Deep Class-Incremental Learning: A Survey

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, Ziwei Liu

Abstract—Deep models, *e.g.*, CNNs and Vision Transformers, have achieved impressive achievements in many vision tasks in the closed world. However, novel classes emerge from time to time in our ever-changing world, requiring a learning system to acquire new knowledge continually. For example, a robot needs to understand new instructions, and an opinion monitoring system should analyze emerging topics every day. Class-Incremental Learning (CIL) enables the learner to incorporate the knowledge of new classes incrementally and build a universal classifier among all seen classes. Correspondingly, when directly training the model with new class instances, a fatal problem occurs — the model tends to *catastrophically forget* the characteristics of former ones, and its performance drastically degrades. There have been numerous efforts to tackle catastrophic forgetting in the machine learning community. In this paper, we survey comprehensively recent advances in deep class-incremental learning and summarize these methods from three aspects, *i.e.*, data-centric, model-centric, and algorithm-centric. We also provide a rigorous and unified evaluation of 16 methods in benchmark image classification tasks to find out the characteristics of different algorithms empirically. Furthermore, we notice that the current comparison protocol ignores the influence of memory budget in model storage, which may result in unfair comparison and biased results. Hence, we advocate fair comparison by aligning the memory budget in evaluation, as well as several memory-agnostic performance measures. The source code to reproduce these evaluations is available at https://github.com/zhoudw-zdw/CIL_Survey/.

Index Terms—Class-Incremental Learning, Continual Learning, Lifelong Learning, Catastrophic Forgetting

1 Introduction

RECENT years have witnessed the rapid progress of deep learning, where deep neural networks have achieved or even surpassed human-level performances in many fields [1], [2], [3]. The typical training process of a deep network requires pre-collected datasets in advance, *e.g.*, large-scale images [4] or texts [5] — the network undergoes training process of the pre-collected dataset multiple epochs. However, training data is often with stream format in the open world [6], [7]. These streaming data cannot be held for long due to storage constraints [8], [9] or privacy issues [10], [11], requiring the model to be updated incrementally with only new class instances. Such requirements trigger the prosperity of the Class-Incremental Learning (CIL) field, aiming to continually build a holistic classifier among all seen classes. The fatal problem in CIL is called *catastrophic forgetting*, *i.e.*, directly optimizing the network with new classes will erase the knowledge of former ones and result in irreversible performance degradation. Hence, how to effectively resist catastrophic forgetting becomes the core problem in building CIL models.

Figure 1 depicts the typical setting of CIL. Training data emerge sequentially in the stream format. In each timestamp, we can get a new training dataset (denoted as 'task' in the figure) and need to update the model with the new classes. For example, the model learns 'birds' and 'dogs' in the first task, 'tigers' and 'fish' in the second task, 'monkeys' and 'sheep' in the third task, etc. Afterward, the model is tested among all seen classes to evaluate whether it has discrimination for them. A good model should strike a balance between depicting the characteristics of new classes and preserving the pattern of formerly learned old classes. This trade-off is also known as the 'stability-plasticity dilemma' in neural system [12], where stability denotes the ability to maintain former knowledge and plasticity represents the ability to adapt to new patterns.

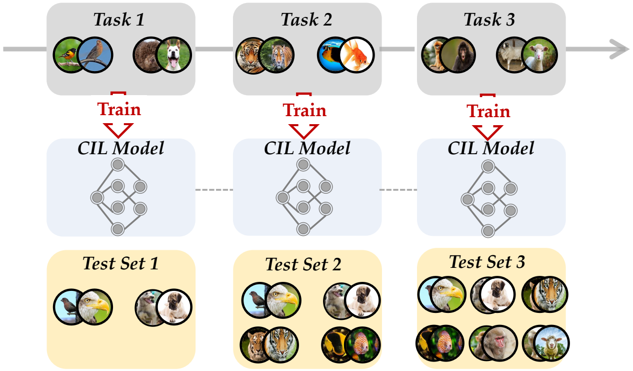


Figure 1: The setting of class-incremental learning. Non-overlapping classes arrive sequentially, and the model needs to learn to classify all the classes incrementally. After the learning process of each task, the model is evaluated among all seen classes. An ideal model should perform well in the newly learned classes and remember the former without catastrophic forgetting.

Apart from class-incremental learning, there are other fine-grained settings addressing the incremental learning problem, *e.g.*, Task-Incremental Learning (TIL) and Domain-Incremental Learning (DIL). We show these three protocols in Figure 2. TIL is a similar setting to CIL, and both of them observe incoming new classes in new tasks. However, the difference lies in the inference stage, where CIL requires the model to differentiate among all classes. By contrast, TIL only requires classifying the instance among the corresponding task space. In other words, it does not require *cross-task* discrimination ability. Hence, TIL is *easier* than CIL, which can be seen as a particular case of CIL.

- *D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, and D.-C. Zhan are with State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China; E-mail: {zhoudw, wangqiwei, qizh, yehj, zhandc}@lamda.nju.edu.cn*
Work done when D.-W. Zhou was a visiting scholar at NTU.
- *Z. Liu is with S-Lab, School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798. E-mail: ziwei.liu@ntu.edu.sg*
- *H.-J. Ye and Z. Liu are corresponding authors.*

Manuscript received April 19, 2005; revised August 26, 2015.

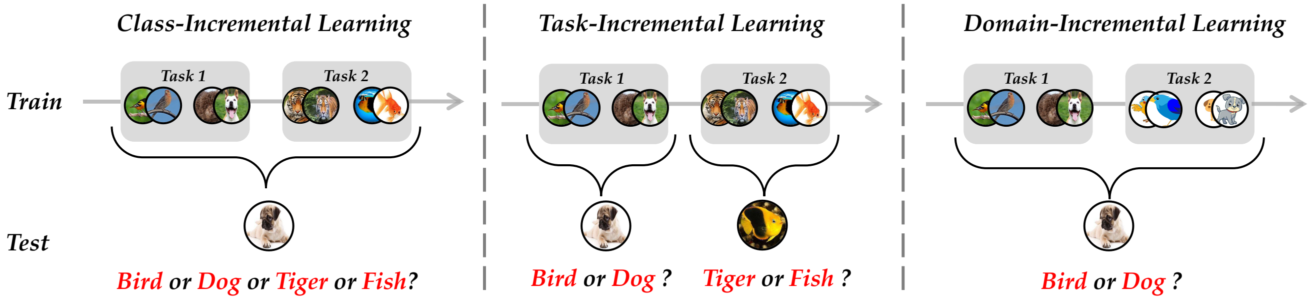

Class-Incremental Learning
Task-Incremental Learning
Domain-Incremental Learning
Train
Test
Task 1
Task 2
Bird or Dog or Tiger or Fish?
Bird or Dog ?
Tiger or Fish ?
Bird or Dog ?

Figure 2: The setting of *Class-Incremental Learning* (CIL), *Task-Incremental Learning* (TIL), and *Domain-Incremental Learning* (DIL). CIL and TIL share the same training protocol, while TIL is much easier during inference, *i.e.*, only requiring classifying among corresponding label spaces. DIL refers to the data stream with distribution change, where new tasks contain the same classes from different domains, *e.g.*, cartoon and clip-art.

On the other hand, DIL concentrates on the scenario with concept drift or distribution change [13], [14], where new tasks contain instances from different domains but with the same label space. In this case, new domains correspond to the images in clip-art format. In this paper, we concentrate on the CIL setting, which is a more challenging scenario in the open world.

There is also research about CIL before the prosperity of deep learning [15]. Typical methods try to solve the catastrophic forgetting problem with traditional machine learning models. However, most of them address the incremental learning within two tasks, *i.e.*, the model is only updated with a single new stage [16], [17], [18]. Furthermore, the rapid development of data collection and processing requires the model to grasp long-term and massive-scale data streams that traditional machine learning models cannot handle. Correspondingly, deep neural networks with powerful representation ability well suit these requirements. As a result, deep learning-based CIL is becoming a hot topic in the machine learning and computer vision community.

There are several related surveys discussing the incremental learning problem. For example, [11] focuses on the task-incremental learning problem and provides a comprehensive survey. [19] is a related survey on the class-incremental learning field, while it only discusses and evaluates the methods till 2020. However, with the rapid development of the CIL field, many great works are emerging day by day, which substantially boost the performance of benchmark settings [20], [21], [22], [23]. On the other hand, with the prosperity of Vision Transformer (ViT) [24] and pre-trained models, a heated discussion about ViT in CIL is attracting the attention of the community. Other surveys either focuses on the specific field [25], [26], [27] or lacking the performance evolution among state-of-the-arts [28], [29], [30]. Hence, it is urgent to provide an up-to-date survey containing popular methods to speed up the development of the CIL field.

In this paper, we aim for a comprehensive review of deep class-incremental learning methods from three aspects, *i.e.*, data-centric, model-centric, and algorithm-centric. We also provide a holistic comparison among different kinds of methods over benchmark datasets, *i.e.*, CIFAR100 [31] and ImageNet100/1000 [4]. On the other hand, we highlight an important factor in CIL model evaluation, *i.e.*, *memory budget*, and advocate fair comparison among different methods with an aligned budget. Correspondingly, we holistically evaluate the extensibility of CIL models with budget-agnostic measures. In general, the contribution of this survey can be summarized as follows:

- We provide a comprehensive survey of deep CIL, including problem definitions, benchmark datasets, and different families of CIL methods. We organize these algorithms taxonomically (Table 1) and chronologically (Figure 3) to give a holistic overview of state-of-the-art.
- We provide a rigorous and unified comparison among different methods on several publicly available datasets, including traditional CNN-backed and modern ViT-backed methods. We also discuss the insights and summarize the common rules to inspire future research.
- To boost real-world applications, CIL models should be deployed not only on high-performance computers but also on edge devices. Therefore, we advocate evaluating different methods holistically by emphasizing the effect of memory budgets. Correspondingly, we provide a comprehensive evaluation of different methods given specific budgets as well as several new performance measures.

The rest of this paper is organized as follows. First, we give the problem definition of class-incremental learning in Section 2. Afterward, we organize current CIL methods taxonomically in Section 3 and provide comprehensive evaluations in Section 4. Finally, we summarize future directions of class-incremental learning in Section 5 and then conclude this paper.

2 Preliminaries

In this section, we first give the problem formulation of class-incremental learning. We also discuss the other variations and CIL model decomposition. Afterward, we introduce a commonly used auxiliary set, *i.e.*, exemplar set for CIL.

2.1 Problem Formulation

Definition 1. Class-Incremental Learning *aims to learn from an evolutive stream with incoming new classes [32]. Assume there is a sequence of B training tasks*[1] $\left\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\right\}$ *without overlapping classes, where* $\mathcal{D}^b = \left\{\left(\mathbf{x}_i^b, y_i^b\right)\right\}_{i=1}^{n_b}$ *is the b-th incremental step with n_b training instances.* $\mathbf{x}_i^b \in \mathbb{R}^D$ *is an instance of class $y_i \in Y_b$, Y_b is the label space of task b, where $Y_b \cap Y_{b'} = \varnothing$ for $b \neq b'$. We can only access data from $\mathcal{D}^b$ when training task b. The ultimate goal of CIL is to continually build a classification model for all classes. In other words, the model should not only acquire the knowledge from the current task $\mathcal{D}^b$ but also preserve the knowledge from former tasks. After each task, the trained model is evaluated over all seen classes $\mathcal{Y}_b = Y_1 \cup \cdots Y_b$.*

[1] Also known as 'session,' 'phase,' and 'stage.'

Formally, CIL aims to fit a model $f(\mathbf{x}) : X \to \mathcal{Y}_b$, which minimizes the expected risk:

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_t^1 \cup \cdots \mathcal{D}_t^b} \mathbb{I}(y \neq f(\mathbf{x})) \,, \tag{1}$$

where $\mathcal{H}$ is the hypothesis space, $\mathbb{I}(\cdot)$ is the indicator function which outputs 1 if the expression holds and 0 otherwise. $\mathcal{D}_t^b$ denotes the data distribution of task b. A good CIL model satisfying Eq. 1 has discriminability among all classes, which not only works well on new classes but also preserves the knowledge of former ones.

