classifier. Nearest Class Mean classifier (NCM) [89] is a popular option in CL [7, 79]. To make prediction for a sample $\mathbf{x}$, NCM computes a prototype vector for each class and assigns the class label with the most similar prototype:

$$\mu_y = \frac{1}{|M_y|} \sum_{\mathbf{x}_m \in M_y} \phi(\mathbf{x}_m) \tag{17}$$

$$y^* = \underset{y=1,\ldots,t}{\arg\min} \left\| \phi(\mathbf{x}) - \mu_y \right\| \tag{18}$$

In the class incremental setting, the true prototype vector for each class cannot be computed due to the unavailability of the training data for previous tasks. Instead, the prototype vectors can be approximated using the data in the memory buffer. In Eq. (17), M_y denotes the memory samples of class y.

Separated Softmax (SS). Since training the whole FC layer with one Softmax output layer results in bias as explained in Section 6.1, SS [95] employs two Softmax output layers: one for new classes and another one for old classes. The loss function can be calculated as below:

$$\begin{aligned} &\mathcal{L}(\mathbf{x}_i, y_i) \\ &= -\log\left(\frac{e^{s_{y_i}}}{\sum_{j \in C_{old}} e^{s_j}}\right) \cdot \mathbb{1}\{y_i \in C_{old}\} \\ &\quad -\log\left(\frac{e^{s_{y_i}}}{\sum_{j \in C_{new}} e^{s_j}}\right) \cdot \mathbb{1}\{y_i \in C_{new}\} \end{aligned} \tag{19}$$

Depending on whether $y_i \in C_{new}$ or C_{old}, the corresponding Softmax is used to compute the cross-entropy loss. We can find

that $\frac{\partial \mathcal{L}}{\partial s_j} = 0$ for $j \in C_{old}$ and $y_i \in C_{new}$. Thus, training with new class samples will not overly penalize the logits for the old classes.

Review Trick (RV). To alleviate the class imbalance, [42] proposes an additional fine-tuning step with a small learning rate, which uses a balanced subset from the memory buffer and the training set of the current task. A temporary distillation loss for new classes is applied to avoid forgetting the new classes during the fine-tuning phase mentioned above. A similar yet simplified version, dubbed Review Trick, is applied in the winning solution in the continual learning challenge at CVPR2020 [24]. At the end of learning the current task, the review trick fine-tunes the model with all the samples in the memory buffer using only the cross-entropy loss. In this work, we compare the review trick from [24] with a learning rate 10 times smaller than the training learning rate.

7. Online Continual Hyperparameter Tuning

In practice, most CL methods have to rely on well-selected hyperparameters to effectively balance the trade-off between stability and plasticity. Hyperparameter tuning, however, is already a challenging task in learning conventional deep neural network models, which becomes even more complicated in the online CL setting. Meanwhile, a large volume of CL works still tune hyperparameters in an offline manner by sweeping over the whole data sequence and selecting the best hyperparameter set with grid-search on a validation set. After that, metrics are reported on the test set with the selected set of hyperparameters. This tuning protocol violates the online CL setting where a classifier can only make a single pass over the data, which implies that the reported results in the CL literature may be too ideal and cannot be reproduced in real online CL applications.

Recently, several hyperparameter tuning protocols that are useful for CL settings have been proposed. [29] introduces a tuning protocol for two tasks (D_1 and D_2). Firstly, they determine the best combination of model hyperparameters using D_1. Then, they tune the learning rate to be used for learning D_2 such that the test accuracy on D_2 is maximized. [9] proposes another protocol that dynamically determines the stability-plasticity trade-off. When the model receives a new task, the hyperparameters are set to ensure minimal forgetting of previous tasks. If a predefined threshold for the current task's performance is not met, the hyperparameters are adjusted until achieving the threshold. While the aforementioned protocols assume the data of a new task is available all at once, [10] presents a protocol targeting the online CL. Specifically, a data stream is divided into two sub-streams — D^{CV}, the stream for cross-validation and D^{EV}, the stream for final training and evaluation. Multiple passes over D^{CV} are allowed for tuning, but a CL algorithm can only perform a single pass over D^{EV} for training. The metrics are reported on the test sets of D^{EV}. Since the setting of our interest in this work is the online CL, we adopt the protocol from [10] for our experiments. We summarize the protocol in Algorithm 2.

8. Experiments

Section 8.1 explains the general setting for all experiments. Then, we focus on the OCI setting in Section 8.2 and 8.4: we evaluate all the compared methods and baselines in Section 8.2 and investigate the effectiveness of the seven tricks in Section 8.4. In Section 8.5, we assess the compared methods in the ODI setting to investigate their abilities to generalize, and Section 8.6 provides general comments on the surveyed methods and tricks.

8.1. Experimental setup

Datasets. We evaluate the nine methods summarized in Section 5 and additional two baselines on three class incremental datasets. We also propose a new domain incremental dataset based on Mini-ImageNet and examine the compared methods in the ODI setting to see how well the methods can generalize to this setting. The summary of dataset statistics is provided in Table 5.

Class Incremental Datasets.

- **Split CIFAR-100** is constructed by splitting the CIFAR-100 dataset [105] into 20 tasks with disjoint classes, and each task has 5 classes. There are 2,500 3×32×32 images for training and 500 images for testing in each task.

- **Split MiniImageNet** splits MiniImageNet dataset [106], a subset of ImageNet [107] with 100 classes, into 20 disjoint tasks as in [35]. Each task contains 5 classes, and every class consists of 500 3×84×84 images for training and 100 images for testing.

- **CORe50-NC** [108] is a benchmark designed for class incremental learning with 9 tasks and 50 classes: 10 classes in the first task and 5 classes in the subsequent 8 tasks. Each class has around 2,398 3×128×128 training images and 900 testing images.

Domain Incremental Datasets.

- **NonStationary-MiniImageNet** (NS-MiniImageNet) The most popular domain incremental datasets are still based on MNIST [109], such as Rotation MNIST [11] and Permutation MNIST [84]. To evaluate the domain incremental setting in a more practical scenario, we propose NS-MiniImageNet with three nonstationary types: noise, blur and occlusion. The number of tasks and the strength of each nonstationary type are adjustable in this dataset. In this survey, we use 10 tasks for each type, and each task comprises 5,000 3×84×84 training images and 1,000 testing images. As shown in Table 6, the nonstationary strength increases over time, and to ensure a smooth distribution shift, the strength always increases by the same constant. More details about the nonstationary strengths used in the experiments can be found in Appendix B.

---

[2]CORe50-NI uses one test set(44972 images) for all tasks

Dataset	#Task	#Train/task	#Test/task	#Class	Image Size	Setting
Split MiniImageNet	20	2500	500	100	3x84x84	OCI
Split CIFAR-100	20	2500	500	100	3x32x32	OCI
CORe50-NC	9	12000~24000	4500~9000	50	3x128x128	OCI
NS-MiniImageNet	10	5000	1000	100	3x84x84	ODI
CORe50-NI[2]	8	15000	44972	50	3x128x128	ODI

Table 5. Summary of dataset statistics

Algorithm 2: Hyperparameter Tuning Protocol

Input : Hyperparameter set $\mathcal{P}$
Require: D^{CV} data stream for tuning, D^{EV} data stream for learning & testing
Require: f classifier, A^{CL} CL algorithm

1   **for** $p \in \mathcal{P}$ **do**   ▷ Multiple passes over D^{CV} for tuning
2   **for** $i \in \{1, \ldots, T^{CV}\}$ **do**   ▷ Single pass over D^{CV} with p
3   **for** $B_n \sim D_i^{CV}$ **do**
4              $A^{CL}(f, B_n, p)$
5   Evaluate(f, D_{test}^{CV}) ▷ Store performance on test set of D^{CV}

6   Best hyperparameters, $\boldsymbol{p^*} \leftarrow$ based on Average Accuracy of D_{test}^{CV}, see Eq.(3)
7   **for** $i \in \{1, \ldots, T^{EV}\}$ **do**   ▷ Learning over D^{EV}
8   **for** $B_n \sim D_i^{EV}$ **do**   ▷ Single pass over D^{EV} with p^*
9          $A^{CL}(f, B_n, \boldsymbol{p^*})$
10   Evaluate(f, D_{test}^{EV})
11   Report performance on D_{test}^{EV}

- **CORe50-NI** [108] is a practical benchmark designed for assessing the domain incremental learning with 8 tasks, where each task has around 15,000 training images of 50 classes with different types of nonstationarity including illumination, background, occlusion, pose and scale. There is one single test set for all tasks, which contains 44,972 images.

Type	Task 1	...	Task 5	...	Task 10
Noise		...		...	
Blur		...		...	
Occlusion		...		...	

Table 6. Example images of different nonstationary types in NS-MiniImageNet. The nonstationary strength increases over time, and to ensure a smooth distribution shift, the strength always increases by the same constant.

Task Order and Task Composition. Since the task order and task composition may impact the performance [108], we take the average over multiple runs for each experiment with different task orders and composition to reliably assess the robustness of the methods. For CORe50-NC and CORe50-NI, we follow the number of runs (i.e., 10), task order and composition provided by the authors. For Split CIFAR-100 and Split MiniIm-

agenet, we average over 15 runs, and the class composition in each task is randomly selected for each run.

Models. Similar to [35, 11], we use the reduced ResNet18 [110] as the base model for all datasets and methods. The network is trained via the cross-entropy loss with a stochastic gradient descent optimizer and a mini-batch size of 10. The size of the mini-batch retrieved from the memory buffer is also set to 10, irrespective of the size of the memory buffer as in [35]. Note that with techniques such as transfer learning (e.g., using a pre-trained model from ImageNet), data augmentation and deeper network architectures, it is possible to achieve much higher performance in this setting [111]. However, since those techniques are orthogonal to our investigation and deviate from the simpler experimental settings of other papers we cite and compare, we do not use them in our experiments.

Baselines. We compare the methods we discussed in Section 5 with two baselines:

- **Finetune** greedily updates the model with the incoming mini-batch without considering the previous task performance. The model suffers from catastrophic forgetting and is regarded as the lower-bound.

- **Offline** trains a model using all the samples in a dataset in an offline manner. The baseline is trained for multiple epochs within each of which mini-batches are sampled i.i.d from differently shuffled dataset. We train the model for 70 epochs with the mini-batch size of 128.

Other Details. We evaluate the performance with 5 metrics we described in Section 3.2: Average Accuracy, Average Forgetting, Forward Transfer, Backward Transfer and Run Time. We

Method	Split CIFAR-100			Split Mini-ImageNet			CORe50-NC		
Finetune	3.7 ± 0.3			3.4 ± 0.2			7.7 ± 1.0		
Offline	49.7 ± 2.6			51.9 ± 0.5			51.7 ± 1.8		
EWC++	3.7 ± 0.4			3.5 ± 0.4			8.3 ± 0.3		
LwF	7.2 ± 0.4			7.6 ± 0.7			7.1 ± 1.9		
Buffer Size	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k
ER	7.6 ± 0.5	17.0 ± 1.9	18.4 ± 1.4	6.4 ± 0.9	14.5 ± 2.1	15.9 ± 2.0	23.5 ± 2.4	27.5 ± 3.5	28.2 ± 3.3
MIR	7.6 ± 0.5	18.2 ± 0.8	19.3 ± 0.7	6.4 ± 0.9	16.5 ± 2.1	21.0 ± 1.1	**27.0 ± 1.6**	**32.9 ± 1.7**	**34.5 ± 1.5**
GSS	7.7 ± 0.5	11.3 ± 0.9	13.4 ± 0.6	5.9 ± 0.7	11.2 ± 0.9	13.5 ± 0.8	19.6 ± 3.0	22.2 ± 4.4	21.1 ± 3.5
iCaRL	**16.7 ± 0.8**	19.2 ± 1.1	18.8 ± 0.9	**14.7 ± 0.4**	17.5 ± 0.6	17.4 ± 1.5	22.1 ± 1.4	25.1 ± 1.6	22.9 ± 3.1
A-GEM	3.7 ± 0.4	3.6 ± 0.2	3.8 ± 0.2	3.4 ± 0.2	3.7 ± 0.3	3.3 ± 0.3	8.7 ± 0.6	9.0 ± 0.5	8.9 ± 0.6
CN-DPM	14.0 ± 1.7	-	-	9.4 ± 1.2	-	-	7.6 ± 0.4	-	-
GDumb	10.4 ± 1.1	**22.1 ± 0.9**	**28.8 ± 0.9**	8.8 ± 0.4	**21.1 ± 1.7**	**31.0 ± 1.4**	15.1 ± 1.2	28.1 ± 1.4	32.6 ± 1.7

Table 7. Average accuracy (end of training) for the OCI setting of Split CIFAR-100, Split Mini-ImageNet and CORe50-NC. Replay-based methods and a strong baseline GDumb show competitive performance across three datasets.

use the hyperparameter tuning protocol described in Section 7 and give each method similar tuning budget. The details of the implementation including hyperparameter selection can be found in Appendix B.2.