Class Overlapping: Typical CIL setting assumes $Y_b \cap Y_{b'} = \varnothing$ for $b \neq b'$, *i.e.*, there are no overlapping classes in different tasks. However, in the real world, it is common to observe the old classes emerging in new tasks [33]. When $Y_b \cap Y_{b'} \neq \varnothing$, the setting is called blurry class-incremental learning (Blurry CIL) [34], [35], [36]. It enables the model to revisit former instances in the later stage, which weakens the learning difficulty. Hence, this paper focuses on the typical CIL setting with no overlapping classes.

In the following discussions, we decompose the CIL model into the embedding module and linear layers, *i.e.*, $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$,[2] where $\phi(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$, $W \in \mathbb{R}^{d \times |\mathcal{Y}_b|}$. The linear layer can be further decomposed into the combination of classifiers: $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_{|\mathcal{Y}_b|}]$, where $\boldsymbol{w}_k \in \mathbb{R}^d$, $|\cdot|$ denotes the size of the set. The logits are then passed to the Softmax activation for further optimization, *i.e.*, the output probability on class k is denoted as:

$$\mathcal{S}_k(f(\mathbf{x})) = \frac{\exp^{\boldsymbol{w}_k^\top \phi(\mathbf{x})/\tau}}{\sum_{j=1}^{|\mathcal{Y}_b|} \exp^{\boldsymbol{w}_j^\top \phi(\mathbf{x})/\tau}} \,, \tag{2}$$

where τ is the temperature parameter.

Online CIL: Although data comes with stream format in CIL, the model can conduct multi-epoch training with each training task $\mathcal{D}^b$, *i.e.*, offline training within each task. There are some works addressing fully online (one-pass) class-incremental learning, where each batch can be processed once and then dropped [37], [38]. It is a specific case of the current setting, and we concentrate on the generalized CIL setting in this paper.

2.2 Exemplars and Exemplar Set

As defined in Definition 1, in each incremental task, the model is only able to access the current dataset $\mathcal{D}^b$. This helps to preserve user privacy and release the storage burden. However, this restriction is relaxed in some cases, and the model can keep a relatively small set, namely *exemplar set*, to reserve the representative instances from former tasks.

Definition 2. Exemplar Set[3] *is an extra collection of instances from former tasks $\mathcal{E} = \{(\mathbf{x}_j, y_j)\}_{j=1}^M$, $y_j \in \mathcal{Y}_{b-1}$. With the help of the exemplar set, the model can utilize $\mathcal{E} \cup \mathcal{D}^b$ for the update within each task. The model manages the exemplar set after the training process of each task.*

Exemplar Set Management: Since the data stream is evolving, there are two main strategies to manage the exemplar set in CIL [93]. The first way is to keep a fixed number of exemplars per class, *e.g.*, R per class. Under such circumstances, the size of the exemplar set will grow as the data stream evolves — the model keeps $R|\mathcal{Y}_b|$ after the b-th task. This will result in a linearly *growing* memory budget, which is inapplicable in real-world learning systems. To this end, another strategy advocates saving a fixed number of exemplars, *e.g.*, M. The model keeps $[\frac{M}{|\mathcal{Y}_b|}]$ instances per class, where $[\cdot]$ denotes floor function. It helps to keep a fixed size of exemplars in the memory and release the storage burden. In this paper, we use the second strategy to organize the exemplar set.

Exemplar Selection: Exemplars are representative instances of each known class, which need to be selected from the entire training set. An intuitive way to choose the exemplars is to randomly sample exemplars for each class, which results in diverse instances. By contrast, a commonly used strategy is called herding [32], [118], aiming to select the most *representative* ones of each class. Given the instance set $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ from class y, herding first calculates the class center with current embedding $\phi(\cdot)$:

$$\mu_y \leftarrow \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i) \,. \tag{3}$$

Afterward, it calculates and ranks the distance of each instance to the class center $\|\mu_y - \phi(\mathbf{x}_i)\|$ in ascending order. The exemplars are then selected based on the ranking, *e.g.*, the top-$[\frac{M}{|\mathcal{Y}_b|}]$ instances with the least distance. Since the class center can be seen as the most representative pattern of each class, selecting exemplars near the class center also enhances the representativeness of exemplars. Herding is now a commonly-used strategy to select exemplars in CIL, and we also adopt it in this paper.

3 Class-Incremental Learning: Taxonomy

There are numerous works addressing class-incremental learning in recent years, raising a heated discussion among machine learning and computer vision society. We organize these methods taxonomically from three aspects, *i.e.*, data-centric, model-centric, and algorithm-centric, as shown in Table 1. Data-centric methods concentrate on solving CIL with exemplars, which can be further classified into data replay and data regularization. Model-centric methods either regularize the model parameters from drifting away or expand the network structure for stronger representation ability. Lastly, algorithm-centric methods either utilize knowledge distillation to resist forgetting or rectify the bias in the CIL model. We list the representative methods chronologically in Figure 3 to show the research focus of different periods. In the following sections, we will discuss CIL methods from these three aspects.

3.1 Data-Centric Class-Incremental Learning

Facing the notorious catastrophic forgetting, data-centric methods seek help from extra data, *e.g.*, exemplars. An intuitive way is to utilize former data for rehearsal, which enables the model to review former classes and resist forgetting. On the other hand, some works build regularization terms with the extra data, aiming to control the optimization direction to avoid catastrophic forgetting.

3.1.1 Data Replay

'Replay' is important in human cognition system [119] — a student facing final exams shall go over the textbooks to recall former memory and knowledge. This phenomenon can also be extended to the network training process, where a network can overcome catastrophic forgetting by revisiting former exemplars. We denote the sequential training process finetuning the model with $\mathcal{D}^b$ as 'Finetune,' whose loss function can be denoted as:

$$\mathcal{L} = \sum_{(\mathbf{x},y)\in\mathcal{D}^b} \ell(f(\mathbf{x}), y) \,, \tag{4}$$

[2] We omit the bias term for ease of discussions.

[3] Also known as the 'replay buffer' and 'memory buffer.' We use them interchangeably in this paper.

Table 1: The taxonomy of class-incremental learning. We classify CIL methods into data-centric, model-centric, and algorithm-centric methods and further decompose them into six sub-categories. The shade color in the last column denotes the subcategory, which is consistent with Figure 3.

Algorithm Category	Subcategory		Reference
§ 3.1 Data-Centric Class-Incremental Learning	§ 3.1.1 Data Replay	Direct Replay	[35], [39], [40], [41], [42], [43], [44], [45], [46], [47]
		Generative Replay	[48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58]
	§ 3.1.2: Data Regularization		[40], [59], [60], [61], [62], [63]
§ 3.2 Model-Centric Class-Incremental Learning	§ 3.2.1 Dynamic Networks	Neuron Expansion	[64], [65], [66]
		Backbone Expansion	[20], [21], [22], [67], [68], [69], [70], [71]
		Prompt Expansion	[23], [72], [73], [74], [75], [76]
	§ 3.2.2: Parameter Regularization		[39], [77], [78], [79], [80], [81], [82], [83], [84]
§ 3.3 Algorithm-Centric Class-Incremental Learning	§ 3.3.1 Knowledge Distillation	Logit Distillation	[32], [85], [86], [87], [88], [89], [90], [91], [92]
		Feature Distillation	[93], [94], [95], [96], [97], [98], [99], [100]
		Relational Distillation	[101], [102], [103], [104], [105]
	§ 3.3.2 Model Rectify	Feature Rectify	[106], [107], [108], [109], [110], [111]
		Logit Rectify	[87], [93], [112], [113], [114]
		Weight Rectify	[115], [116], [117]

where $\ell(\cdot,\cdot)$ measures the discrepancy between inputs, *e.g.*, cross-entropy loss. Finetune is also known as the baseline method for class-incremental learning since it only concentrates on learning the new concepts in the current task. Consequently, it suffers severe forgetting since the model pays no attention to going over former ones. To this end, an intuitive way [32] is to save an extra exemplar set $\mathcal{E}$ (as defined in Definition 2) and include it into the model updating process:

$$\mathcal{L} = \sum_{(\mathbf{x},y)\in(\mathcal{D}^b\cup\mathcal{E})} \ell(f(\mathbf{x}), y) \, . \tag{5}$$

Comparing Eq. 5 to Eq. 4, we can find that exemplars are concatenated to the training set for updating, enabling the retrospective review of former knowledge when learning new concepts.

Utilizing the exemplar set for rehearsal is simple yet effective, which leads to numerous following works. The exemplar sampling process is also similar to the active learning protocol [120], and some works propose corresponding sampling measures to select the *informative* ones. For example, [39] suggests sampling exemplars with high prediction entropy and near the decision boundary. The model will achieve higher generalization ability by replaying these 'hard' exemplars. [35] proposes estimating exemplars' uncertainty via data augmentation. They choose instances with large prediction diversity by aggregating the predictions of multiple augmented instances. Similarly, [40] proposes to sample exemplars with a greedy strategy in online incremental learning. It proves that exemplar selection is equivalent to maximizing the diversity of exemplars with parameters gradient as the feature. If there are no explicit task boundaries, [41], [42] explore the reservoir sampling process to ensure the exemplars are i.i.d. sampled. [43] proposes to formulate the replay process into a bi-level optimization and keep intact predictions on some anchor points of past tasks. [44] introduces data replay in prototypical network [45], and utilizes the exemplars as pseudo-prototypes for embedding evaluation.

Since exemplars are raw images, directly saving a set of instances may consume enormous memory costs. To this end, several works are proposed to build a *memory-efficient* replay buffer [22]. [46] argues that extracted features $\phi(\mathbf{x})$ are with lower dimension than raw images $\mathbf{x}$ and proposes to save features in the exemplar set to release the burden. Similarly, [47] proposes to keep low-fidelity images instead of raw images to save memory budget. However, since the distributions of extracted features and low-fidelity images may differ from the raw images, an extra adaptation process is needed for these methods, adding to the algorithm's complexity.

The above-mentioned methods achieve competitive performance by replaying former instances in the memory. Moreover, apart from directly saving instances in the exemplar set, generative models show the potential to model the distribution and generate instances [121], [122], [123], which have also been applied to class-incremental learning. We denote the aforementioned methods directly saving instances for replay as 'direct replay,' and the methods utilize generative models as 'generative replay.'

There often exist two models in generative replay-based CIL, *i.e.*, the generative model for data generation and the classification model for prediction. GR [48] firstly proposes to utilize the generative adversarial network (GAN) [121] in CIL. In each updating process, it utilizes the GAN to generate the instances from former classes and then updates GAN and classification model with both old and new classes. ESGR [49] extends GR by saving extra exemplars. It also proposes to train the separate GAN for each incremental task, which does not require updating GAN incrementally. [50] extends GR by introducing the dynamic parameter generator for model adaptation at test time. FearNet [51] explores brain-inspired CIL and uses a dual-memory system in which new memories are consolidated from a network for recent memories. Recently, [52], [53], [54] explore the application of conditional GAN [124] in CIL, and [55], [56] adopt variational auto-encoders (VAE) [122] to model data distribution. Similarly, [57], [58] model each class into a Gaussian distribution and sample instances directly from the class center.