8.2. Performance comparison in Online Class Incremental (OCI) setting

Regularization-based Methods. As shown in Table 7, EWC++ has almost the same performance as Finetune in the OCI setting. We find that the gradient explosion of the regularization terms is the root cause. Specifically, λ in EWC++ controls the regularization strength, and we need a larger λ to avoid forgetting. However, when λ increases to a certain value, the gradient explosion occurs. If we take θ_j in Eq. (7) as an example, the regularization term for θ_j has the gradient $\lambda F_j(\theta_j - \theta^*)$. Some model weights change significantly when it receives data with new classes, and therefore the gradients for those weights are prone to explode with a large λ. The Huber regularization proposed lately could be a possible remedy [112]. Surprisingly, we also observe that LwF, a method relying on KD, has similar performance as replay-based methods with a small memory buffer(1k) such as ER, MIR and GSS in Split CIFAR-100 and even outperforms them in Split Mini-ImageNet. In the larger and more realistic CORe50-NC, however, both EWC++ and LwF fail. This also confirms the results of three recent studies, where [113] shows the shortcomings of regularization-based approaches in the class incremental setting, [114] theoretically explains why regularization-based methods underperform memory-based methods and [115] empirically demonstrates that KD is more useful in small-scale datasets.

Memory-based Methods. Firstly, A-GEM does not work in this setting as it has almost the same performance as Finetune, implying that the indirect use of the memory samples is less efficient than the direct relay in the OCI setting. Secondly, given a small memory buffer in Split CIFAR-100 and Mini-ImageNet, iCaRL—proposed in 2017—shows the best performance. On the other hand, other replay-based methods such as ER, MIR and GSS do not work well because simply replaying with a small memory buffer yields severe class imbalance. When

equipped with a larger memory buffer (5k and 10k), GDumb— a simple baseline that trains with the memory buffer only— outperforms other methods by a large margin. Additionally, as shown in Fig. 1a and Fig. 1b, GDumb dominates the average accuracy not only at the end of training but also at any other evaluation points along the data stream. This raises concerns about the progress in the OCI setting in the literature. Next, in the larger CORe50-NC dataset, GDumb is less effective since it only relies on the memory and the memory is smaller in a larger dataset in proportion. MIR is a robust and strong method as it exhibits remarkable performance across different memory sizes. Also, even though GSS is claimed to be an enhanced version of ER, ER consistently surpasses GSS across different memory sizes and datasets, which is also confirmed by other studies [18, 60].

Parameter-isolation-based methods. As one of the first dynamic-architecture methods without using a task-ID, CN-DPM shows competitive results in Split CIFAR-100 and Mini-ImageNet but fails in CORe5-NC. The key reason is that CN-DPM is very sensitive to hyperparameters, and when applying CN-DPM in a new dataset, a good performance cannot be guaranteed given a limited tuning budget.

Other Metrics. We show the performance of all five metrics in Fig. 2. Generally speaking, we find that a high average accuracy comes with low forgetting, but methods using KD such as iCaRL and LwF, and dynamic-architecture methods such as CN-DPM have lower forgetting. The reason for the low forgetting of these methods is intransigence, the model's inability to learn new knowledge [8]. For iCaRL and LwF, KD imposes a strong regularization, which may lead to a lower accuracy on new tasks. For CN-DPM, the inaccurate expert selector is the cause of the intransigence [18]. Furthermore, most methods have similar running time except for CN-DPM, GDumb and GSS. CN-DPM requires a significantly longer training time as it needs to train multiple experts, and each expert contains a generative model (VAE [94]) and a classifier(10-layer ResNet [110]). GDumb has the second longest running time as it requires training the model from scratch with the memory
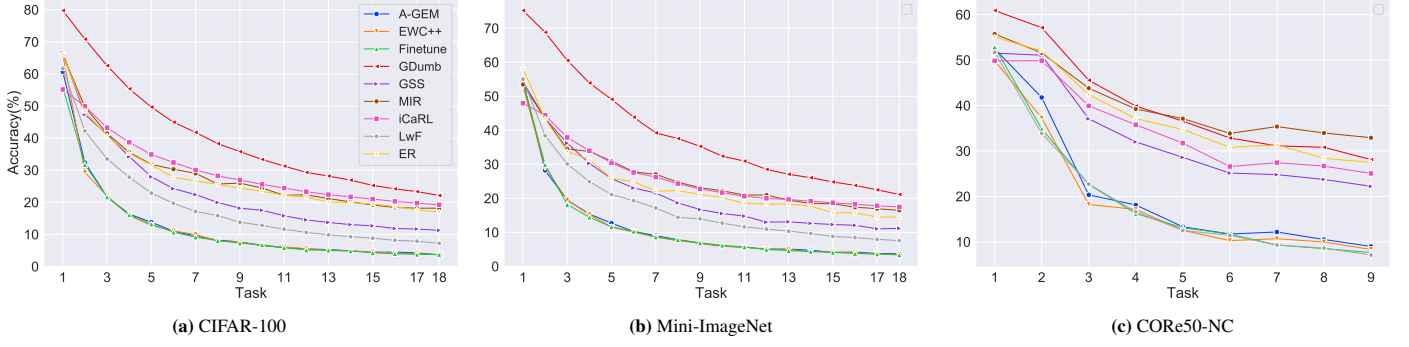
11

(a) CIFAR-100   **(b)** Mini-ImageNet   **(c)** CORe50-NC

Fig. 1. The average accuracy measured by the end of each task for the OCI setting with a 5k memory buffer. More detailed results for different memory buffer sizes are shown in Appendix C.1.
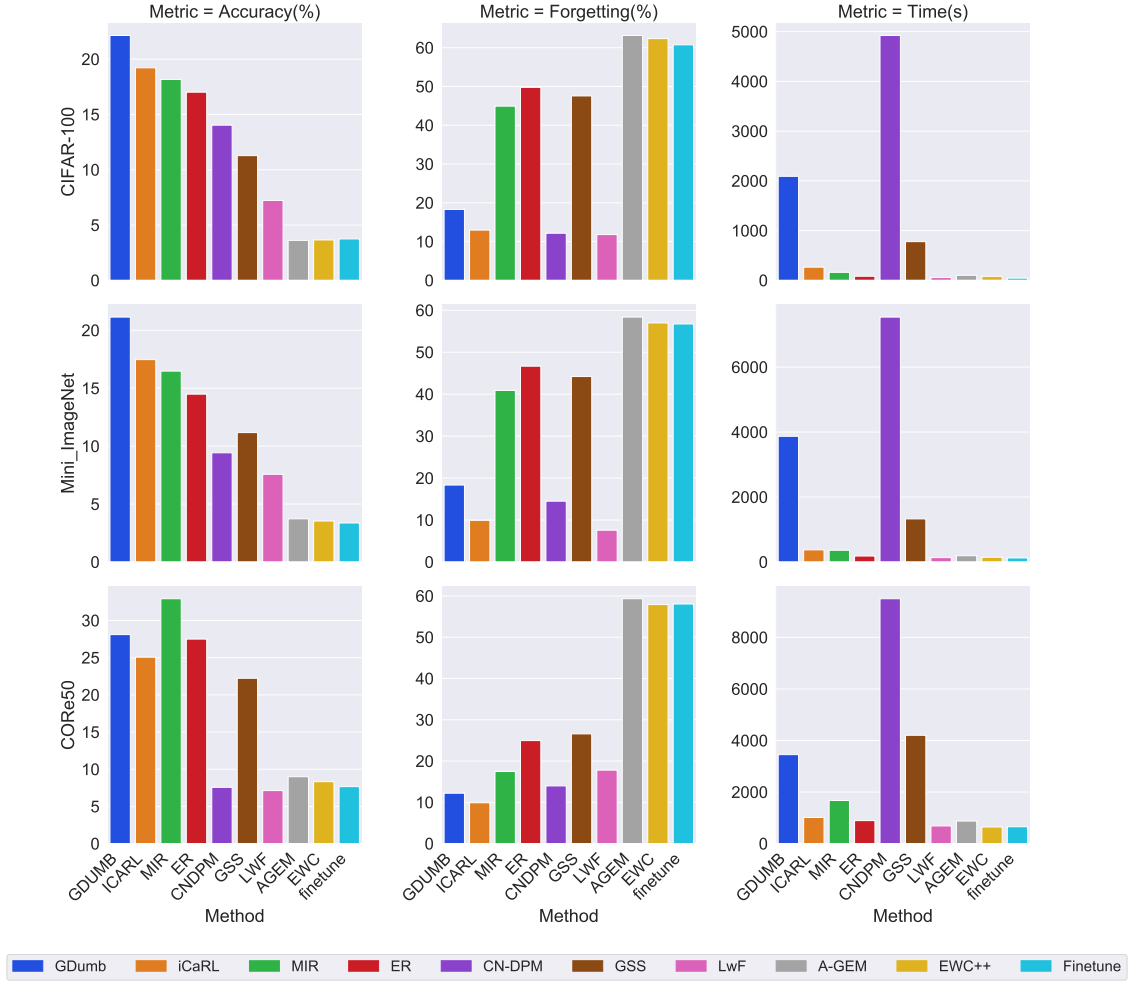

Fig. 2. Average accuracy, forgetting and running time for the OCI setting with a 5k memory buffer. Each column represents a metric and each row represents a dataset. Note that in the OCI setting, none of the methods show any forward and backward transfer and hence they are not shown in this figure.

12

Method	e(n, o)	e(n, n)	e(o, o)	e(o, n)	er(n, o)	er(o, n)
A-GEM	0	177	0	9500	0%	100%
ER	37	148	2269	5852	20%	72%
MIR	54	113	2770	5330	32%	66%

Table 8. Error analysis of CIFAR-100 by the end of training with M=5k. e(n, o) & e(n, n) represent the number of test samples from new classes that are misclassified as old classes and new classes, respectively. Same notation rule is applied to e(o, o) & e(o, n).
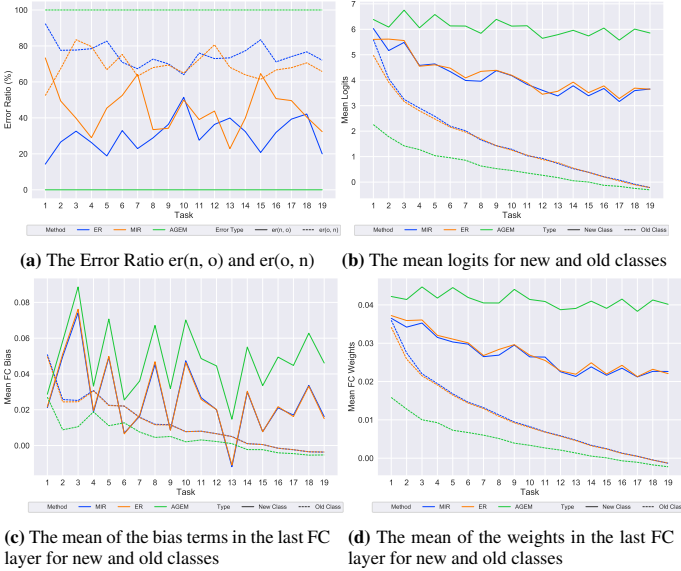

(a) The Error Ratio er(n, o) and er(o, n)

(b) The mean logits for new and old classes

(c) The mean of the bias terms in the last FC layer for new and old classes

(d) The mean of the weights in the last FC layer for new and old classes

Fig. 3. Error analysis for three memory-based methods(A-GEM, ER and MIR) with 5k memory buffer on Split CIFAR-100. The Error Ratio er(n,o) in (a) is the ratio of new class test samples misclassified as old classes to the total number of new class test samples. Same notation rule is applied to er(o, n).

at every evaluation point. Lastly, we notice none of the methods show any forward and backward transfer, which is expected since a model tends to classify all the test samples as the current task labels due to the strong bias in the last FC layer.