However, the performance of generative replay methods relies on the quality of generated data, which may fail in complex, large-scale inputs [125]. On the other hand, sequentially updating

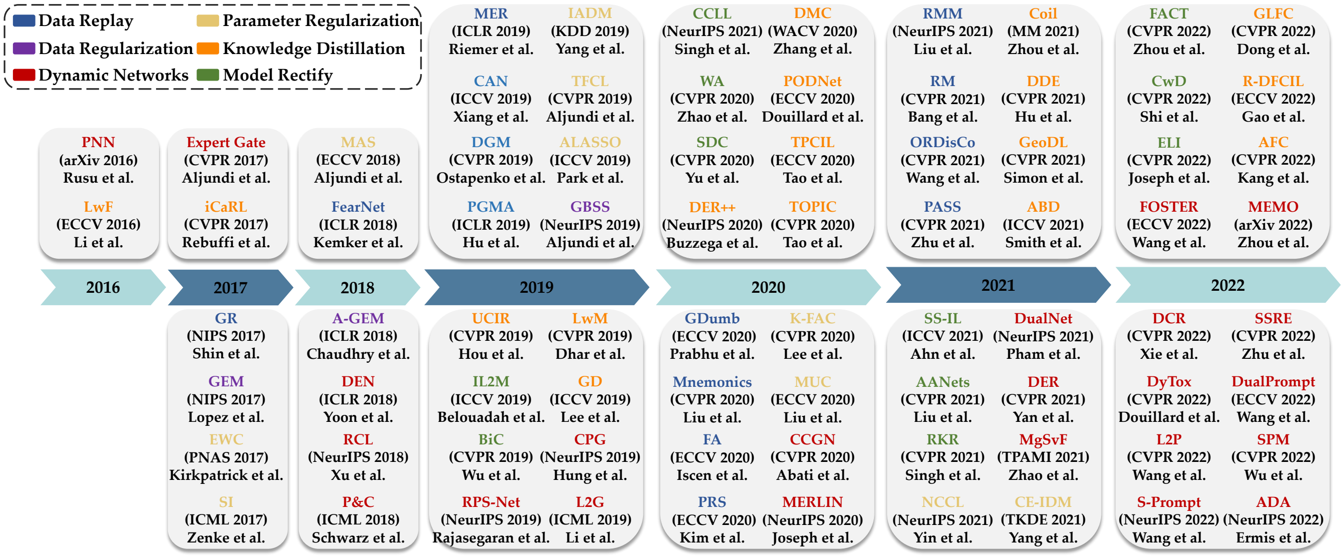

Data Replay
Parameter Regularization
Data Regularization
Knowledge Distillation
Dynamic Networks
Model Rectify
PNN (arXiv 2016) Rusu et al.
LwF (ECCV 2016) Li et al.
Expert Gate (CVPR 2017) Aljundi et al.
iCaRL (CVPR 2017) Rebuffi et al.
MAS (ECCV 2018) Aljundi et al.
FearNet (ICLR 2018) Kemker et al.
MER (ICLR 2019) Riemer et al.
IADM (KDD 2019) Yang et al.
CAN (CVPR 2019) Xiang et al.
TFCL (CVPR 2019) Aljundi et al.
DGM (CVPR 2019) Ostapenko et al.
ALASSO (ICCV 2019) Park et al.
PGMA (ICLR 2019) Hu et al.
GBSS (NeurIPS 2019) Aljundi et al.
CCLL (NeurIPS 2021) Singh et al.
DMC (WACV 2020) Zhang et al.
WA (CVPR 2020) Zhao et al.
PODNet (ECCV 2020) Douillard et al.
SDC (CVPR 2020) Yu et al.
TPCIL (ECCV 2020) Tao et al.
DER++ (NeurIPS 2020) Buzzega et al.
TOPIC (CVPR 2020) Tao et al.
RMM (NeurIPS 2021) Liu et al.
Coil (MM 2021) Zhou et al.
RM (CVPR 2021) Bang et al.
DDE (CVPR 2021) Hu et al.
ORDisCo (CVPR 2021) Wang et al.
GeoDL (CVPR 2021) Simon et al.
PASS (CVPR 2021) Zhu et al.
ABD (ICCV 2021) Smith et al.
FACT (CVPR 2022) Zhou et al.
GLFC (CVPR 2022) Dong et al.
CwD (CVPR 2022) Shi et al.
R-DFCIL (ECCV 2022) Gao et al.
ELI (CVPR 2022) Joseph et al.
AFC (CVPR 2022) Kang et al.
FOSTER (ECCV 2022) Wang et al.
MEMO (arXiv 2022) Zhou et al.
2016
2017
2018
2019
2020
2021
2022
GR (NIPS 2017) Shin et al.
GEM (NIPS 2017) Lopez et al.
EWC (PNAS 2017) Kirkpatrick et al.
SI (ICML 2017) Zenke et al.
A-GEM (ICLR 2018) Chaudhry et al.
DEN (ICLR 2018) Yoon et al.
RCL (NeurIPS 2018) Xu et al.
P&C (ICML 2018) Schwarz et al.
UCIR (CVPR 2019) Hou et al.
LwM (CVPR 2019) Dhar et al.
IL2M (ICCV 2019) Belouadah et al.
GD (ICCV 2019) Lee et al.
BiC (CVPR 2019) Wu et al.
CPG (NeurIPS 2019) Hung et al.
RPS-Net (NeurIPS 2019) Rajasegaran et al.
L2G (ICML 2019) Li et al.
GDumb (ECCV 2020) Prabhu et al.
K-FAC (CVPR 2020) Lee et al.
Mnemonics (CVPR 2020) Liu et al.
MUC (ECCV 2020) Liu et al.
FA (ECCV 2020) Iscen et al.
CCGN (CVPR 2020) Abati et al.
PRS (ECCV 2020) Kim et al.
MERLIN (NeurIPS 2020) Joseph et al.
SS-IL (ICCV 2021) Ahn et al.
DualNet (NeurIPS 2021) Pham et al.
AANets (CVPR 2021) Liu et al.
DER (CVPR 2021) Yan et al.
RKR (CVPR 2021) Singh et al.
MgSvF (TPAMI 2021) Zhao et al.
NCCL (NeurIPS 2021) Yin et al.
CE-IDM (TKDE 2021) Yang et al.
DCR (CVPR 2022) Xie et al.
SSRE (CVPR 2022) Zhu et al.
DyTox (CVPR 2022) Douillard et al.
DualPrompt (ECCV 2022) Wang et al.
L2P (CVPR 2022) Wang et al.
SPM (CVPR 2022) Wu et al.
S-Prompt (NeurIPS 2022) Wang et al.
ADA (NeurIPS 2022) Ermis et al.

Figure 3: The roadmap of class-incremental learning. We organize representative methods chronologically to show the concentration at different stages. Different colors of these methods denote the sub-categories in Table 1. Knowledge distillation and data replay dominated the research before the year 2021, while model rectify and dynamic networks became popular after 2021.

the generative model also suffers the catastrophic forgetting problem [126], [127], [128], which in turn increases the forgetting in incremental model training.

3.1.2 Data Regularization

Apart from directly replaying former data, another group of works tries to regularize the model with former data and control the optimization direction. Since learning new classes will result in the catastrophic forgetting of old ones, the intuitive idea is to make sure that optimizing the model for new classes will *not hurt* former ones. GEM [59] aims to find the model that satisfies:

$$f^* = \mathop{\mathrm{argmin}}_{f\in\mathcal{H}} \sum_{(\mathbf{x},y)\in\mathcal{D}^b} \ell\left(f(\mathbf{x}),y\right) \qquad (6)$$

$$\text{s.t.} \sum_{(\mathbf{x}_j,y_j)\in\mathcal{E}} \ell\left(f(\mathbf{x}_j),y_j\right) \leq \sum_{(\mathbf{x}_j,y_j)\in\mathcal{E}} \ell\left(f^{b-1}(\mathbf{x}_j),y_j\right),$$

where f^{b-1} stands for the incremental model after training the last task $\mathcal{D}^{b-1}$. Eq. 6 optimizes the model with a restriction, which requires the loss calculated with the exemplar set not to exceed the former model. Since exemplars are representative instances from former classes, GEM strikes a balance between learning new classes and preserving former knowledge. Furthermore, it transforms the constraints in Eq. 6 into:

$$\langle g, g_{old}\rangle := \left\langle \frac{\partial \ell\left(f(\mathbf{x}),y\right)}{\partial\theta}, \frac{\partial \ell\left(f,\mathcal{E}\right)}{\partial\theta} \right\rangle \geq 0\,, \qquad (7)$$

where g, g_{old} denotes the gradients of the current updating step and exemplar set, respectively. Eq. 7 requires the angle between gradients to be *acute*. If all the inequality constraints are satisfied, then the proposed parameter update is unlikely to increase the loss at previous tasks. However, if violations occur, GEM proposes to project the gradients g to the closest gradient $\tilde{g}$ satisfying the constraints. GEM further transforms the optimization into a Quadratic Program (QP) problem. However, since the regularization in Eq. 7 is defined among all exemplars, it requires calculating loss among exemplar set and solving the QP problem in each optimization step. Hence, optimizing GEM is very time-consuming. To this end, A-GEM [60] is proposed to speed up the optimization by relaxing the constraints in Eq. 7 into a random batch. A similar idea is also adopted in [40].

There are other methods to address the regularization problem with exemplars. For example, Adam-NSCL [61] proposes to sequentially optimize network parameters by projecting the candidate parameter update into the approximate null space of all previous tasks. OWM [62] only allows weights modification in the direction orthogonal to the subspace spanned by all previously learned inputs. LOGD [63] further decomposes the gradients into shared and task-specific ones. In model updating, the gradient should be close to the gradient of the new task, consistent with the gradients shared by all old tasks, and orthogonal to the space spanned by the gradients specific to the old tasks.

3.1.3 Discussions about Data-Centric Methods

Data-centric CIL algorithms mainly concentrate on resisting forgetting with the help of former data. Replay is a simple yet effective strategy, which has been widely applied to image-based camera localization [129], semantic segmentation [130], [131], [132], video classification [133], and action recognition [95]. However, since the exemplar set only saves a tiny portion of the training set, data replay may suffer the overfitting problem [59], [134]. Simultaneously, the data-imbalance problem [135], [136] also occurs due to the gap between the few-shot exemplars and the many-shot training set. Several works address this problem with balanced sampling [21], [112]. On the other hand, data regularization methods rely on specific assumptions, *e.g.*, treating the loss of exemplars as the *indicator* of forgetting. These assumptions may not stand in some cases [60], which results in poor performance.

Lastly, both of these methods rely on saving exemplars from the history, which may violate user privacy in some cases [137]. Hence, when developing class-incremental learning in privacy-sensitive scenarios, we should consider learning in an exemplar-free [57], [107], [138], [139] manner.

3.2 Model-Centric Class-Incremental Learning

Model-centric CIL methods mainly concentrate on model evolvement in the learning process. For example, dynamic networks

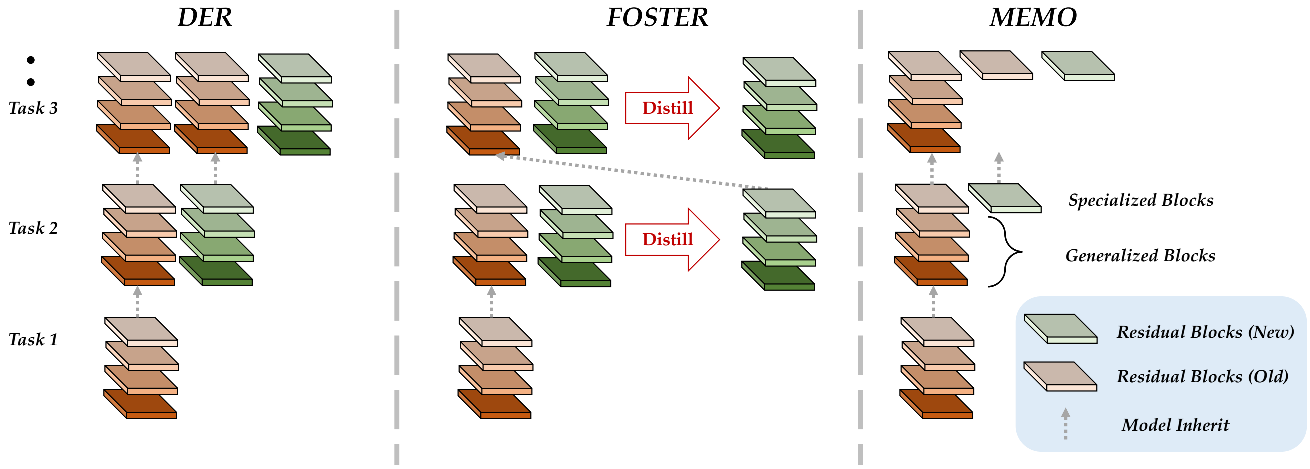


Figure 4: Illustration of network structure evolving in backbone expansion. **Left:** DER [20] expands a new backbone per incremental task. **Middle:** FOSTER [21] adds an extra model compression stage, which maintains limited model storage. **Right:** MEMO [22] decouples the network structure and only expands specialized blocks.

assume the network's capacity is *finite* and propose expanding the network when needed to enhance representation ability. On the other hand, parameter regularization methods estimate the importance of parameters and regularize important ones to prevent them from drifting away.

3.2.1 Dynamic Networks

Deep neural networks are proven to produce task-specific features [140], [141]. For example, when the training dataset contains 'cars,' the model tends to depict the wheels and windows. However, if the model is updated with new classes containing 'cats,' the features would be adapted for beards and stripes. Since the capacity of a model is limited, adapting to new features will result in the overwriting of old ones and forgetting [64]. Hence, utilizing the extracted features for beards and strides is inefficient for recognizing a car. To this end, dynamic networks are designed to dynamically adjust the model's representation ability to fit the evolving data stream. There are several ways to expand representation ability, and we divide them into three sub-groups, *i.e.*, neuron expansion, backbone expansion, and prompt expansion.

Early works focus on *neuron expansion*, which adds neurons when the representation ability is not enough to capture new classes. DEN [64] formulates the adjusting process into selection, expansion, duplication, and elimination. Facing a new incremental task, the model first selectively retrains the neurons that are relevant to this task. If the retrained loss is still above some threshold, DEN considers expanding new neurons top-down and eliminating the useless ones with group-sparsity regularization. Afterward, it calculates the neuron-wise drift and duplicates neurons that drift too much from the original values. Apart from heuristically expanding and shrinking the network structure, RCL [65] formulates the network expansion process into a reinforcement learning problem and searches for the best neural architecture for each incoming task. Similarly, Neural Architecture Search (NAS) [142] is also adopted to find the optimal structure for each of the sequential tasks [66].

Expanding neurons shows competitive results with expandable representation. Correspondingly, several works try to duplicate the backbone network for stronger representation ability. PNN [67] proposes learning a new backbone for each new task and fixing the former in incremental learning. It also adds layer-wise connections between old and new models to reuse former features. Expert Gate [68] also expands the backbone per incremental task, while it requires learning an extra gate to map the instance to the most suitable pathway during inference. To release the expansion cost, P&C [69] suggests a progression-compression protocol — it first expands the network to learn representative representations. Afterward, a compression process is conducted to control the total budget. [70], [71] maintain a dual-branch network for class-incremental learning, one for fast adaptation and one for slow adaption. Recently, DER [20] has been proposed to address the CIL problem. Similar to PNN, it expands a new backbone when facing new tasks and aggregates the features with a larger FC layer. Take the second incremental task for an example, where the model output is aggregated as:

$$f(\mathbf{x}) = W_{new}^{\top}[\phi_{old}(\mathbf{x}), \phi_{new}(\mathbf{x})]\,, \tag{8}$$

where ϕ_{old} is the former backbone, and ϕ_{new} is the newly initialized backbone. $[\cdot,\cdot]$ denotes feature aggregation, and $W_{new} \in \mathbb{R}^{2d\times|\mathcal{Y}_b|}$ is the newly initialized FC layer. In the model updating process, the old backbone is frozen to maintain former knowledge:

$$\mathcal{L} = \sum_{k=1}^{|\mathcal{Y}_b|} -\mathbb{I}(y=k)\log \mathcal{S}_k(W_{new}^{\top}[\bar{\phi}_{old}(\mathbf{x}), \phi_{new}(\mathbf{x})])\,, \tag{9}$$

where $\bar{\phi}_{old}(\mathbf{x})$ denotes the old backbone is frozen. DER also adopts an auxiliary loss to differentiate between old and new classes. Eq. 9 depicts a way to continually expand the model with new features. Under such circumstances, if the old backbone is optimized with 'cars,' the features extracted by ϕ_{old} are then representative of wheels and windows. The new backbone trained with 'cats' is responsible for extracting beards and stripes. Since the old backbone is frozen in later stages, learning new classes will not overwrite the features of old ones, and forgetting is alleviated. Figure 4 (left) depicts the model evolution of DER.

DER has achieved powerful performance with humble forgetting. However, saving a backbone per task requires numerous memory size, and many works are proposed to obtain expandable features with a *limited* memory budget. FOSTER [21] formulates the learning process in Eq. 9 as a feature-boosting [143] problem. It argues that not all expanded features are needed for incremental learning and need to be integrated to reduce redundancy. For example, suppose old classes contain 'tigers,' and new classes contain 'zebras.' In that case, the stripe will be a useful feature that could be extracted by both old and new backbones. Under such circumstances, forcing the new backbone to extract the same kind of features is less effective for recognition. Hence, FOSTER adds an extra model compression process by knowledge distillation [144]:

$$\min_{f_s(\mathbf{x})} \mathrm{KL}\left(\mathcal{S}\left(f_t(\mathbf{x})\right) \| \mathcal{S}\left(f_s(\mathbf{x})\right)\right) . \tag{10}$$

Eq. 10 aims to find the student model f_s that has the same discrimination ability as the teacher model f_t by minimizing the discrepancy between them. The teacher is the *frozen* expanded model with two backbones: $f_t(\mathbf{x}) = W_{new}^\top[\phi_{old}(\mathbf{x}), \phi_{new}(\mathbf{x})]$, and the student is the newly initialized model $f_s(\mathbf{x}) = W^\top\phi(\mathbf{x})$. Hence, the number of backbones is consistently *limited* to a single one, and the memory budget will not suffer catastrophic expansion. Figure 4 (middle) depicts the model evolution of FOSTER.

Recently, MEMO [22] has been proposed to address the memory problem in CIL, aiming to enable model expansion with the *least* budget cost. It finds that in class-incremental learning, shallow layers of different models are similar, while deep layers are diverse. In other words, shallow layers are more generalizable, while deep layers are specific to the task, making expanding shallow layers less memory-efficient for CIL. Hence, MEMO proposes to decouple the backbone at middle layers: $\phi(\mathbf{x}) = \phi_s(\phi_g(\mathbf{x}))$, where specialized block ϕ_s corresponds to the deep layers in the network, while generalized block ϕ_g corresponds to the rest shallow layers. Compared to DER, MEMO only expands specialized blocks ϕ_s, and transforms Eq. 9 into:

$$\sum_{k=1}^{|\mathcal{Y}_b|} -\mathbb{I}(y=k) \log \mathcal{S}_k(W_{new}^\top[\phi_{s_{old}}(\phi_g(\mathbf{x})), \phi_{s_{new}}(\phi_g(\mathbf{x}))]) ,$$

which indicates that task-specific deep layers can be built for each task upon the shared shallow layers $\phi_g(\mathbf{x})$. Figure 4 (right) depicts the model evolution of MEMO.

Recently, Vision Transformer (ViT) [24] has attracted the attention of the computer vision community, and many works tend to design CIL learners using ViT as the backbone. DyTox [23] is the first work to explore ViT in CIL, which finds that model expansion in ViT is much easier than in convolutional networks. In DyTox, only task tokens are expanded for each new task, which requires much less memory than saving the whole backbone. Similarly, L2P [72] and DualPrompt [73] explore how to build CIL learners with pre-trained ViT. They borrow ideas from Visual Prompt Learning (VPT) [145] to incrementally finetune the model with prompts. In L2P, the pre-trained ViT is frozen during the learning process, and the model only optimizes the prompts to fit new patterns. The prompt pool is defined as:

$$\mathbf{P} = \{P_1, P_2, \cdots, P_M\} , \tag{11}$$

where M is the total number of prompts, $P_i \in \mathbb{R}^{L_p \times d}$ is a single prompt with token length L_p and the same embedding size d as the instance embedding $\phi(\mathbf{x})$. The prompts are organized as key-value pairs — each instance selects the most similar prompts in the prompt pool via KNN search. It obtains instance-specific predictions by adapting the input embeddings as:

$$\mathbf{x}_p = [P_{s_1}; \cdots; P_{s_N}; \phi(\mathbf{x})] , \quad 1 \leq N \leq M . \tag{12}$$

P_{s_j} are selected prompts via KNN search. The adapted embeddings are then fed into the self-attention layers [146] to obtain instance-specific representations. CODA-Prompt [74] extends the prompt search with the attention mechanism. Apart from pre-trained ViT, S-Prompt [75] utilizes the pre-trained language-vision model CLIP [147] for incremental learning, which simultaneously learns language prompts and visual prompts to boost representative embeddings. [76] extends CLIP with spatial and temporal prompts for incremental video classification.

3.2.2 Parameter Regularization

Dynamic networks seek to adjust model capacity with data evolves. However, if the model structure is fixed and unchangeable, how to adjust the plasticity to resist catastrophic forgetting? Parameter regularization methods consider that the contribution of each parameter to the task is not equal. Hence, they seek to evaluate the *importance* of each parameter to the network and keep the important ones static to maintain former knowledge.

Typical works estimate a distribution over the model parameters and use it as the prior when learning new tasks. Due to the large amounts of parameters, the estimation process often assumes them to be independent. EWC [77] is the first work addressing parameter regularization. It maintains an importance matrix with the same scale of the network, *i.e.*, Ω. Denote the k-th model parameters as θ_k, the importance of θ_k is represented by $\Omega_k \geq 0$ (the larger Ω_k indicates θ_k is more important). Apart from the training loss in Eq. 4 to learn new classes, EWC builds an additional regularization term to remember old ones:

$$\mathcal{L} = \ell(f(\mathbf{x}), y) + \frac{1}{2}\lambda \sum_k \Omega_k(\theta_k^{b-1} - \theta_k)^2 . \tag{13}$$

The parameter-wise regularization term is calculated based on two parts. θ_k^{b-1} denotes the k-th parameter after learning last task $\mathcal{D}^{b-1}$. Hence, $(\theta_k^{b-1} - \theta_k)^2$ represents the parameter drift from the last stage, and Ω_k weighs it to make sure important parameters do not shift away from the last stage. Since the model at the last stage represents the 'old' knowledge, consolidating important parameters can fix the knowledge from being forgotten.

Eq. 13 depicts a way to penalize essential parameters, and there are different ways to calculate the importance matrix Ω. In EWC, Fisher information matrix [148] is adopted to estimate Ω. However, the importance calculation in EWC is conducted at the end of each task, which ignores the optimization dynamics along the model training trajectory. To this end, SI [78] proposes to estimate Ω in an online manner and weigh the importance via its contribution to loss decay. RWalk [39] combines these importance estimation techniques. [79], [80] resort to an extra unlabeled dataset for online evaluation. IMM [81] finds a maximum of the mixture of Gaussian posteriors with the estimated Fisher information matrix. IADM [82] and CE-IDM [83] analyze the capacity and sustainability of different layers in class-incremental learning and find that different layers have different characteristics in CIL. In detail, shallow layers converge faster but have limited representation ability. By contrast, deep layers converge slowly while having powerful discrimination abilities. Hence, IADM augments EWC with an ensemble of different layers and learns layer-wise importance matrix in an online manner. K-FAC [84] extends the Fisher information matrix approximation with the Kronecker factorization technique.