8.3. Error analysis of memory-based methods in the OCI setting

We perform quantitative error analysis for three memory-based methods with 5k memory buffer (A-GEM, ER and MIR) on Split CIFAR-100. We define $e(n, o)$ and $e(n, n)$ as the number of test samples from new classes that are misclassified as old classes and new classes, respectively. The same notation rule is applied to $e(o, o)$ and $e(o, n)$. Also, $er(n, o)$ denotes the ratio of new class test samples misclassified as old classes to the total number of new class test samples. $er(o, n)$ is similarly defined.

As shown in Table 8, all methods have strong bias towards new classes by the end of the training: A-GEM classifies all old class samples as new classes; ER and MIR misclassify 72% and 66% old class samples as new classes, respectively. Moreover, as we can see in Fig. 3a, $er(o, n)$ is higher than $er(n, o)$ most of the times along the training process for all three methods. These phenomena are not specific to the OCI setting, and [95, 69] also

found similar results in the offline class incremental learning. Additionally, we easily find that ER and MIR are always better than A-GEM in terms of $er(o, n)$. This is because ER and MIR use the memory samples more directly, namely replaying them with the new class samples. The indirect use of memory samples in A-GEM is less effective in the class incremental setting.

To empirically verify the claim discussed in Section 6.1, we also analyze the mean of logits for new and old classes. As shown in Fig. 3b, the mean of logits for new classes is always much higher than that for old classes, which explains the high $er(o, n)$ for all three methods. As we can see from Eq. (12), both feature extractor $\phi(\mathbf{x})$ and FC layer $\mathbf{W}$ may potentially contribute to the logit bias. However, previous works have found that even a small memory buffer (implying high class imbalance) can greatly alleviate catastrophic forgetting in the multi-head setting where the model can utilize the task-id to select the corresponding FC layer for each task [35, 10]. This suggests that the feature extractor is not heavily affected by the class imbalance, and therefore we hypothesize that the FC layer is biased.

To validate the hypothesis, we plot the means of bias terms and weights in the FC layer for new and old classes in Fig. 3c and 3d. As we can see, the means of weights for the new classes are much higher than those for the old classes, and the means of bias terms for the new classes are also higher than those for the old classes most of the times. Since the biased weights and bias terms in $\mathbf{W}$ have direct impacts on the output logits, the model has a higher chance of predicting a sample as new classes. [95, 43, 69] have also verified the same hypothesis but with different validation methods.

8.4. Effectiveness of tricks in the OCI setting

We evaluate Label trick (LB), Knowledge Distillation and Classification (KDC), Multiple Iterations (MI), Separated Softmax(SS), Nearest Class Mean (NCM) classifier, Review trick(RV) on three memory-based methods: A-GEM, ER and MIR. The results are shown in Table 9 and Fig. 4.

Firstly, although all tricks enhance the basic A-GEM, only RV can bring A-GEM closer to ER and MIR, which reiterates that the direct replay of the memory samples is more effective than the gradient projection approach in A-GEM.

For replay-based ER and MIR, LB and NCM are the most effective when M=1k and can improve the accuracy by around 100% (7.6% → 14.5% on average). KDC, KDC* and SS have similar performance improvement effect and can boost the accuracy by around 64% (7.6% → 12.4% on average). With a larger memory size, NCM remains very effective, and RV becomes much more helpful. When M=10k, RV boosts ER's performance by 74% (18.4% → 32.0%) and improve MIR's performance by 82% (from 19.3% → 35.2%). Also, KDC fails with a larger memory due to over-regularization of the KD term, and the modified KDC* has much better performance. Compared with other tricks, MI and RV are more sensitive to the memory size since these tricks highly depend on the memory. Note that when equipped with NCM or RV, both ER and MIR can outperform the best performance achieved by the compared methods (Table 7) when M=5k or 10k.
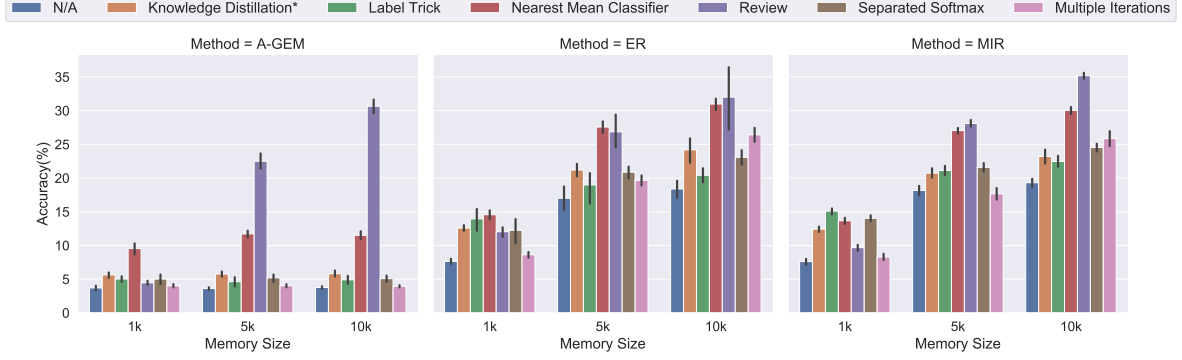
Fig. 4. Comparison of various tricks for the OCI setting on Split CIFAR-100. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k. N/A denotes the performance of the base methods (A-GEM, ER, MIR) without any trick.

		A-GEM			ER			MIR	
Finetune					3.7 ± 0.3				
Offline					49.7 ± 2.6				
Method		A-GEM			ER			MIR	
Buffer Size	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k
N/A	3.7 ± 0.4	3.6 ± 0.2	3.8 ± 0.2	7.6 ± 0.5	17.0 ± 1.9	18.4 ± 1.4	7.6 ± 0.5	18.2 ± 0.8	19.3 ± 0.7
LB	5.0 ± 0.5	4.6 ± 0.8	4.9 ± 0.7	14.0 ± 2.0	19.0 ± 2.6	20.4 ± 1.2	**15.1 ± 0.6**	21.1 ± 0.8	22.5 ± 0.9
KDC	8.3 ± 0.7	8.8 ± 0.7	7.7 ± 1.1	11.7 ± 0.8	10.9 ± 1.9	11.9 ± 2.2	12.0 ± 0.5	12.3 ± 0.7	11.8 ± 0.6
KDC*	5.6 ± 0.5	5.8 ± 0.5	5.8 ± 0.5	12.6 ± 0.5	21.2 ± 1.1	24.2 ± 1.9	12.4 ± 0.5	20.7 ± 0.8	23.2 ± 1.2
MI	4.0 ± 0.3	4.0 ± 0.3	4.0 ± 0.2	8.6 ± 0.5	19.7 ± 0.9	26.4 ± 1.2	8.5 ± 0.4	17.7 ± 1.0	25.9 ± 1.2
SS	5.0 ± 0.8	5.2 ± 0.6	5.1 ± 0.5	12.3 ± 2.1	20.9 ± 1.0	23.1 ± 1.2	14.0 ± 0.5	21.6 ± 0.7	24.5 ± 0.7
NCM	**9.5 ± 0.9**	11.7 ± 0.6	11.5 ± 0.7	**14.6 ± 0.7**	**27.6 ± 1.0**	31.0 ± 1.0	13.7 ± 0.5	27.0 ± 0.5	30.0 ± 0.6
RV	4.5 ± 0.4	**22.5 ± 1.3**	**30.7 ± 1.2**	12.0 ± 0.8	26.9 ± 2.8	**32.0 ± 5.3**	9.7 ± 0.5	**28.1 ± 0.6**	**35.2 ± 0.5**
Best OCI	16.7 ± 0.8	22.1 ± 0.9	28.8 ± 0.9	16.7 ± 0.8	22.1 ± 0.9	28.8 ± 0.9	16.7 ± 0.8	22.1 ± 0.9	28.8 ± 0.9

Table 9. Performance of compared tricks for the OCI setting on Split CIFAR-100. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k. Best OCI refers to the best performance achieved by the compared methods in Table 7.

Trick	Running Time(s)		
	M=1k	M=5k	M=10k
N/A	83	82	84
LB	87	88	89
KDC	105	106	106
KDC*	105	105	107
MI	328	324	325
SS	89	90	90
NCM	126	282	450
RV	98	159	230

Table 10. Running time of different tricks applying to ER with different memory sizes on Split CIFAR-100.

As shown in Table 10, the running times of LB, KDC, KDC*, MI and SS do not depend on the memory size and have a limited increase compared to the baseline. Since NCM needs to calculate the means of all the classes in the memory and RV requires additional training of the whole memory before evaluation, their running times increase as the memory size grows.

To sum up, all of the tricks are beneficial to the base methods (A-GEM, ER, MIR); NCM is a useful and robust trick across all memory sizes, while LB and RV are more advantageous in smaller and larger memory, respectively. The running times of NCM and RV go up with the increase in the memory size, and other tricks only add a fixed overhead to the running time.

We also get similar results in Split Mini-ImageNet, as shown in Appendix C.

8.5. Performance comparison in Online Domain Incremental (ODI) setting

Since most of the surveyed methods are only evaluated in the class incremental setting in the original papers, we evaluate them in the ODI setting to investigate their robustness and ability to generalize to other CL settings. We assess the methods with CORe50-NI—a dataset designed for ODI—and the proposed NS-MiniImageNet dataset consisting of three nonstationary types: noise, occlusion, and blur (see Table 6). The average accuracy at the end of training is summarized in Table 11.