3.2.3 Discussions about Model-Centric Methods

Model-centric methods either design model structure adjustment techniques or build generalization terms to resist forgetting. Apart from these groups, there are works addressing network masks to divide a large network into sub-networks for each task [149], [150], [151], [152]. However, deciding the activation of a specific sub-network requires the task identifier or learning extra task classifiers. On the other hand, several works [153], [154], [155] propose to design specific modules for incremental new tasks, *e.g.*, adapters [156]. However, manually handcrafting these modules requires heuristic designs or task-specific priors.

Learning dynamic networks, especially backbone expansion methods, have achieved state-of-the-art performance in recent years [22]. However, it often requires expandable memory budgets, which is unsuitable for incremental learning on edge devices. Secondly, pre-trained models are needed for the prompt expansion methods. However, a pre-trained model is not always available for some specific downstream tasks, *e.g.*, face recognition [157], speech recognition [158], etc. Therefore, how to get rid of the dependence on pre-trained models is essential for these methods in real-world applications.

On the other hand, parameter regularization methods have attracted the attention of the community in the early years. However, estimating parameter importance requires saving the matrix with the same scale as the model. It faces the same risk as dynamic networks that the memory budget is linearly increasing. On the other hand, the importance matrix may conflict at different incremental stages [92], making it hard to optimize the model.

3.3 Algorithm-Centric Class-Incremental Learning

Algorithm-centric CIL methods focus on designing algorithms to maintain the model's knowledge in former tasks. For example, the core idea of knowledge distillation-based CIL methods is to build the mapping between old and new models and reflect the characteristics of the old model in the updating process. On the other hand, model rectify-based methods aim to discover and reduce the bias in incremental models.

3.3.1 Knowledge Distillation

Training data is evolving in the incremental learning process, requiring tuning the model sequentially. We can denote the model after the previous stage f^{b-1} as the 'old model' and the current updating model f as the 'new model.' Assuming the old model is a good classifier for all the seen classes in $\mathcal{Y}_{b-1}$, how can we utilize it to resist forgetting in the new model?

To enable the old model to assist the new model, an intuitive way utilizes the concept proposed in [144], [159], [160], *i.e.*, knowledge distillation (KD). KD enables the knowledge transfer from a teacher model to the student model, with which we can teach the new model not to forget. There are several ways to build the distillation relationship, and we divide these KD-based methods into three subgroups, *i.e.*, logit distillation, feature distillation, and relational distillation.

LwF [85], [86] is the first success to apply knowledge distillation into CIL. Similar to Eq. 13, it builds the regularization term via knowledge distillation to resist forgetting:

$$\mathcal{L} = \underbrace{\ell(f(\mathbf{x}), y)}_{\text{Learning New Classes}} + \underbrace{\sum_{k=1}^{|\mathcal{Y}_{b-1}|} -\mathcal{S}_k(f^{b-1}(\mathbf{x})) \log \mathcal{S}_k(f(\mathbf{x}))}_{\text{Remembering Old Classes}} , \tag{14}$$

where the old model f^{b-1} is frozen during updating. The regularization term builds the mapping between the old and new models by forcing the predicted probability among old classes to be the same. Given a specific input $\mathbf{x}$, the output probability of the k-th class reveals the semantic similarity of the input to this class. Hence, Eq. 14 forces the semantic relationship of the old model and new model to be the same and resists forgetting. iCaRL [32] extends LwF with the exemplar set, which helps to further recall former knowledge during incremental learning. Additionally, it drops the fully-connected layers and follows [161] to utilize nearest-mean-of-exemplars during inference. Eq. 14 strikes a trade-off between old and new classes, where the former part aims to learn new classes and the latter one maintains old knowledge. Since the number of old and new classes may differ in different incremental stages, BiC [87] extends Eq 14 by introducing a *dynamic* trade-off term:

$$\mathcal{L} = (1-\lambda)\ell(f(\mathbf{x}), y) + \lambda \sum_{k=1}^{|\mathcal{Y}_{b-1}|} -\mathcal{S}_k(f^{b-1}(\mathbf{x})) \log \mathcal{S}_k(f(\mathbf{x})) ,$$

where $\lambda = \frac{|\mathcal{Y}_{b-1}|}{|\mathcal{Y}_b|}$ denotes the proportion of old classes among all classes. It increases as incremental tasks evolve, indicating that the model pays more attention to old ones.

LwF inspires the community to build the mapping between models, making knowledge distillation a useful tool in CIL. D+R [88] suggests changing the first part in Eq. 14 into a distillation loss by training an extra expert model. GD [89] proposes to select wild data for model distillation and designs a confidence-based sampling method to effectively leverage external data. Similarly, DMC [90] proposes to train a new model in each incremental stage and then compress them into a single model via an extra unlabeled dataset. Finally, if no additional data is available for knowledge distillation, ABD [91] proposes distilling synthetic data for incremental learning. These methods only concentrate on utilizing the old model to help resist forgetting in the new model. However, COIL [92] suggests conducting *bidirectional* distillation with co-transport, where semantic relationships between old and new models are both utilized.

Apart from distilling the logits, some works propose to distill the intermediate product of deep models, *e.g.*, extracted features. UCIR [93] replaces the regularization term in Eq. 14 into:

$$\mathcal{L} = \ell(f(\mathbf{x}), y) + (1 - \langle \frac{\phi^{b-1}(\mathbf{x})}{\|\phi^{b-1}(\mathbf{x})\|}, \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|} \rangle) . \tag{15}$$

Eq. 15 forces the features extracted by the new embedding module to be the same as the old one, which is a stronger regularization than Eq. 14. Several works follow it to utilize feature distillation in CIL [94], [95], while others address distilling other products. LwM [96] suggests penalizing the changes in classifiers' attention maps to resist forgetting. AFC [97] conducts the distillation considering the importance of different feature maps. PODNet [98] minimizes the difference of the pooled intermediate features in the height and width directions instead of performing element-wise distillations. CVIC [162] decouples the distillation term into spatial and temporal features for video classification. DDE [99] distills the causal effect from the old training to preserve the old knowledge. GeoDL [100] proposes to conduct distillation based on the projection of two sets of features from old and new models.

However, both logit and feature distillation address the instance-wise mapping between old and new models. To reveal the structural information in model distillation, several works suggest conducting *relational* knowledge distillation [163]. The differences between these groups of knowledge distillation are shown in Figure 5.

To conduct relational distillation, a group of instances needs to be extracted, *e.g.*, triplets. We denote the extracted triplets as $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\}$, where $\mathbf{x}_i$ is called the anchor. In a triplet, the target

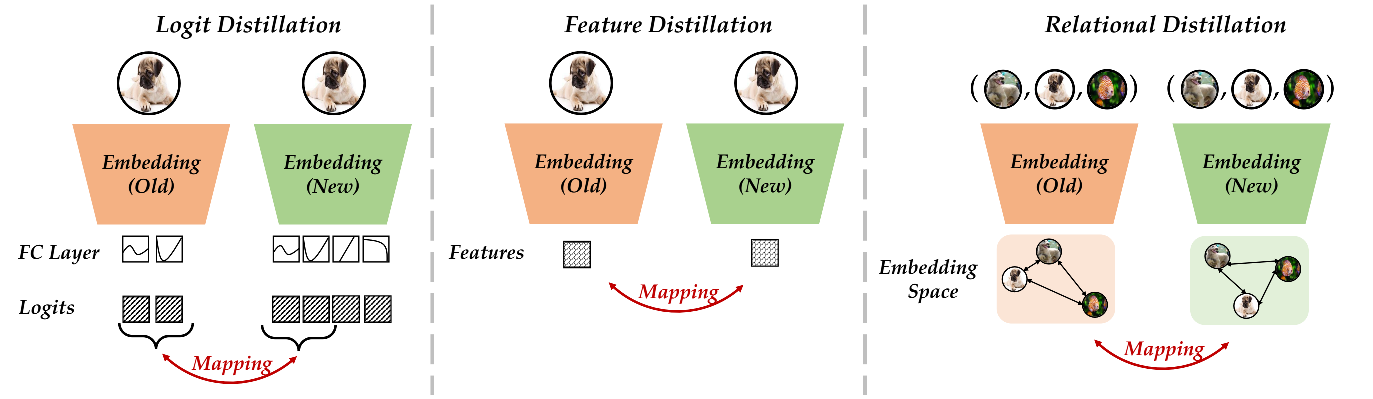


Figure 5: Illustration of knowledge distillation methods in class-incremental learning. **Left:** Logit distillation (*e.g.*, iCaRL [32]) aligns the model outputs to make the old model and new model share the same semantic relationship. **Middle:** Feature distillation (*e.g.*, UCIR [93]) aligns the features produced by the old and new models to make sure the new model does not forget old features. **Right:** Relational distillation (*e.g.*, R-DFCIL [101]) resorts to structural inputs, *e.g.*, triples, and aligns the input relationship of the old and new model.

neighbor $\mathbf{x}_j$ is similar to the anchor $\mathbf{x}_i$ with the same class, while the impostor $\mathbf{x}_k$ is dissimilar to $\mathbf{x}_i$ (usually from different classes). R-DFCIL [101] suggests mapping the angle among triplets:

$$\sum_{\{\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k\}\in\mathcal{D}^b} \left\|\cos\angle\mathbf{t}_i\mathbf{t}_j\mathbf{t}_k - \cos\angle\mathbf{s}_i\mathbf{s}_j\mathbf{s}_k\right\| , \tag{16}$$

where $\mathbf{t}_m = \phi^{b-1}(\mathbf{x}_m)$ is the representation in the old model's embedding space, and $\mathbf{s}_m = \phi(\mathbf{x}_m)$ denotes the representation in the current model. The cosine value is calculated in the corresponding embedding space. Eq. 16 provides a way to encode the old model's structural information into the new model and align the feature space softly. ERL [102] extends this regularization into few-shot CIL scenarios. TPCIL [103] models the relationship with the elastic Hebbian graph and penalizes the changing of the topological relations between vertices. TOPIC [104] further explores neural gas network to model the class-wise relationship. Apart from the triplet relationship, MBP [105] extends the regularization to the instance neighborhood, requiring the old and new models to have the same distance ranking in the neighborhood.

3.3.2 Model Rectify

Assuming we can get all the training datasets at once and shuffle them for training with multiple epochs, the model will not suffer any forgetting and will perform well among all classes. Such a protocol is known as the upper bound of class-incremental learning, denoted as 'Oracle.' However, since the models trained with incremental data suffer catastrophic forgetting, several methods try to find the abnormal behaviors in CIL models and rectify them like the oracle model. These abnormal behaviors include the output logits, classifier weights, and feature embedding.

The first method addressing the bias of the CIL model is UCIR [93]. It finds that the weight norm of new classes is significantly larger than old ones, and the model tends to predict instances as the new classes with larger weights. Hence, UCIR proposes to utilize a cosine classifier to avoid the influence of biased classifiers: $f(\mathbf{x}) = \frac{W}{\|W\|}^\top \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|}$. Hence, the weight norm will not influence model predictions in incremental learning. WA [115] further normalizes the weight after every optimization step. It also introduces weight clipping to ensure the predicted probability is proportionate to classifier weights. SS-IL [116] explains the reason for weight drifting, which is caused by the imbalance phenomena between old and new instances. Since the number of new class instances is much more than that of old ones, optimizing the model with cross-entropy loss will increase the weight of new classes and decrease old ones. Hence, SS-IL suggests separated softmax operation and task-wise knowledge distillation to alleviate the influence of imbalanced data. RPC [117] claims that the classifiers of all classes can be pre-allocated and fixed. This makes it impossible for classifiers to be biased toward new classes.

On the other hand, several works find that the predicted logits of new classes are much larger than old ones. E2E [112] proposes to finetune the fully-connected layers with a balanced dataset after each stage. Furthermore, BiC [87] proposes to attach an extra rectification layer to adjust the predictions. The extra layer only have two parameters, *i.e.*, re-scale parameter α and bias parameter β, and the rectified output for the k-th class is denoted as:

$$\hat{f}(\mathbf{x})_k = \begin{cases} \alpha \boldsymbol{w}_k^\top \phi(\mathbf{x}) + \beta, & k \in Y_b \\ \boldsymbol{w}_k^\top \phi(\mathbf{x}), & otherwise \end{cases} . \tag{17}$$

Note that only the logits for new classes ($k \in Y_b$) are rectified after each incremental task. BiC separates an extra validation set from the exemplar set, *i.e.*, $\mathcal{E} = \mathcal{E}_{train} \cup \mathcal{E}_{val}$, and uses the validation set to tune the rectification layer. On the other hand, IL2M [113] suggests re-scaling the outputs with historical statistics. Suppose an instance is predicted as a new class. In that case, the logits will be re-scaled to ensure the predictions of old and new classes follow the same distribution. AANets [114] learns to adaptively aggregate the stable and plastic blocks and adjust the predictions.

Lastly, since the embedding module is sequentially updated in CIL, some works try to rectify the biased representations of incremental models. For example, SDC [107] utilizes a nearest-mean-of-exemplars classifier, which calculates the class centers and assigns instances to the nearest class center. However, since the embedding is incrementally updated, the class centers calculated in the former stage may suffer a drift in the next stage, making the classification results unreliable. Since old class instances are not available in the current stage, SDC aims to calibrate the class centers of old classes with the drift of new classes. CwD [108] analyzes the differences of embeddings among the CIL model and oracle model and finds that the embeddings of the oracle model scatter more uniformly. It aims to make the CIL model similar to the oracle by enforcing the eigenvalues to be close. ConFiT [109] relieves the feature drift from the middle layers. Other works address the model weight rectification. CCLL [110] aims to calibrate the activation maps of old models during incremental learning. RKR [111] proposes to rectify the convolutional weights of old models when learning new tasks. Recently, FACT [106] depicts a new training paradigm for CIL, namely *forward compatible training*. Since the embedding space is endlessly adjusted for new classes, FACT proposes to pre-assign the embedding space for new classes to relieve the burden of embedding tuning.

3.3.3 Discussions about Algorithm-Centric Methods

Algorithm-centric methods design task-specific algorithms to resist forgetting. Knowledge distillation-based methods have been widely applied to various incremental tasks, *e.g.*, semantic segmentation [164], [165], [166], [167], person re-identification [168], [169], human action recognition [95], and federated learning [137]. However, since the knowledge distillation term aims to strike a balance between learning the new and remembering the old, it will undoubtedly suffer catastrophic forgetting compared to dynamic networks. [22] finds that knowledge distillation-based methods and dynamic networks have their advantages given different memory budgets. Specifically, knowledge distillation methods show stronger performance given limited memory, while dynamic networks require an adequate memory budget to perform competitively.

On the other hand, model rectification methods aim to reduce the biased inductive bias in the CIL model and align it to the oracle model. This line of work helps to understand the inherent factor of catastrophic forgetting. Apart from the rectification methods listed in Section 3.3.2, some methods [116], [170] address that the bias of the CIL model is from the imbalanced data stream. [171] finds that the embedding trained with contrastive loss suffers less forgetting than cross-entropy loss. [172] finds that the batch normalization layers [173] are biased in CIL and proposes to re-normalize the layer outputs. It is worth exploring other factors that could result in catastrophic forgetting and corresponding solutions in the future.

3.4 Discussions about Classification Criterion and Scopes of This Survey

In this paper, we divide current class-incremental learning methods into three groups based on the key point of these methods. For the methods discussing how to save and utilize exemplars, we divide them into data-centric methods. Similarly, for the methods designing model expansion and parameter-wise regularization, we denote them as model-centric methods. Lastly, we assign the methods that develop specific regularization to pass knowledge from the old to the new model and rectify the current model to the algorithm-centric group. It must be noted that these methods also learn from each other, and there is no strict boundary between them. For example, iCaRL [32] utilizes the knowledge distillation and exemplar replay, but we assign it to the knowledge distillation-based methods. In fact, data replay is becoming a standard protocol in the class-incremental learning community, which is widely adopted in most methods. As shown in Figure 3, dynamic networks and knowledge distillation methods have dominated publications in recent years. However, it does not indicate that traditional methods like data replay are no longer popular. By contrast, most of these methods rely on the exemplar set to get better performance.

On the other hand, in the discussion of this paper, we mainly focus on the development of class-incremental learning in the image classification field. The main reason is that most influential works about CIL address the image classification problem. Furthermore, it has shown the potential to apply the techniques in classification into other tasks, *e.g.*, semantic segmentation [164], [165], [166], [167], [174], object detection [175], [176], [177], [178], [179], [180], [181], video understanding [95], [133], medical surgery [182], [183], language model [184], [185], [186], [187], language-vision model [188], [189], etc. Hence, the scope of our discussion is mainly focused on image classification tasks.

4 Experimental Evaluation

In this section, we conduct comprehensive experiments to evaluate the performance of different kinds of CIL methods with benchmark datasets. We first introduce the benchmark experimental setting, dataset split, evaluation protocol, and implementation details. Afterward, we aim to compare these methods from three aspects:

- How do these methods perform on *benchmark* datasets?
- Are they *fairly* compared? How to fairly compare them?
- How to evaluate them with *memory-agnostic* measure?

Sections 4.2 and 4.3 answer the first question, and Section 4.5 answers the second question. We provide the holistic performance measures in Section 4.6 to answer the third question. We also analyze the influence of incremental learning by visualizing the confusion matrix in Section 4.4 and summarize the comparison results in Section 4.7. Finally, we report more experimental results, implementation details, and visualizations of fully connected layers in the supplementary material.

4.1 Experimental Settings

4.1.1 Benchmark Datasets

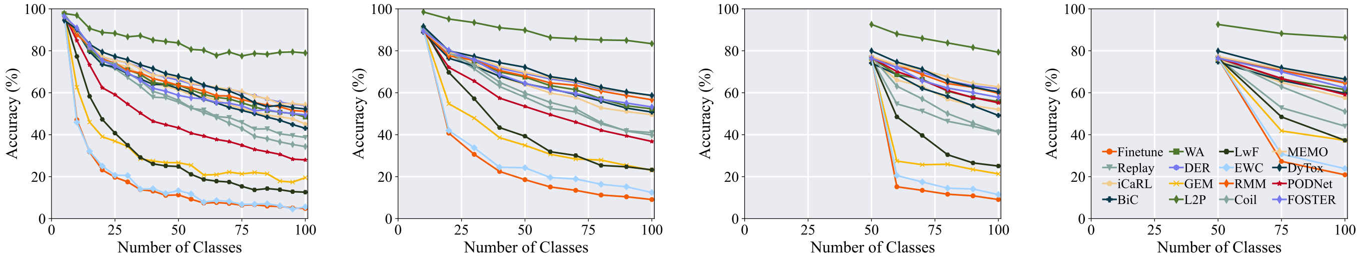
iCaRL [32] firstly formulates the comparison protocol of class-incremental learning, which was widely followed and compared in other works. It suggests using CIFAR100 [31] and ImageNet100/1000 [190] for evaluation. They are listed as:

- **CIFAR100**: There are 100 classes with 60,000 images, in which 50,000 are training instances and 10,000 are testing ones, with 100 images per class. Each image is represented by 32×32 pixels.
- **ImageNet1000**: is a large-scale dataset with 1,000 classes, with about 1.28 million images for training and 50,000 for validation. The spatial size of images varies.
- **ImageNet100**: is the subset of ImageNet1000 containing 100 classes [32]. These classes are selected from the first 100 classes after a random shuffle.

There are other works [75], [77], [82], [92], [104], [116], [139] utilizing MNIST [191], Core50 [192], Google Landmark-v2 [193], CUB200-2011 [194], TinyImageNet [195], and *mini*ImageNet [190] for model evaluation, while the aforementioned datasets are the most widely adopted in the current CIL community. Hence, we choose CIFAR100 and ImageNet100/1000 for model evaluation.

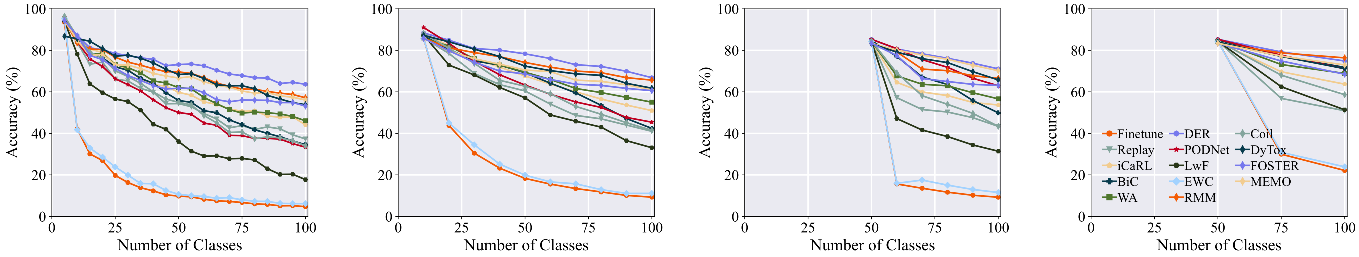
Dataset Split: Following the protocol defined in [32], all classes are first shuffled by Numpy random seed 1993. Afterward, there are two different ways to split the classes into incremental stages:

- **Train from scratch (TFS)**: splits all the classes equally into each incremental stage. For example, if there are B stages and C classes in total, each incremental task contains C/B classes for training.
- **Train from half (TFH)**: splits half of the total classes as the first incremental task and equally assigns the rest classes into the following stages. Specifically, it assigns $C/2$ classes to the first task and $C/2(B-1)$ classes to the rest tasks.



(a) CIFAR100 Base0 Inc5 (b) CIFAR100 Base0 Inc10 (c) CIFAR100 Base50 Inc10 (d) CIFAR100 Base50 Inc25

Figure 6: Incremental accuracy of different methods on CIFAR100. Legends are shown in (d).



(a) ImageNet100 Base0 Inc5 (b) ImageNet100 Base0 Inc10 (c) ImageNet100 Base50 Inc10 (d) ImageNet100 Base50 Inc25

Figure 7: Incremental accuracy of different methods on ImageNet100. Legends are shown in (d).

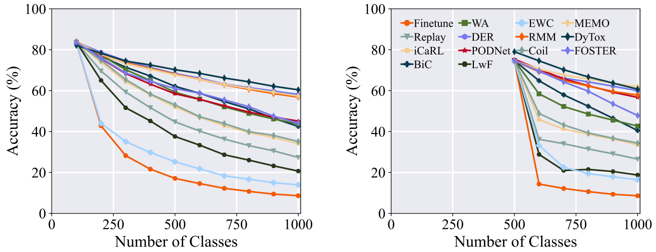
Table 2: Average and last accuracy performance comparison on CIFAR100. '#P' represents the number of parameters (million). The full table is reported in the supplementary.

Method	Base0 Inc5			Base0 Inc10		
	#P	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	#P	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	0.46	17.59	4.83	0.46	26.25	9.09
EWC	0.46	18.42	5.58	0.46	29.73	12.44
LwF	0.46	30.93	12.60	0.46	43.56	23.25
GEM	0.46	31.73	19.48	0.46	40.18	23.03
Replay	0.46	58.20	38.69	0.46	59.31	41.01
RMM	0.46	65.72	51.10	0.46	68.54	56.64
iCaRL	0.46	63.51	45.12	0.46	64.42	49.52
PODNet	0.46	47.88	27.99	0.46	55.22	36.78
Coil	0.46	57.68	34.33	0.46	60.27	39.85
WA	0.46	64.65	48.46	0.46	67.09	52.30
BiC	0.46	62.38	43.08	0.46	65.08	50.79
FOSTER	0.46	63.38	49.42	0.46	66.49	53.21
DER	9.27	67.99	53.95	4.60	69.74	**58.59**
MEMO	7.14	**68.10**	**54.23**	3.62	**70.20**	58.49
DyTox	10.7	68.06	52.23	10.7	71.07	58.72
L2P	85.7	84.00	78.96	85.7	89.35	83.39

Both of these settings are widely adopted in the current CIL community [87], [93]. Hence, to unify these settings, we use 'Base-m, Inc-n' to denote these settings where m stands for the number of classes in the first stage, and n stands for the number of classes in each incremental task. $m = 0$ stands for the 'Train from scratch' protocol. We use the *same* training splits for every compared method for a fair comparison. The testing set is the same as the original one for holistic evaluation.