Generally speaking, all replay-based methods (ER, MIR, GSS) show comparable performance across three memory sizes and outperform all other methods. GDumb, the strong baseline that dominates the OCI setting in most cases, is no longer as competitive as the replay-based methods and fails completely in CORe50-NI. One of the reasons is that class imbalance, the key cause of forgetting in the OCI setting, does not exist in ODI since the class labels are the same for all tasks. Moreover, in the ODI setting, samples in the data stream change gradually and smoothly with different nonstationary strengths (NS-MiniIMageNet) or nonstationary types (CORe50-NI). Learn-

Method	Mini-ImageNet-Noise			Mini-ImageNet-Occlusion			Mini-ImageNet-Blur			CORe50-NI		
Finetune	11.1 ± 1.0			13.8 ± 1.6			2.4 ± 0.2			14.0 ± 2.8		
Offline	37.3 ± 0.8			38.6 ± 4.7			11.9 ± 1.0			51.7 ± 1.8		
EWC	12.5 ± 0.8			14.8 ± 1.1			2.6 ± 0.2			11.6 ± 1.5		
LwF	9.2 ± 0.9			12.8 ± 0.8			3.4 ± 0.4			11.1 ± 1.1		
Buffer Size	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k
ER	**19.4 ± 1.3**	21.6 ± 1.1	24.3 ± 1.2	**19.2 ± 1.5**	**23.4 ± 1.4**	23.7 ± 1.1	5.3 ± 0.6	**8.6 ± 0.8**	9.4 ± 0.7	24.1 ± 4.2	28.3 ± 3.5	30.0 ± 2.8
MIR	18.1 ± 1.1	**22.5 ± 1.4**	**24.4 ± 0.9**	17.6 ± 0.7	22.0 ± 1.1	**23.8 ± 1.2**	**5.5 ± 0.5**	8.1 ± 0.6	9.6 ± 1.0	**26.5 ± 1.0**	**34.0 ± 1.0**	**33.3 ± 1.7**
GSS	18.9 ± 0.8	21.4 ± 0.9	23.2 ± 1.1	17.7 ± 0.8	21.0 ± 2.2	23.2 ± 1.4	5.2 ± 0.5	7.6 ± 0.6	8.0 ± 0.6	25.5 ± 2.1	27.2 ± 2.0	25.3 ± 2.1
A-GEM	14.0 ± 1.3	14.6 ± 0.7	14.2 ± 1.4	16.4 ± 0.7	13.9 ± 2.6	14.4 ± 2.0	4.4 ± 0.4	4.4 ± 0.4	4.3 ± 0.5	12.4 ± 1.1	13.8 ± 1.2	15.0 ± 2.2
CN-DPM	4.6 ± 0.5	-	-	3.9 ± 0.8	-	-	2.2 ± 0.2	-	-	9.6 ± 3.9	-	-
GDumb	5.4 ± 1.0	12.5 ± 0.7	15.2 ± 0.5	5.4 ± 0.4	14.2 ± 0.6	20.2 ± 0.4	3.3 ± 0.2	7.5 ± 0.2	**10.0 ± 0.2**	9.6 ± 1.5	11.2 ± 2.0	11.5 ± 1.7

Table 11. The Average Accuracy (end of training) for the ODI setting of Mini-ImageNet with three nonstationary types (Noise, Occlusion, Blur) and CORe50-NI.
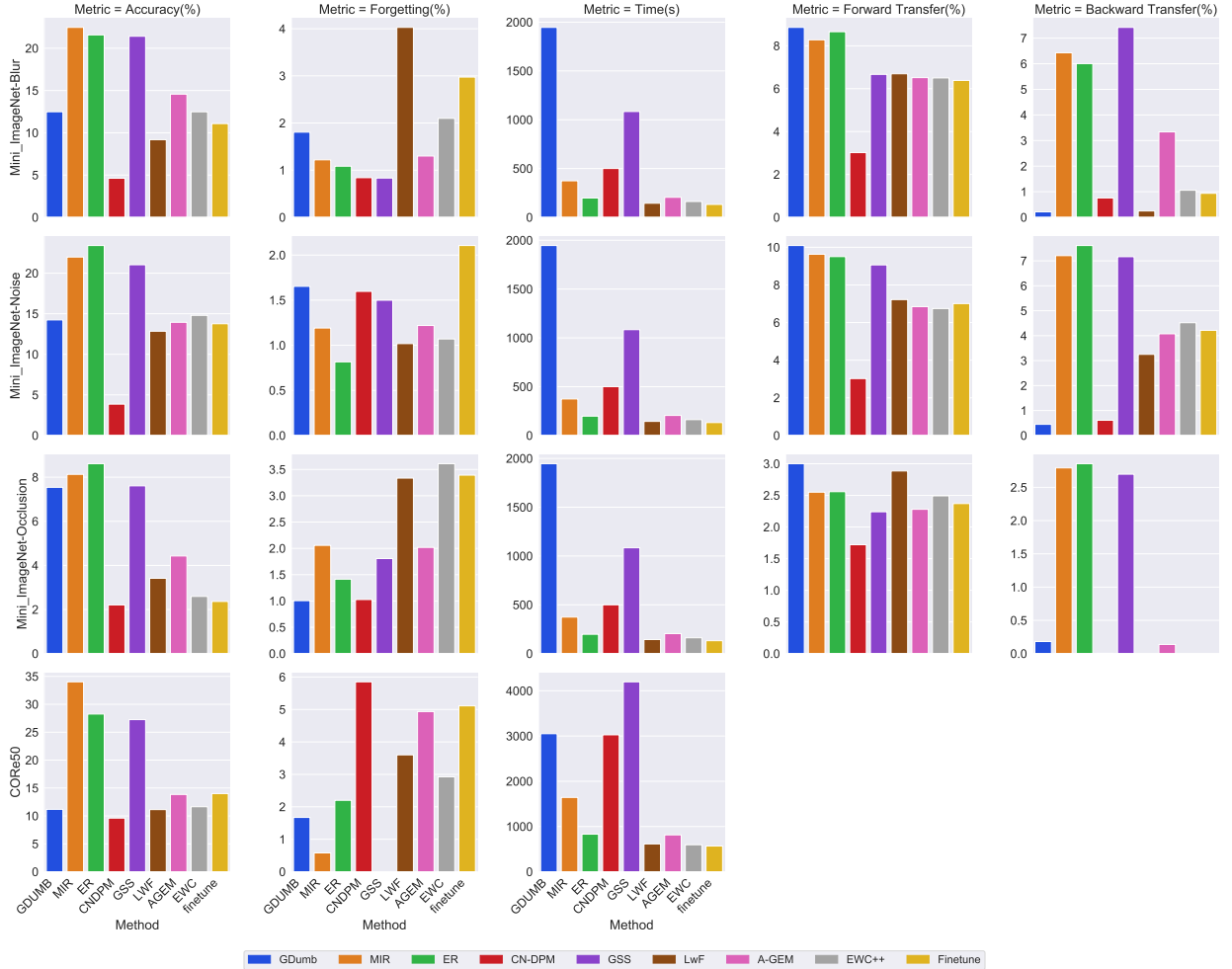

Fig. 5. Average Accuracy, Forgetting, Running Time, Forward Transfer and Backward Transfer for the ODI setting with a 5k memory buffer. Each column represents a metric and each row represents a dataset. Forward and backward transfer are not applicable in CORe50 since it uses one test set for all tasks.

ing new samples sequentially with the replay samples(replay-based) may be more effective for the model to adapt to the gradually changing nonstationarity than learning only the samples in the buffer (GDumb). Additionally, the greedy memory update strategy (see Algorithm A3 in Appendix) in GDumb is not suitable for the ODI setting as the buffer will comprise mostly of samples in the latest tasks due to the greedy update, and GDumb will have very limited access to samples in the earlier tasks. Using reservoir sampling as the update strategy may alleviate this shortcoming since reservoir sampling ensures every data point has the same probability to be stored in the memory.

CN-DPM has terrible performance in this setting because it is very sensitive to hyperparameters, and the method cannot find the hyperparameter set that works in this setting within the same tuning budget as other methods.

Regarding methods without a memory buffer, the KD-based LwF underperforms EWC and Finetune, implying KD may not be useful in the ODI setting.

Another interesting observation is that all methods, including Offline, show unsatisfactory results in the blur scenario. The pivotal cause may be due to the backbone model we use (ResNet18 [110]) since a recent study points out that Gaussian blur can easily degrade the performance of ResNet [116].

In terms of other metrics, as shown in Fig. 5, we find that all methods show positive forward transfer, and replay-based methods show much better backward transfer than others. The first reason is that tasks in the ODI setting share more cross-task resemblances than the OCI setting; secondly, the bias towards the current task due to class imbalance does not happen since new tasks contain the same class labels as old tasks. Thus, the model is able to perform zero-shot learning (forward transfer) and improve the preceding tasks (backward transfer).

In summary, replay-based methods exhibit more robust and surpassing performance in the ODI setting. Considering the running time and the performance in the larger scale dataset, MIR stands out as a versatile and competitive method in this setting.

8.6. Overall comments for methods and tricks

We summarize the key findings and give comments on each method and trick based on our findings in Table 12 and Table 13.

9. Trendy Directions in Online CL

In this section, we discuss some emerging directions in online CL that have attracted interest and are expected to gain more attention in the future.

Raw-Data-Free Methods. In some applications, storing raw images is not feasible due to privacy and security concerns, and this calls for CL methods that maintain reasonable performance without storing raw data. Regularization [31, 84] is one of the directions but [113] shows that this approach has theoretical limitations in the class incremental setting and cannot

Method	Comments
Regularization-based	
EWC++	• Ineffective in both OCI and ODI settings • Suffers from gradient explosion
LwF	• Effective on small scale datasets in OCI (achieves similar performance as replay-based methods with small memory) • Ineffective in ODI setting
Memory-based	
ER	• Efficient training time over other memory-based methods • Better than GSS in most cases but worse than MIR, especially with a large memory buffer
MIR	• A versatile and competitive method in both OCI and ODI settings • Works better on a large scale dataset and a large memory buffer
GSS	• Inefficient training time • Worse than other memory-based methods in most cases
iCaRL	• Best performance (with large margins) with a small memory buffer on small scale datasets
A-GEM	• Ineffective in both OCI and ODI settings
GDumb	• Best performance with a large memory buffer on small scale datasets in OCI setting • Ineffective in ODI mostly due to its memory update strategy • Inefficient training time due to training from scratch at every inference point
Parameter-isolation-based	
CN-DPM	• Effective when memory size is small • Sensitive to hyperparameters and when testing on a new dataset, it may not find a working hyperparameter set given the same tuning budget as others • Longest training time among compared methods

Table 12. Overall comments for compared methods

be used alone to reach decent performance. We also have empirically confirmed their claims in this work. Generative replay [117, 48] is another direction but it is not viable for more complex datasets as the current deep generative models still cannot generate satisfactory images for such datasets [16, 51].

Feature replay is a promising direction where latent features of the old samples at a given layer (feature extraction layer) are relayed instead of raw data [66, 118, 119, 120, 121]. Since the model changes along the training process, to keep the latent features valid, [120] proposes to slow-down—in the limit case, freeze—the learning of all the layers before the feature extraction layer, while [66] proposes a feature adaptation method to map previous features to their correct values as the model is updated. Another way is to generate latent features with a deep generative model [119].

There are other lately proposed approaches which do not require storing the raw data [122, 79, 12, 123, 74, 124, 34]. For example, SDC [79] leverages embedding networks [123] and the nearest class mean classifier [89]. The approach proposes a method to estimate the drift of features during learning the current task and compensate for the drift in the absence of previous samples. DMC [74] trains a separate model for new tasks and combines the new and old models using publicly available unlabeled data via a double distillation training objective. DSLDA [124] freezes the feature extractor and uses deep Streaming Linear Discriminant Analysis [125] to train the output layer incrementally. InstAParam [12] leverages the "instance awareness" where each data instance is classified by a path in the network searched by the controller from a meta-graph (updated online). It encourages or restricts the gradient updates for the current path based on the similarity between the current instance and previous instances. EBM-CL[126] interprets continual learning for classification as learning an Energy-Based Model [127] across classes and proposes a contrastive divergence based training objective, which can naturally handle dynamically growing numbers of classes.

With the increasing data privacy and security concerns, the raw-data-free methods are expected to attract more research endeavour in the coming years.

Meta Learning. Meta-learning is an emerging learning paradigm where a neural network evolves from multiple related learning episodes and generalizes the learned knowledge to unseen tasks [128]. Since meta-learning builds up a potential framework to advance CL, a lot of meta-learning based CL methods have been proposed recently, and most of them support the online setting. MER [61] combines experience replay with optimization based meta-learning to maximize transfer and minimize interference based on future gradients. OML [129] is a meta-objective that uses interference as a training signal to learn a representation that accelerates future learning and avoid catastrophic interference. More recently, iTAML [82] proposes to learn a task-agnostic model that automatically predicts the task and quickly adapts to the predicted task with meta-update. La-MAML [65] proposes an efficient gradient-based meta-learning that incorporates per-parameter learning rates for online CL. MERLIN [67] proposes an online

CL method based on consolidation in a meta-space, namely, the latent space that generates model weights for solving downstream tasks. In [130], authors propose Continual-MAML, an online extension of MAML [131], that can cope the new CL scenario they propose. We believe meta-learning based online CL methods will continue to be popular with recent advances in meta-learning.

CL in Other Areas. Although image classification and reinforcement learning are the main focuses for most CL works, CL has drawn more and more attention in other areas. Object detection has been another emerging topic in CL, and multiple works have been proposed lately to tackle this problem. Most methods leverage KD [41] to alleviate CF, and the main differences between the methods are the base object detector and distillation parts in the network [132, 133, 134, 135]. More recently, a meta-learning based approach is proposed to reshape model gradients for better information share across incremental tasks [136]. A replay-based method is introduced to address streaming object detection by replaying compressed representation in a fixed memory buffer [137].

Beyond computer vision, CL with sequential data and recurrent neural network (RNN) has gained attention over the past few years. Recent works have confirmed that RNNs, including LSTMs, are also immensely affected by CF [138, 139, 140]. In [138], the authors unify GEM [11] and Net2Net [141] to tackle forgetting in RNN. More recently, [142] shows that weight-importance based CL in RNNs are limited and that the hypernetwork-based approaches are more effective in alleviating forgetting. Meanwhile, [143] proposes a learning rule to preserve network dynamics within subspaces for previous tasks and encourage interfering dynamics to explore orthogonal subspaces when learning new tasks. Moreover, multiple works are proposed to address general CL language learning [144, 145] and specific language tasks, such as dialogue systems [146, 147, 148], image captioning [149], sentiment classification [150] and sentence representation learning [151].

Recommender systems have also started to adopt CL [152, 153, 154, 155]. ADER [154] is proposed to handle CF in session-based recommendation using the adaptive distillation loss and replay with heading [90] technique. GraphSAIL [155] is introduced for Graph Neural Networks based recommender systems to preserve a user's long-term preference during incremental model updates using local structure distillation, global structure distillation and self-embedding distillation.

Several works also address the deployment of CL in practice. [156] introduces on-the-job learning, which requires a deployed model to discover new tasks, collect training data continuously and incrementally learn new tasks without interrupting the application. The author also uses chat-bots and self-driving cars as the examples to highlight the necessity of on-the-job learning. [157] presents a reference architecture for self-maintaining intelligent systems that can adapt to shifting data distributions, cope with outliers, retrain when necessary, and learn new tasks incrementally. [158] discusses the clinical application of CL from three perspectives: diagnosis, prediction and treatment decisions. [159] addresses a practical scenario where a high-

Trick	Comments
LB	• Effective when memory buffer is small • Fixed and limited training time overhead
KDC	• Fails because of over-regularization of knowledge distillation loss
KDC*	• Provides moderate improvement with fixed and acceptable training time overhead
MI	• Better improvement with a larger memory buffer • Training time increases with more iterations
SS	• Similar improvement as KDC* but with less training time overhead
NCM	• Provides very strong improvement across different memory sizes. • Baselines equipped with it outperform state-of-the-art methods when the memory buffer is large • Inference time increases with the growth of the memory size
RV	• Presents very competitive improvement, especially with a larger memory buffer • Baselines equipped with it outperform state-of-the-art methods • Training time increases with the growth of the memory size but it is more efficient than NCM

Table 13. Overall comments for compared tricks

capacity server interacts with a large group of resource-limited edge devices and proposes a Dual User-Adaptation framework which disentangles user-adaptation into model personalization on the server and local data regularization on the user device.

10. Conclusion

To better understand the relative advantages of recently proposed online CL approaches and the settings where they work best, we performed extensive experiments with nine methods and seven tricks in the online class incremental (OCI) and online domain incremental (ODI) settings.

Regarding the performance in the OCI setting (see Table 7, Fig. 1 and 2), we conclude:

- For memory-free methods, LwF is effective in CIFAR100 and Mini-ImageNet, showing similar performance as replay-based methods with a small memory buffer. However, all memory-free methods fail in the larger CORe50-NC.

- When the memory buffer is small, iCaRL shows the best

performance in CIFAR100 (a small margin) and Mini-ImageNet (a large margin), followed by CN-DPM.

- With a larger memory buffer, GDumb—a simple baseline—outperforms methods designed specifically for the CL problem in CIFAR100 and Mini-ImageNet at the expense of much longer training times.

- In the larger and more realistic CORe50-NC dataset, MIR consistently surpasses all the other methods across different memory sizes.

- We experimentally and theoretically confirm that a key cause of CF is the bias towards new classes in the last fully connected layer due to the imbalance between previous data and new data [43, 69, 95].

- None of the methods show any positive forward and backward transfer due to the bias mentioned above.

The conclusions from our experiments for the OCI tricks (see Table 9, Fig. 4) are as follows:

- When the memory size is small, LB and NCM are the most effective, showing around 64% relative improvement.

- With a larger memory buffer, NCM remains effective, and RV becomes more helpful, showing around 80% relative improvement.

- When equipped with NCM or RV, both ER and MIR can outperform the best performance of the compared methods without tricks.

- The running times of NCM and RV increase with the growth in memory size, but other tricks only add a fixed overhead to the running time.

For the ODI setting (see Table 11, Fig. 5), we conclude:

- Generally speaking, all replay-based methods (ER, MIR, GSS) show comparable performance across three memory sizes and outperform all other methods.

- GDumb, the strong baseline that dominates the OCI setting in most cases, is no longer effective, possibly due to its memory update strategy.

- Other OCI methods cannot generalize to the ODI setting.

Detailed comments for compared methods and tricks can be found in Table 12 and Table 13.

In conclusion, by leveraging the best methods and tricks identified in this comparative survey, online CL (with a very small mini-batch) is now approaching offline performance, bringing CL much closer to its ultimate goal of matching offline training that opens up CL for effective deployment on edge and other RAM-limited devices.

11. Acknowledgement

References

[1] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of learning and motivation, Vol. 24, Elsevier, 1989, pp. 109–165.

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

[3] G. A. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Computer vision, graphics, and image processing 37 (1) (1987) 54–115.

[4] M. Mermillod, A. Bugaiska, P. Bonin, The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, Frontiers in psychology 4 (2013) 504.

[5] Z. Chen, B. Liu, Lifelong machine learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 12 (3) (2018) 1–207.

[6] J. Yoon, E. Yang, J. Lee, S. J. Hwang, Lifelong learning with dynamically expandable networks, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=Sk7KsfW0-

[7] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, icarl: Incremental classifier and representation learning, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[8] A. Chaudhry, P. K. Dokania, T. Ajanthan, P. H. Torr, Riemannian walk for incremental learning: Understanding forgetting and intransigence, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 532–547.

[9] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: Defying forgetting in classification tasks, arXiv preprint arXiv:1909.08383 (2019).

[10] A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with a-GEM, in: International Conference on Learning Representations, 2019.

[11] D. Lopez-Paz, M. A. Ranzato, Gradient episodic memory for continual learning, in: Advances in Neural Information Processing Systems 30, 2017, pp. 6467–6476.

[12] H.-J. Chen, A.-C. Cheng, D.-C. Juan, W. Wei, M. Sun, Mitigating forgetting in online continual learning via instance-aware parameterization, Advances in Neural Information Processing Systems 33 (2020) 17466–17477.

[13] Q. Pham, C. Liu, D. Sahoo, et al., Contextual transformation networks for online continual learning, in: International Conference on Learning Representations, 2020.

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

[15] S. Farquhar, Y. Gal, Towards robust evaluations of continual learning (2018). arXiv:1805.09733.

[16] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, L. Page-Caccia, Online continual learning with maximal interfered retrieval, in: Advances in Neural Information Processing Systems 32, 2019, pp. 11849–11860.

[17] R. Aljundi, M. Lin, B. Goujaud, Y. Bengio, Gradient based sample selection for online continual learning, in: Advances in Neural Information Processing Systems 32, 2019, pp. 11816–11825.

[18] S. Lee, J. Ha, D. Zhang, G. Kim, A neural dirichlet process mixture model for task-free continual learning, in: International Conference on Learning Representations, 2020.

[19] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, G. Wayne, Experience replay for continual learning, in: Advances in Neural Information Processing Systems 32, 2019, pp. 350–360.

[20] A. Nagabandi, C. Finn, S. Levine, Deep online learning via meta-learning: Continual adaptation for model-based RL, in: International Conference on Learning Representations, 2019.

[21] C. Kaplanis, M. Shanahan, C. Clopath, Continual reinforcement learning with complex synapses, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 2497–2506.

[22] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, R. Hadsell, Continual unsupervised representation learning, in: Advances in Neural Information Processing Systems, 2019, pp. 7647–7657.

[23] A. Prabhu, P. H. Torr, P. K. Dokania, Gdumb: A simple approach that questions our progress in continual learning, in: European Conference on Computer Vision, Springer, 2020, pp. 524–540.

[24] Z. Mai, H. Kim, J. Jeong, S. Sanner, Batch-level experience replay with review for continual learning, arXiv preprint arXiv:2007.05683 (2020).

[25] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Networks 113 (2019) 54 – 71.

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

[27] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, Z. Kira, Re-evaluating continual learning scenarios: A categorization and case for strong baselines, arXiv preprint arXiv:1810.12488 (2018).

[28] G. M. van de Ven, A. Tolias, Three scenarios for continual learning, ArXiv abs/1904.07734 (2019).

[29] B. Pfülb, A. Gepperth, A comprehensive, application-oriented study of catastrophic forgetting in DNNs, in: International Conference on Learning Representations, 2019.

[30] R. Kemker, A. Abitino, M. McClure, C. Kanan, Measuring catastrophic forgetting in neural networks, in: AAAI, 2018.

[31] Z. Li, D. Hoiem, Learning without forgetting, in: ECCV, Springer, 2016, pp. 614–629.

[32] A. Gepperth, B. Hammer, Incremental learning algorithms and applications, 2016.

[33] T. L. Hayes, R. Kemker, N. D. Cahill, C. Kanan, New metrics and experimental paradigms for continual learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 2031–2034.

[34] E. Fini, S. Lathuiliere, E. Sangineto, M. Nabi, E. Ricci, Online continual learning under extreme memory constraints, in: European Conference on Computer Vision, Springer, 2020, pp. 720–735.

[35] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. S. Torr, M. Ranzato, On tiny episodic memories in continual learning (2019). arXiv:1902.10486.

[36] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, B.-T. Zhang, Overcoming catastrophic forgetting by incremental moment matching, in: Advances in neural information processing systems, 2017, pp. 4652–4662.

[37] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, T. Tuytelaars, Memory aware synapses: Learning what (not) to forget, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 139–154.

[38] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, Proceedings of machine learning research 70 (2017) 3987.

[39] H. Ritter, A. Botev, D. Barber, Online structured laplace approximations for overcoming catastrophic forgetting, in: Advances in Neural Information Processing Systems, 2018, pp. 3738–3748.

[40] X. He, H. Jaeger, Overcoming catastrophic interference using conceptor-aided backpropagation, in: International Conference on Learning Representations, 2018.

[41] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 (2015).

[42] F. M. Castro, M. J. Marin-Jimenez, N. Guil, C. Schmid, K. Alahari, End-to-end incremental learning, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[43] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 374–382.

[44] A. Rannen, R. Aljundi, M. B. Blaschko, T. Tuytelaars, Encoder based lifelong learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1320–1328.

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

[46] X. Tao, X. Chang, X. Hong, X. Wei, Y. Gong, Topology-preserving class-incremental learning, in: European Conference on Computer Vision, Springer, 2020, pp. 254–270.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[48] H. Shin, J. K. Lee, J. Kim, J. Kim, Continual learning with deep gen-

erative replay, in: Advances in Neural Information Processing Systems, 2017, pp. 2990–2999.

[49] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, Y. Fu, Incremental classifier learning with generative adversarial networks, arXiv preprint arXiv:1802.00853 (2018).

[50] G. M. van de Ven, A. S. Tolias, Generative replay with feedback connections as a general strategy for continual learning, arXiv preprint arXiv:1809.10635 (2018).

[51] T. Lesort, H. Caselles-Dupré, M. Garcia-Ortiz, A. Stoian, D. Filliat, Generative models from the perspective of continual learning, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

[52] A. Mallya, S. Lazebnik, Packnet: Adding multiple tasks to a single network by iterative pruning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7765–7773.