4.1.2 Evaluation Metrics

There are several metrics to evaluate the CIL model. We denote the Top-1 accuracy after the b-th task as $\mathcal{A}_b$, and higher $\mathcal{A}_b$ indicates a better prediction accuracy. Since the CIL model is continually updated, the accuracy often decays with more tasks incorporated. Hence, the accuracy after the last stage ($\mathcal{A}_B$) is a proper metric to measure the overall accuracy among all classes.



(a) ImageNet1000 Base0 Inc100 (b) ImageNet1000 Base500 Inc100

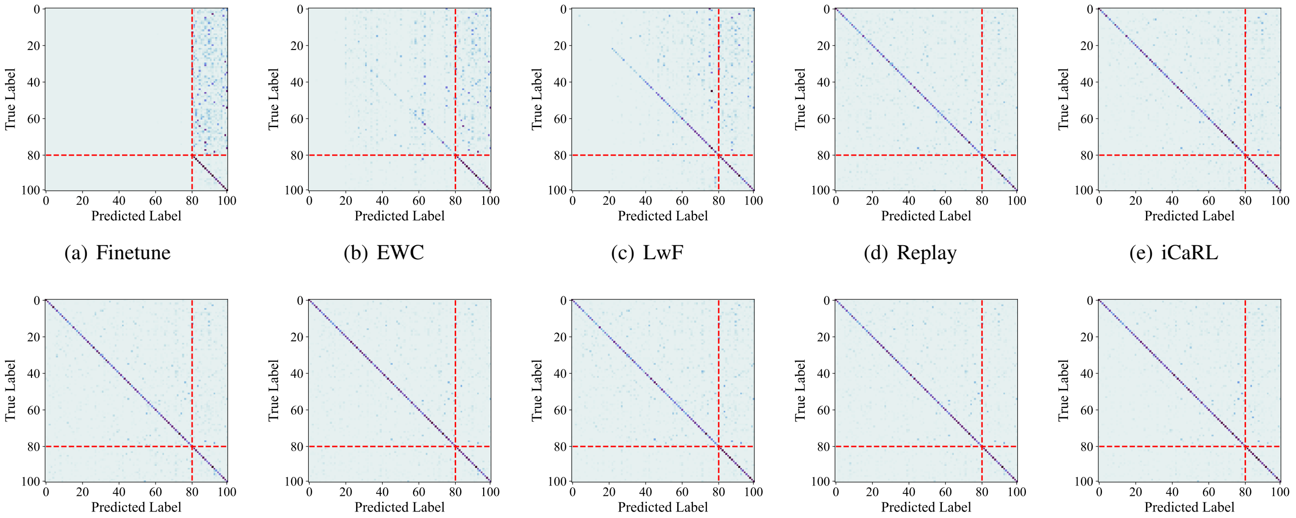
Figure 8: Incremental accuracy of different methods on ImageNet1000. Legends are shown in (b).

On the other hand, only comparing the final accuracy ignores the performance evolution along the learning trajectory. Hence, another metric denoted as 'average accuracy' considers the performance of all incremental stages: $\bar{\mathcal{A}} = \frac{1}{B}\sum_{b=1}^{B}\mathcal{A}_b$. A higher average accuracy denotes a stronger performance along the incremental stages. Apart from Top-1 accuracy, some works address Top-5 accuracy for the ImageNet dataset. In the supplementary, we also report the Top-5 accuracy of these comparisons.

4.1.3 Implementation Details

Selected methods: In the comparison, we aim to contain all kinds of methods listed in the taxonomy of Table 1. We systematically choose 16 methods, including: Replay [196], RMM [197] (*data replay*), GEM [59] (*data regularization*), EWC [77] (*parameter regularization*), FOSTER [21], MEMO [22], DER [20], DyTox [23], L2P [72] (*dynamic networks*), LwF [85], iCaRL [32], PODNET [98], Coil [92] (*knowledge distillation*), WA [115], BiC [87] (*model rectify*). We also report the baseline method 'finetune,' which updates the model with Eq. 4.

Among these methods, RMM is a specific technique to efficiently organize the memory budget, which can be orthogonally combined with other methods, and we combine it with FOSTER, denoted as FOSTER+RMM. L2P requires pre-trained ViT as the backbone model, while others are trained from scratch.



(a) Finetune (b) EWC (c) LwF (d) Replay (e) iCaRL

(f) BiC (g) DER (h) FOSTER (i) MEMO (j) Dytox

Figure 9: Confusion matrix of different methods on CIFAR100 Base0 Inc20 setting after the last incremental stage. The dotted red line splits the last 20 classes from the 80 classes learned in former tasks.

Training details: We implement the above methods with PyTorch [198] and PyCIL [199]. Specifically, we use the *same* network backbone for *all* CNN-based compared methods, *i.e.*, ResNet32 [200] for CIFAR100 and ResNet18 for ImageNet. We use SGD with an initial learning rate of 0.1 and momentum of 0.9. The training epoch is set to 170 for all datasets with a batch size of 128. The learning rate suffers a decay of 0.1 at 80 and 120 epochs. For ViT-based methods like DyTox and L2P, we follow the original implementation and use ConViT [201] for DyTox and pre-trained ViT-B/16 [24] for L2P. The optimization parameters of them are set according to the original paper since ViT has a different optimization preference to CNN. We follow the original paper to set the algorithm-specific parameters, *e.g.*, splitting 10% exemplars from the exemplar set as validation for BiC, setting the temperature τ to 5 and using a 10 epochs warm-up for DER, using ℓ_2 norm to normalize the fully-connected layers in WA.

Research has shown that a good starting point (*i.e.*, the first stage accuracy $\mathcal{A}_1$) implies better transferability and will suffer less forgetting [112]. It must be noted that the performance of the first stage should be *eliminated* in order not to affect the forgetting evaluation. Hence, to make the methods share the same starting point, we utilize the same training strategy, data augmentation, and hyperparameters. We use basic data augmentation, *e.g.*, random crop, horizontal flip, and color jitter for CIFAR100 and ImageNet.

It must be noted that Finetune, EWC, LwF, and L2P are exemplar-free methods, and we do not use any exemplar set for them. For other methods, we follow the benchmark setting to set the number of exemplars to 2,000 for CIFAR100 and ImageNet100 and 20,000 for ImageNet1000. These exemplars are equally sampled from each seen class via the herding [118] algorithm, as introduced in Definition 2.

4.2 Comparison on Small-Scale Datasets

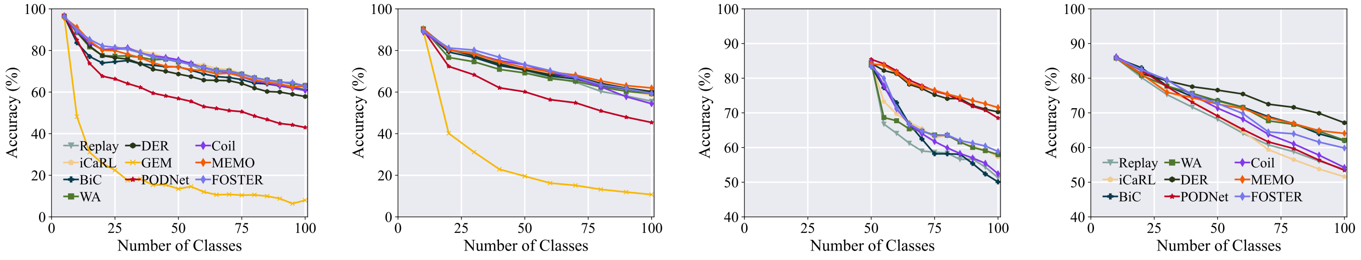
In this section, we evaluate different methods on the small-scale dataset, *i.e.*, CIFAR100. We compare these methods under both the TFS and TFH setting and design four data splits (two more splits are reported in supplementary). We report the incremental performance of different methods in Figure 6 and summarize the average and final performance, the number of parameters in Table 2.

As we can infer from these figures, finetune shows the worst performance among all settings, verifying the fact that the model will suffer forgetting when sequentially learning new concepts. Regularizing the parameters, *i.e.*, EWC, shows a negligible improvement over finetune. By contrast, LwF adds knowledge distillation loss to resist forgetting, which substantially improves the performance. When the exemplar set is available, directly replaying them during model updating can further enhance the performance by a substantial margin. Coil and iCaRL combine the exemplar repay and knowledge distillation and further obtain a performance boost than the vanilla replay. Model rectify methods, *i.e.*, BiC and WA, rectify the bias in iCaRL and further improve the accuracy. However, recent methods based on dynamic networks (*i.e.*, DER, FOSTER, MEMO, and DyTox) show competitive results with the help of multiple backbones. It indicates that saving more backbones can substantially help the model overcome forgetting. When it comes to pre-trained models, L2P obtains the best performance among all methods. However, other methods are trained from scratch, while L2P relies on the ViT pre-trained on ImageNet-21K, making it unfair to directly compare these two lines of methods.

On the other hand, we can infer from different settings that TFH requires more *stability* than TFS. Since the evaluation is based on the accuracy among all classes, remembering old classes becomes more critical when there is a large group of base classes.

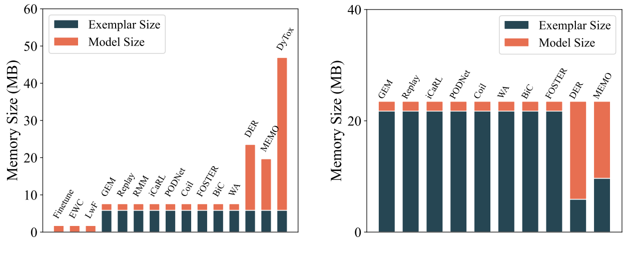
4.3 Comparison on Large-Scale Datasets

In this section, we evaluate different methods on the large-scale dataset, *i.e.*, ImageNet100 and ImageNet1000. We compare these methods under the TFS and TFH settings and design four data splits for ImageNet100 (two more splits are reported in supplementary) and two for ImageNet1000. We report the incremental performance (top-1 accuracy) of different methods in Figure 7 and Figure 8. We report the top-5 accuracy and summarize the average and final performance, the number of parameters in the supplementary. Note that GEM requires saving a large-scale matrix for solving the QP problem, which cannot be conducted with the ImageNet dataset. On the other hand, since the ViT in L2P is pre-trained on ImageNet-21K, incrementally training it on ImageNet is meaningless, and we do not report its results.



(a) CIFAR100 Base0 Inc5 (b) CIFAR100 Base0 Inc10 (c) ImageNet100 Base50 Inc5 (d) ImageNet100 Base0 Inc10

Figure 10: Incremental accuracy of different methods with aligned memory cost. Legends are shown in (a) and (d).



(a) Memory size of Figure 6(b) (b) Memory size of Figure 10(b)

Figure 11: Memory size of different comparison protocols. Dark bars denote the budget for saving exemplars, and red bars represent the budget for keeping the model in memory. Different methods should be aligned to the same budget for a fair comparison, as shown in (b).

Table 3: Average and last accuracy performance comparison on CIFAR100 with aligned memory cost. '#P' represents the number of parameters (million). '#$\mathcal{E}$' denotes the number of exemplars, and 'MS' denotes the memory size (MB). The full table is reported in the supplementary.