[53] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra, Pathnet: Evolution channels gradient descent in super neural networks, arXiv preprint arXiv:1701.08734 (2017).

[54] J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: International Conference on Machine Learning, 2018, pp. 4548–4557.

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

[56] J. Yoon, E. Yang, J. Lee, S. J. Hwang, Lifelong learning with dynamically expandable networks, in: International Conference on Learning Representations, 2018.

[57] R. Aljundi, P. Chakravarty, T. Tuytelaars, Expert gate: Lifelong learning with a network of experts, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3366–3375.

[58] J. Rajasegaran, M. Hayat, S. H. Khan, F. S. Khan, L. Shao, Random path selection for continual learning, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, pp. 12669–12679.

[59] Z. Borsos, M. Mutný, A. Krause, Coresets via bilevel optimization for continual learning and streaming, arXiv preprint arXiv:2006.03875 (2020).

[60] P. Buzzega, M. Boschini, A. Porrello, D. Abati, S. Calderara, Dark experience for general continual learning: a strong, simple baseline, arXiv preprint arXiv:2004.07211 (2020).

[61] M. Reimer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, G. Tesauro, Learning to learn without forgetting by maximizing transfer and minimizing interference, in: ICLR, 2019.

[62] A. Chrysakis, M.-F. Moens, Online continual learning from imbalanced data, Proceedings of Machine Learning Research (2020).

[63] X. Jin, J. Du, X. Ren, Gradient based memory editing for task-free continual learning, arXiv preprint arXiv:2006.15294 (2020).

[64] C. D. Kim, J. Jeong, G. Kim, Imbalanced continual learning with partitioning reservoir sampling, arXiv preprint arXiv:2009.03632 (2020).

[65] G. Gupta, K. Yadav, L. Paull, La-maml: Look-ahead meta learning for continual learning, arXiv preprint arXiv:2007.13904 (2020).

[66] A. Iscen, J. Zhang, S. Lazebnik, C. Schmid, Memory-efficient incremental learning through feature adaptation, arXiv preprint arXiv:2004.00713 (2020).

[67] K. Joseph, V. N. Balasubramanian, Meta-consolidation for continual learning, arXiv preprint arXiv:2010.00352 (2020).

[68] D. Roy, P. Panda, K. Roy, Tree-cnn: a hierarchical deep convolutional neural network for incremental learning, Neural Networks 121 (2020) 148–160.

[69] B. Zhao, X. Xiao, G. Gan, B. Zhang, S.-T. Xia, Maintaining discrimination and fairness in class incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13208–13217.

[70] S. Hou, X. Pan, C. C. Loy, Z. Wang, D. Lin, Learning a unified classifier incrementally via rebalancing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 831–839.

[71] E. Belouadah, A. Popescu, Il2m: Class incremental learning with dual memory, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 583–592.

[72] J. He, R. Mao, Z. Shao, F. Zhu, Incremental learning in online scenario, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13926–13935.

[73] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, R. Chellappa, Learning without memorizing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5138–5146.

[74] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, C.-C. J. Kuo, Class-incremental learning via deep model consolidation, in: The IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 1131–1140.

[75] M. Riemer, T. Klinger, D. Bouneffouf, M. Franceschini, Scalable recollections for continual lifelong learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 1352–1359.

[76] L. Caccia, E. Belilovsky, M. Caccia, J. Pineau, Online learned continual compression with adaptive quantization modules, arXiv (2019) arXiv–1911.

[77] D. Maltoni, V. Lomonaco, Continuous learning in single-incremental-task scenarios, Neural Networks 116 (2019) 56–73.

[78] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, Q. Sun, Mnemonics training: Multi-class incremental learning without forgetting, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12245–12254.

[79] L. Yu, B. Twardowski, X. Liu, L. Herranz, K. Wang, Y. Cheng, S. Jui, J. v. d. Weijer, Semantic drift compensation for class-incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6982–6991.

[80] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, M. Nabi, Learning to remember: A synaptic plasticity driven framework for continual learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11321–11329.

[81] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, Y. Fu, Incremental classifier learning with generative adversarial networks, arXiv preprint arXiv:1802.00853 (2018).

[82] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, M. Shah, itaml: An incremental task-agnostic meta-learning approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13588–13597.

[83] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, B. E. Bejnordi, Conditional channel gated networks for task-aware continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3931–3940.

[84] 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, Proceedings of the National Academy of Sciences of the United States of America 114 13 (2017) 3521–3526.

[85] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, R. Hadsell, Progress & compress: A scalable framework for continual learning, in: ICML, 2018.

[86] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, A. D. Bagdanov, Rotate your networks: Better weight consolidation and less catastrophic forgetting, in: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE, 2018, pp. 2262–2268.

[87] D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, J. Jang, Online class-incremental continual learning with adversarial shapley value, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 9630–9638.

[88] J. S. Vitter, Random sampling with a reservoir, ACM Transactions on Mathematical Software (TOMS) 11 (1) (1985) 37–57.

[89] T. Mensink, J. Verbeek, F. Perronnin, G. Csurka, Distance-based image classification: Generalizing to new classes at near-zero cost, IEEE transactions on pattern analysis and machine intelligence 35 (11) (2013) 2624–2637.

[90] M. Welling, Herding dynamical weights to learn, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 1121–1128.

[91] T. L. Hayes, N. D. Cahill, C. Kanan, Memory efficient experience replay for streaming learning, in: 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 9769–9776.

[92] T. S. Ferguson, Bayesian density estimation by mixtures of normal distributions, in: Recent advances in statistics, Elsevier, 1983, pp. 287–302.

[93] D. Lin, Online learning of nonparametric mixture models via sequential variational approximation, in: Advances in Neural Information Process-

ing Systems, 2013, pp. 395–403.

[94] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[95] H. Ahn, T. Moon, A simple class decision balancing for incremental learning (2020). `arXiv:2003.13947`.

[96] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[97] Q. Dong, S. Gong, X. Zhu, Imbalanced deep learning by minority class incremental rectification, IEEE transactions on pattern analysis and machine intelligence 41 (6) (2018) 1367–1381.

[98] X. Yin, X. Yu, K. Sohn, X. Liu, M. Chandraker, Feature transfer learning for deep face recognition with under-represented data, arXiv preprint arXiv:1803.09014 (2018).

[99] S. Gidaris, N. Komodakis, Dynamic few-shot visual learning without forgetting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4367–4375.

[100] K. Lee, K. Lee, J. Shin, H. Lee, Overcoming catastrophic forgetting with unlabeled data in the wild, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 312–321.

[101] H. Jung, J. Ju, J. Kim, Less-forgetful learning for domain expansion in deep neural networks, in: AAAI-18: Thirty-Second AAAI Conference on Artificial Intelligence, the Association for the Advancement of Artificial Intelligence, 2018.

[102] A. Douillard, M. Cord, C. Ollion, T. Robert, Podnet: Pooled outputs distillation for small-tasks incremental learning, Springer.

[103] C. Zeno, I. Golan, E. Hoffer, D. Soudry, Task agnostic continual learning using online variational bayes (2018). `arXiv:1803.10123`.

[104] K. Javed, F. Shafait, Revisiting distillation and incremental classifier learning, in: Asian Conference on Computer Vision, Springer, 2018, pp. 3–17.

[105] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep., University of Toronto (April 2009).

[106] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in neural information processing systems, 2016, pp. 3630–3638.

[107] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[108] V. Lomonaco, D. Maltoni, Core50: a new dataset and benchmark for continuous object recognition, in: Conference on Robot Learning, 2017, pp. 17–26.

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

[110] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[111] V. Lomonaco, L. Pellegrini, P. Rodriguez, M. Caccia, Q. She, Y. Chen, Q. Jodelet, R. Wang, Z. Mai, D. Vazquez, et al., Cvpr 2020 continual learning in computer vision competition: Approaches, results, current challenges and future directions, arXiv preprint arXiv:2009.09929 (2020).

[112] L. Liu, Z. Kuang, Y. Chen, J. H. Xue, W. Yang, W. Zhang, Incdet: In defense of elastic weight consolidation for incremental object detection, IEEE Transactions on Neural Networks and Learning Systems (2020) 1–14`doi:10.1109/TNNLS.2020.3002583`.

[113] T. Lesort, A. Stoian, D. Filliat, Regularization shortcomings for continual learning, arXiv preprint arXiv:1912.03049 (2019).

[114] J. Knoblauch, H. Husain, T. Diethe, Optimal continual learning has perfect memory and is np-hard, arXiv preprint arXiv:2006.05188 (2020).

[115] E. Belouadah, A. Popescu, Scail: Classifier weights scaling for class incremental learning, in: The IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 1266–1275.

[116] P. Roy, S. Ghosh, S. Bhattacharya, U. Pal, Effects of degradations on deep neural network architectures, arXiv preprint arXiv:1807.10108 (2018).

[117] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al., Memory replay gans: Learning to generate new categories without forgetting, in: Advances in Neural Information Processing Systems, 2018, pp. 5962–5972.

[118] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, C. Kanan, Remind your neural network to prevent catastrophic forgetting, in: European Conference on Computer Vision, Springer, 2020, pp. 466–483.

[119] X. Liu, C. Wu, M. Menta, L. Herranz, B. Raducanu, A. D. Bagdanov, S. Jui, J. van de Weijer, Generative feature replay for class-incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 226–227.

[120] L. Pellegrini, G. Graffieti, V. Lomonaco, D. Maltoni, Latent replay for real-time continual learning, arXiv preprint arXiv:1912.01100 (2019).

[121] K. Wang, J. van de Weijer, L. Herranz, Acae-remind for online continual learning with compressed feature replay, Pattern Recognition Letters 150 (2021) 122–129.

[122] S. Madireddy, A. Yanguas-Gil, P. Balaprakash, Neuromodulated neural architectures with local error signals for memory-constrained online continual learning, arXiv preprint arXiv:2007.08159 (2020).

[123] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, IEEE, 2005, pp. 539–546.

[124] T. L. Hayes, C. Kanan, Lifelong machine learning with deep streaming linear discriminant analysis, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 220–221.

[125] S. Pang, S. Ozawa, N. Kasabov, Incremental linear discriminant analysis for classification of data streams, IEEE transactions on Systems, Man, and Cybernetics, part B (Cybernetics) 35 (5) (2005) 905–914.

[126] S. Li, Y. Du, G. M. van de Ven, I. Mordatch, Energy-based models for continual learning, arXiv preprint arXiv:2011.12216 (2020).

[127] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, A tutorial on energy-based learning, Predicting structured data 1 (0) (2006).

[128] T. Hospedales, A. Antoniou, P. Micaelli, A. Storkey, Meta-learning in neural networks: A survey, arXiv preprint arXiv:2004.05439 (2020).

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

[130] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Caccia, I. Laradji, I. Rish, A. Lacoste, D. Vazquez, et al., Online fast adaptation and knowledge accumulation: a new approach to continual learning, arXiv preprint arXiv:2003.05856 (2020).

[131] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: ICML, 2017.

[132] K. Shmelkov, C. Schmid, K. Alahari, Incremental learning of object detectors without catastrophic forgetting, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3400–3409.

[133] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, L. Heck, Rilod: near real-time incremental learning for object detection at the edge, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 113–126.

[134] L. Chen, C. Yu, L. Chen, A new knowledge distillation for incremental object detection, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–7.

[135] X. Liu, H. Yang, A. Ravichandran, R. Bhotika, S. Soatto, Multi-task incremental learning for object detection.

[136] K. Joseph, J. Rajasegaran, S. Khan, F. S. Khan, V. Balasubramanian, L. Shao, Incremental object detection via meta-learning, arXiv preprint arXiv:2003.08798 (2020).

[137] M. Acharya, T. L. Hayes, C. Kanan, Rodeo: Replay for online object detection, arXiv preprint arXiv:2008.06439 (2020).

[138] S. Sodhani, S. Chandar, Y. Bengio, Toward training recurrent neural networks for lifelong learning, Neural computation 32 (1) (2020) 1–35.

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

[140] G. Arora, A. Rahimi, T. Baldwin, Does an lstm forget more than a cnn? an empirical study of catastrophic forgetting in nlp, in: Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association, 2019, pp. 77–86.

[141] T. Chen, I. Goodfellow, J. Shlens, Net2net: Accelerating learning via knowledge transfer, arXiv preprint arXiv:1511.05641 (2015).

[142] B. Ehret, C. Henning, M. R. Cervera, A. Meulemans, J. von Oswald, B. F. Grewe, Continual learning in recurrent neural networks with hy-

pernetworks, arXiv preprint arXiv:2006.12109 (2020).

[143] L. Duncker, L. Driscoll, K. V. Shenoy, M. Sahani, D. Sussillo, Organizing recurrent network dynamics by task-computation to enable continual learning, Advances in Neural Information Processing Systems 33 (2020).

[144] F.-K. Sun, C.-H. Ho, H.-Y. Lee, Lamol: Language modeling for lifelong language learning, in: International Conference on Learning Representations, 2019.

[145] Y. Li, L. Zhao, K. Church, M. Elhoseiny, Compositional language continual learning, in: International Conference on Learning Representations, 2019.

[146] F. Mi, L. Chen, M. Zhao, M. Huang, B. Faltings, Continual learning for natural language generation in task-oriented dialog systems, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, 2020, pp. 3461–3474.

[147] S. Mazumder, B. Liu, S. Wang, N. Ma, Lifelong and interactive learning of factual knowledge in dialogues, arXiv preprint arXiv:1907.13295 (2019).

[148] B. Liu, S. Mazumder, Lifelong learning dialogue systems: Chatbots that self-learn on the job, arXiv preprint arXiv:2009.10750 (2020).

[149] R. Del Chiaro, B. Twardowski, A. Bagdanov, J. van de Weijer, Ratt: Recurrent attention to transient tasks for continual image captioning, Advances in Neural Information Processing Systems 33 (2020).

[150] Z. Ke, B. Liu, H. Wang, L. Shu, Continual learning with knowledge transfer for sentiment classification, in: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2020.

[151] T. Liu, L. Ungar, J. Sedoc, Continual learning for sentence representations using conceptors, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 3274–3279.

[152] F. Mi, B. Faltings, Memory augmented neural model for incremental session-based recommendation, arXiv preprint arXiv:2005.01573 (2020).

[153] F. Yuan, G. Zhang, A. Karatzoglou, X. He, J. Jose, B. Kong, Y. Li, One person, one model, one world: Learning continual user representation without forgetting, arXiv preprint arXiv:2009.13724 (2020).

[154] F. Mi, X. Lin, B. Faltings, Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation, in: Fourteenth ACM Conference on Recommender Systems, 2020, pp. 408–413.

[155] Y. Xu, Y. Zhang, W. Guo, H. Guo, R. Tang, M. Coates, Graphsail: Graph structure aware incremental learning for recommender systems, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 2861–2868.

[156] B. Liu, Learning on the job: Online lifelong and continual learning., in: AAAI, 2020, pp. 13544–13549.

[157] T. Diethe, T. Borchert, E. Thereska, B. Balle, N. Lawrence, Continual learning in practice, arXiv preprint arXiv:1903.05202 (2019).

[158] C. S. Lee, A. Y. Lee, Clinical applications of continual learning machine learning, The Lancet Digital Health 2 (6) (2020) e279–e281.

[159] M. D. Lange, X. Jia, S. Parisot, A. Leonardis, G. Slabaugh, T. Tuytelaars, Unsupervised model personalization while preserving privacy and scalability: An open problem, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14463–14472.

Appendix A. Algorithms

In this section, we provide the algorithms for different memory update strategies described in Section 5.

Algorithm A1: Reservoir sampling
---
1 **procedure MemoryUpdate** (mem_sz, t, n, B)
3     $j \leftarrow 0$
5   **for** (x, y) *in* B **do**
7         $M \leftarrow |\mathcal{M}|$ ▷ Number of samples currently stored in the memory
9   **if** $M < \text{mem\_sz}$ **then**
11             $\mathcal{M}.\text{append}(x, y, t)$
12   **else**
14             $i = \text{randint}(0, n + j)$
16   **if** $i < \text{mem\_sz}$ **then**
18                 $\mathcal{M}[i] \leftarrow (\mathbf{x}, y, t)$ ▷ Overwrite memory slot
20         $j \leftarrow j + 1$
21   **return** $\mathcal{M}$

Algorithm A2: GSS-Greedy
---
2 **Input :** n, M
4 **Initialize:** $\mathcal{M}, C$
6 **Receive:** (x, y)
8 **Update:** $(x, y, \mathcal{M})$
10 $X, Y \leftarrow \text{RandomSubset}(\mathcal{M}, n)$
12 $g \leftarrow \nabla \ell_\theta(x, y); G \leftarrow \nabla_\theta \ell(X, Y)$
14 $c = \max_i \left(\frac{\langle g, G_i \rangle}{\|g\| \|G_i\|} \right) + 1$ ▷ make the score positive
16 **if** $\text{len}(\mathcal{M}) >= M$ **then**
18   **if** $c < 1$ **then**
19   ▷ cosine similarity < 0
21         $i \sim P(i) = C_i / \sum_j C_j$
23         $r \sim \text{uniform}(0, 1)$
25   **if** $r < C_i / (C_i + c)$ **then**
27             $\mathcal{M}_i \leftarrow (x, y); C_i \leftarrow c$
29   **end if**
31   **end if**
33 **else**
35     $\mathcal{M} \leftarrow \mathcal{M} \cup \{(x, y)\}; C \cup \{c\}$
37 **end if**

Appendix B. Experiment Details

Appendix B.1. Dataset Detail

The summary of dataset statistics is provided in Table B.1.
The strength of each nonstationary type used in the experiments are summarized below.

- Noise: [0.0, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6]

- Occlusion: [0.0, 0.07, 0.13, 0.2, 0.27, 0.33, 0.4, 0.47, 0.53, 0.6]

- Blur: [0.0, 0.28, 0.56, 0.83, 1.11, 1.39, 1.67, 1.94, 2.22, 2.5]

Algorithm A3: Greedy Balancing Sampler
---
2 **Init:** counter $C_0 = \{\}, \mathcal{D}_0 = \{\}$ with capacity k. Online samples arrive from t = 1
4 **function** SAMPLE $(x_t, y_t, \mathcal{D}_{t-1}, \mathcal{Y}_{t-1})$ ▷ Input: New sample and past state
6     $k_c = \frac{k}{|\mathcal{Y}_{t-1}|}$
8   **if** $y_t \notin \mathcal{Y}_{t-1}$ *or* $C_{t-1}[y_t] < k_c$ **then**
10   **if** $\sum_i C_i >= k$ **then**
11   ▷ If memory is full, replace
13             $y_r = \text{argmax}(C_{t-1})$ ▷ Select largest class, break ties randomly
15             $(x_i, y_i) = \mathcal{D}_{t-1} \cdot \text{random}(y_r)$▷ Select random sample from class y_r
17             $\mathcal{D}_t = \left(\mathcal{D}_{t-1} - (x_i, y_i) \right) \cup (x_t, y_t)$
19             $C_t[y_r] = C_{t-1}[y_r] - 1$
21   **else**
22   ▷ If memory has space, add
24             $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (x_t, y_t)$
26   **end if**
28         $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup y_t$
30         $C_t[y_t] = C_{t-1}[y_t] + 1$
32   **end if**
34   **return** $\mathcal{D}_t$
36 **end function**

Appendix B.2. Implementation Details

This section describes the implementation details of each method, including the hyperparameter grid considered for each dataset (see Table B.2). As we described in Section 7 of the main paper, the first D^{CV} tasks are used for hyperparameter tuning to satisfy the requirement that the model does not see the data of a task more than once, and D^{CV} is set to 2 in this work.

- EWC++: We set the α in Eq. (8) to 0.9 as suggested in the original paper. We tune three hyperparameters in EWC++, learning rate (LR), weight decay(WD) and λ in Eq. (7).

- LwF: We set the temperature factor $T = 2$ as the original paper and other CL papers. The coefficient λ for $\mathcal{L}_{KD}$ is set to $\frac{|C_{new}|}{|C_{old}| + |C_{new}|}$ following the idea from [43] and the coefficient for $\mathcal{L}_{CE}$ is set to $1 - \lambda$.

- ER: The reservoir sampling used in *MemoryUpdate* follows Algorithm A1 in Appendix A. For *MemoryRetrieval*, we randomly select samples with mini-batch size of 10 irrespective of the size of the memory buffer.

- MIR: To reduce the computational cost, MIR selects C random samples from the memory buffer as the candidate set to perform the criterion search. We tune LR, WD as well as C.

- GSS: For every incoming sample, GSS computes the cosine similarity of the new sample gradient to n gradient vectors of samples randomly drawn from the memory buffer (see Algorithm A2 in Appendix A). Other than LR and WD, we also tune n.

Dataset	#Task	#Train/task	#Test/task	#Class	Image Size	Setting
Split MiniImageNet	20	2500	500	100	3x84x84	OCI
Split CIFAR-100	20	2500	500	100	3x32x32	OCI
CORe50-NC	9	12000~24000	4500~9000	50	3x128x128	OCI
NS-MiniImageNet	10	5000	1000	100	3x84x84	ODI
CORe50-NI	8	15000	44972	50	3x128x128	ODI

Table B.1. Summary of dataset statistics

- iCaRL: We replace the herding-based [90] memory update method with reservoir sampling to accommodate the online setting. We use random sampling for *MemoryRetrieval* and tune LR and WD.

- A-GEM: We use reservoir sampling for *MemoryUpdate* and random sampling for *MemoryRetrieval* and tune LR and WD.

- CN-DPM: CN-DPM is much more sensitive to hyperparameters than others. We need to use different hyperparameter grids for different scenarios and datasets. Other than LR, we tune α, the concentration parameter controlling how sensitive the model is to new data and classifier_chill cc, the parameter used to adjust the VAE loss to have a similar scale as the classifier loss.

- GDumb: We use batch size of 16 and 30 epochs for all memory sizes. We clip gradient norm with max norm 10.0 and tune LR and WD.

Appendix C. Additional Experiments and Results

Appendix C.1. More Results for OCI Setting

Fig C.1, C.2 and C.3 show the average accuracy measured by the end of each task on Split CIFAR-100, Mini-ImageNet and CORe50-NC with three different memory buffer sizes (1k, 5k, 10k).

Appendix C.2. OCI Tricks on Split Mini-ImageNet

We evaluate the tricks described in Section 6.2 on Split Mini-ImageNet. As shown in Table C.1 and Fig. C.4, we find similar results as in Split CIFAR-100 that all tricks are beneficial. LB and KDC* are most useful when the memory buffer is small, and NCM and RV are more effective when the memory buffer is large. One main difference is that NCM is not as effective as in CIFAR-100 with a 10k memory buffer as base methods with NCM cannot outperform the best OCI performance.

Appendix C.3. More Results for ODI Setting

Fig. C.5, C.6 and C.7 show the average accuracy measured by the end of each task on Mini-ImageNet-Noise, Mini-ImageNet-Occlusion and CORe50-NI with three different memory buffer sizes (1k, 5k, 10k).

Method	CIFAR-100	Mini-ImageNet	CORe50-NC	NS-MiniImageNet	CORe50-NI
EWC++		LR: [0.0001, 0.001, 0.01, 0.1]			
		WD: [0.0001, 0.001], λ: [0, 100, 1000]			
LwF		LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1]			
		WD: [0.0001, 0.001, 0.01, 0.1]			
ER		LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1]			
		WD: [0.0001, 0.001, 0.01, 0.1]			
MIR		LR:[0.0001, 0.001, 0.01, 0.1]			
		WD: [0.0001, 0.001], C: [25, 50, 100]			
GSS		LR:[0.0001, 0.001, 0.01, 0.1]			
		WD: [0.0001, 0.001], n: [10, 20, 50]			
iCaRL		LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1]			
		WD: [0.0001, 0.001, 0.01, 0.1]			
A-GEM		LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1]			
		WD: [0.0001, 0.001, 0.01, 0.1]			
CN-DPM	LR: [0.0001, 0.001, 0.01, 0.1]	[0.001, 0.005, 0.01]	[0.001, 0.01]	[0.001, 0.005, 0.01]	[0.001, 0.01]
	cc: [0.001, 0.01, 0.1]	[0.001, 0.0015, 0.002]	[0.0005, 0.001, 0.002]	[0.0005, 0.001, 0.002]	[0.0005, 0.001, 0.002]
	α: [-100, -300, -500]	[-1200, -1000, -800]	[-1200, -1000, -800, -300]	[-15000, -5000, -500]	[-1200, -1000, -800, -300]
GDumb		LR: .001, 0.01, 0.1], WD:[0.0001, 0.000001]			

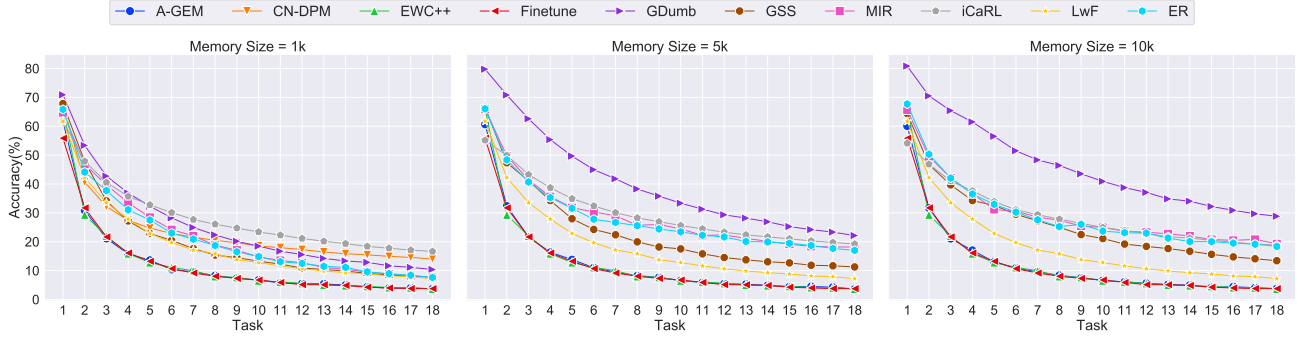**Table B.2.** Hyperparameter grid for the compared methods.

Fig. C.1. The average accuracy measured by the end of each task for the OCI setting on Split CIFAR-100 with three memory sizes.

Finetune					3.4 ± 0.2				
Offline					51.9 ± 0.5				
Method		A-GEM			ER			MIR	
Buffer Size	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k	M=1k	M=5k	M=10k
NA	3.4 ± 0.2	3.7 ± 0.3	3.3 ± 0.3	6.4 ± 0.9	14.5 ± 2.1	15.9 ± 2.0	6.4 ± 0.9	16.5 ± 2.1	21.0 ± 1.1
LB	5.8 ± 0.8	5.8 ± 0.5	5.4 ± 0.9	14.4 ± 2.1	19.3 ± 2.3	22.1 ± 1.1	**17.1 ± 0.9**	21.7 ± 0.7	23.0 ± 0.8
KDC	8.0 ± 1.1	7.5 ± 1.5	8.2 ± 1.7	12.3 ± 2.5	15.4 ± 0.4	14.6 ± 2.1	14.3 ± 0.5	15.8 ± 0.4	15.5 ± 0.5
KDC*	5.6 ± 0.4	5.5 ± 0.5	5.4 ± 0.4	**16.4 ± 0.8**	20.3 ± 2.5	23.0 ± 3.1	16.4 ± 0.6	25.1 ± 0.8	26.1 ± 0.9
MI	3.5 ± 0.2	3.7 ± 0.2	3.6 ± 0.3	6.4 ± 0.6	16.3 ± 1.3	24.1 ± 1.3	6.6 ± 0.6	15.2 ± 1.1	22.0 ± 1.9
SS	5.7 ± 0.9	6.2 ± 0.8	5.7 ± 0.8	12.5 ± 1.9	20.5 ± 2.1	24.1 ± 1.1	14.2 ± 1.0	21.9 ± 0.8	24.7 ± 0.9
RV	4.1 ± 0.2	**19.9 ± 3.7**	**25.5 ± 4.7**	11.4 ± 0.6	**32.1 ± 0.8**	**36.3 ± 1.5**	9.1 ± 0.5	**29.9 ± 0.7**	**37.3 ± 0.5**
NCM	**10.2 ± 0.4**	11.7 ± 1.5	13.0 ± 0.5	14.2 ± 0.7	26.7 ± 0.7	28.2 ± 0.6	13.6 ± 0.6	26.4 ± 0.7	28.6 ± 0.4
Best OCI	14.7 ± 0.4	21.1 ± 1.7	31.0 ± 1.4	14.7 ± 0.4	21.1 ± 1.7	31.0 ± 1.4	14.7 ± 0.4	21.1 ± 1.7	31.0 ± 1.4

Table C.1. Performance of compared tricks for the OCI setting on Split Mini-ImageNet. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k. Best OCI refers to the best performance from the compared methods in Table 7.

Fig. C.2. The average accuracy measured by the end of each task for the OCI setting on Split Mini-ImageNet with three memory sizes.

Fig. C.3. The average accuracy measured by the end of each task for the OCI setting on CORe50-NC with three memory sizes.
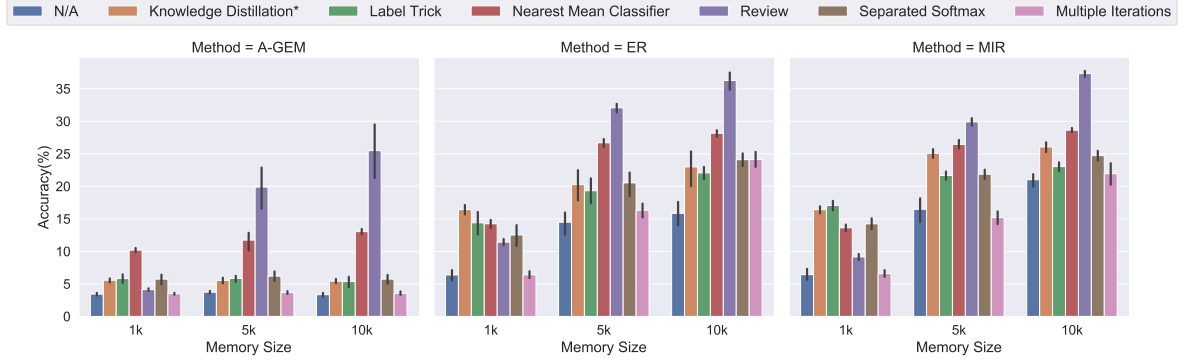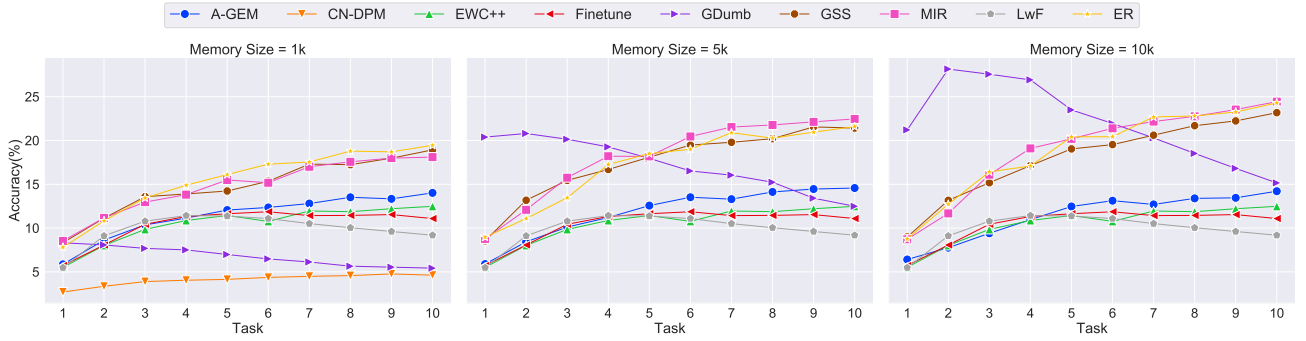

Fig. C.4. Comparison of various tricks for the OCI setting on Split Mini-ImageNet. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k.

Fig. C.5. The average accuracy measured by the end of each task for the ODI setting on Mini-ImageNet-Noise with three memory sizes.
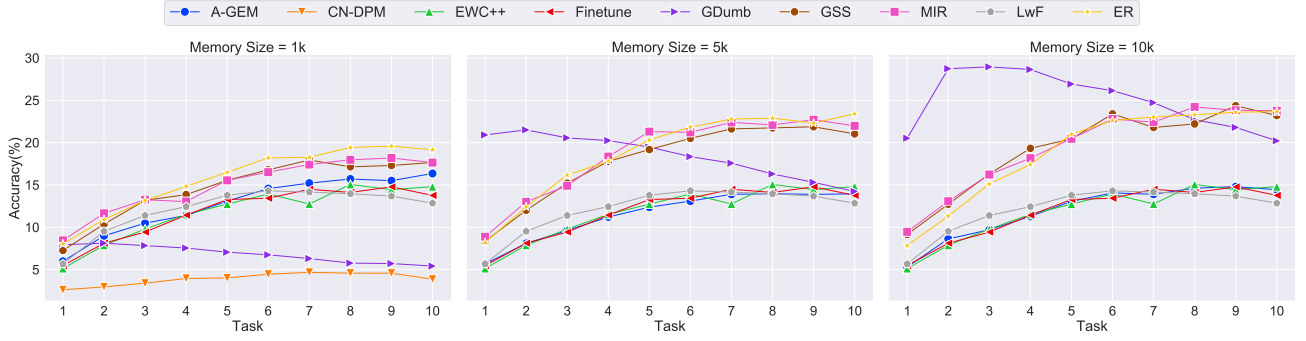
Fig. C.6. The average accuracy measured by the end of each task for the ODI setting on Mini-ImageNet-Occlusion with three memory sizes.
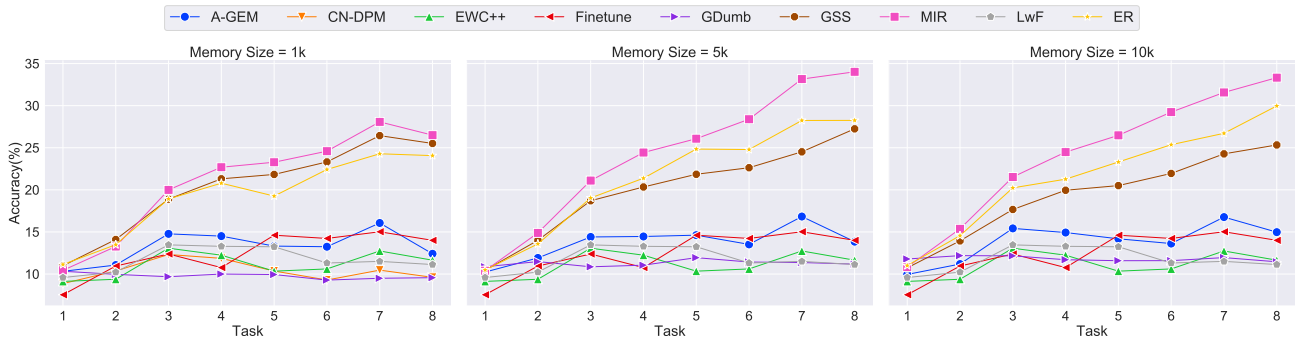

Fig. C.7. The average accuracy measured by the end of each task for the ODI setting on CORe50-NI with three memory sizes.