Method	CIFAR100 Base0 Inc10				
	#P	#$\mathcal{E}$	MS	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
GEM	0.46	7431	23.5	27.03	10.72
Replay	0.46	7431	23.5	69.97	55.61
iCaRL	0.46	7431	23.5	70.94	58.52
PODNet	0.46	7431	23.5	60.80	45.38
Coil	0.46	7431	23.5	70.69	54.40
WA	0.46	7431	23.5	69.55	59.26
BiC	0.46	7431	23.5	70.69	59.60
FOSTER	0.46	7431	23.5	72.28	59.39
DER	4.60	2000	23.5	71.47	60.26
MEMO	3.62	3312	23.5	**72.37**	**61.98**

As we can infer from these figures, most methods share the same trend as CIFAR100. Replay acts as a strong baseline in both small-scale and large-scale inputs, verifying the effectiveness of exemplars in incremental learning. Dynamic networks consistently show the best performance among all settings, outperforming other methods by a substantial margin in the benchmark comparison.

4.4 Visualization of Confusion Matrix

Incremental models are biased toward the latest task, triggering the development of model rectify-based methods. In this section, we choose a typical phenomenon to show the model bias and how the CIL models rectify them, *i.e.*, confusion matrix. We also visualize the weight norms of different methods in the supplementary.

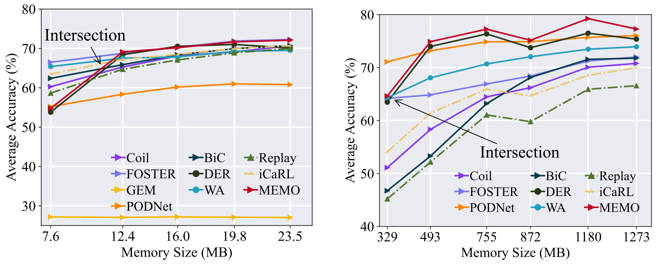
Specifically, we show the confusion matrix of 10 typical methods in Figure 9 and report the full results in supplementary. The matrix is drawn under CIFAR100 Base0 Inc20 setting after the last incremental task. We can infer that finetune and EWC forget most knowledge from former classes, while LwF shows less forgetting with knowledge distillation. When equipping the model with exemplars, the diagonal elements become brighter, implying that forgetting is alleviated. Dynamic networks still show the most competitive performance in these visualizations, indicating that the former backbones help the model retrieve old features and resist catastrophic forgetting in the incremental learning process.

4.5 Memory-Aligned Comparison

As we can infer from former experimental evaluations, dynamic networks show the best performance among all methods. However, are these methods *fairly* compared? We argue that these dynamic networks implicitly introduce an extra memory budget, namely *model buffer* for keeping old models. The additional buffer results in an unfair comparison to those methods without storing models. Taking Table 2 for an example, the number of parameters in DER is ten times that of iCaRL in the CIFAR100 Base0 Inc10 setting. We visualize the memory cost of different methods in Figure 11(a) and find that dynamic networks obtain better performance at the expense of more memory budgets. It makes directly comparing different kinds of methods *unfair*.

In this section, we follow [22] and aim for a fair comparison among different methods with different memory budgets. For those methods with different memory costs, we need to *align* the performance measure at the same memory scale for a fair comparison. For example, a ResNet32 model costs $463,504$ parameters (float), while a CIFAR image requires $3 \times 32 \times 32$ integer numbers (int). Hence, the budget for saving a backbone is equal to saving $463,504$ floats $\times 4$ bytes/float $\div (3 \times 32 \times 32)$ bytes/image $\approx$ 603 instances for CIFAR. A fair comparison between different methods can be made by equipping the other methods with more exemplars. Since DER requires saving all the backbones from history, we align the memory cost of other methods to DER with extra exemplars. We visualize the memory budget of different methods under the current protocol in Figure 11(b), where the total budgets of different methods are aligned to the same scale.

We report the results under the current protocol in Figure 10 and the detailed performance in Table 3. Since Finetune, LwF, and EWC cannot be combined with the exemplar set, we do not report the results of these methods. Similarly, DyTox and L2P utilize different kinds of backbones, and we do not report their results. As we can infer from these results, the gap between dynamic networks and other methods is no longer large under fair comparison. For example, DER outperforms iCaRL by 9.07 in terms of the final accuracy in the benchmark comparison of CIFAR100 Base0 Inc10, while the gap decreases to 1.74 under the current protocol.



(a) CIFAR100 Base0 Inc10 (b) ImageNet100 Base50 Inc5

Figure 12: Average performance-memory curve of different methods with different datasets. Dynamic networks perform better with large budgets, while other methods dominate small ones.

4.6 Memory-Agnostic Measure

Section 4.5 enables fair comparison among different methods by aligning the memory cost. However, the comparison is made by aligning the memory cost of other methods to DER. In contrast, open-world applications require conducting class-incremental learning in various scenarios, *i.e.*, high-performance computers and edge devices are both essential. Hence, it requires a *memory-agnostic measure* for CIL to measure the model's extendability given *any* memory budget.

To this end, we can set several 'comparison budgets' and align the memory cost of different methods to them. The budget list starts from a small value and incrementally enlarges, containing the requirement of different scale models. In this setting, we set the budget of the start point to a single backbone and gradually increase it to the budget cost of DER. For those algorithms without extra model storage, we equip more exemplars to meet the selected budget. However, when the selected budget is smaller than the required size, dynamic networks (*i.e.*, DER and MEMO) cannot be deployed with the benchmark backbone. Therefore, we choose smaller backbones with fewer parameters for alignment. Hence, we can measure the performance of different methods at different memory scales and draw the performance-memory curve, as shown in Figure 12. The X-coordinate corresponds to the memory cost, and the Y-coordinate indicates the average performance $\bar{\mathcal{A}}$ (or last performance $\mathcal{A}_B$). We suggest the area under the performance-memory curve (AUC) since the curve of each method indicates the dynamic ability with the change of model size. We calculate **AUC-A** and **AUC-L**, standing for the AUC under the average performance-memory and last performance-memory curves. The results are reported in Table 4.

As we can infer from these results, there are two main conclusions. Firstly, there exists an *intersection* between dynamic networks and other methods, where other methods perform better given a small memory size, while dynamic networks perform better given a large memory size. In other words, *there is no free lunch in CIL*, and different kinds of methods have their dominant domains. Secondly, the AUC measure provides a holistic way to measure the *extendability* of different methods given various memory budgets. We find that FOSTER and MEMO show competitive results regarding the AUC-A/L measure, implying their stronger extendability. It would be interesting to introduce it as the performance measure and design CIL methods for real-world applications. We report implementation details and the performance of each budget point in the supplementary.

Table 4: Memory-agnostic performance measures for CIL. AUC depicts the dynamic ability with the change of memory size.

Method	CIFAR100 Base0 Inc10		ImageNet100 Base50 Inc5	
	AUC-A	AUC-L	AUC-A	AUC-L
GEM	4.31	1.70	-	-
Replay	10.49	8.02	553.6	470.1
iCaRL	10.81	8.64	607.1	527.5
PODNet	9.42	6.80	701.8	624.9
Coil	10.60	7.82	601.9	486.5
WA	10.80	8.92	666.0	581.7
BiC	10.73	8.30	592.7	474.2
FOSTER	**11.12**	**9.03**	638.7	566.3
DER	10.74	8.95	699.0	639.1
MEMO	10.85	**9.03**	**713.0**	**654.6**

4.7 Discussions about the Comparison

With the above experimental evaluations from three aspects, we have the following conclusions:

- Equipping the model with exemplars is a simple and effective way to resist forgetting in CIL models.
- Knowledge distillation performs better than parameter regularization in resisting forgetting with the same cost.
- Model rectification can further boost the performance of other CIL models in a plug-and-play manner.
- Pre-trained models can ease the burden of incremental learning and show very strong performance. However, since the features of pre-trained models are already available and do not need to be incrementally learned, comparing pre-trained models to other methods may be unfair.
- Dynamic networks show the best performance in the evaluation at the cost of extra memory budgets. However, when changing the extra budgets into equal size of exemplars, the performance gap becomes smaller.
- AUC-A and AUC-L provide a way to evaluate different CIL methods in a *memory-agnostic* manner, which can help select the method with *extendability* given any budget.

5 Future Directions

In this section, we discuss the possible future directions of the class-incremental learning field.

CIL with Complex Inputs: In the real world, data is often with complex format [202], [203], *e.g.*, few-shot [204], imbalanced [136], with noisy label [205], weak supervised [206], multi modal [207], concept drift [13], novel classes [208], [209] etc. Incremental learning methods should be able to handle these real-world scenarios for better generalizability. Figure 13 depicts possible future applications of class-incremental learning.

Specifically, [104], [106], [210], [211] address the few-shot class-incremental learning problem, where the model is required to be adapted to incoming few-shot classes. [212], [213], [214] propose to tackle the long-tailed class-incremental learning problem, where the head classes are easy to collect with adequate instances while tail classes are scarce. Recently, with the prosperity of CLIP [147], incrementally training the language vision models to handle multi-modal data streams [188], [189] is becoming popular. Since the labeling cost is always expensive in the real world, there are some works addressing training CIL models in a semi-supervised [215] or unsupervised manner [171]. [33] proposes a unified framework to handle CIL with concept drift. If the test dataset contains instances from unknown classes, open-set recognition [208] and novel class discovery [216], [217] can equip

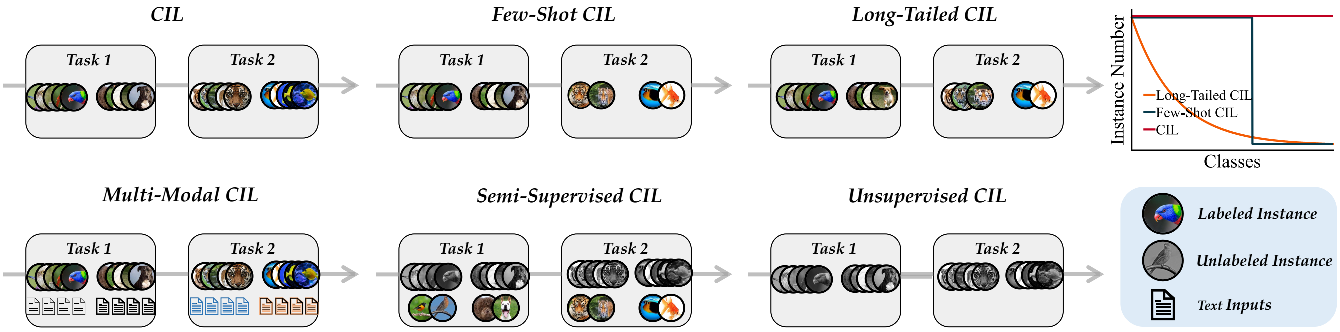


Figure 13: Future directions of class-incremental learning. Class-incremental learning often emerges with other open-world problems, *e.g.*, few-shot, long-tailed, multi-modal, weak supervision.

the model with the detection ability, which refers to the open-world recognition problem [218]. Besides, test-time adaptation [219] enables the model to handle these drifts on-the-fly. Lastly, real-world data may emerge hierarchically, and the labels could evolve from coarse-grained to fine-grained [220]. Learning incremental models with refined concepts [221], [222], [223], [224], [225] is also vital to building real-world learning systems.

CIL with General Data Stream: Current CIL methods have a set of restrictions on the data stream, *e.g.*, saving exemplars for rehearsal, undergoing multi-epoch offline training within an incremental task, etc. Future class-incremental learning algorithms should be able to conduct fully online training [226], [227], [228], [229], [230] without requiring the task boundaries [231], [232], [233]. On the other hand, saving exemplars from the history may violate user privacy in some cases. To this end, exemplar-free CIL [57], [58], [138], [139] should be conducted to enable the model to be adapted without the help of exemplars. Lastly, most CIL methods rely on the number of base classes to define hyper-parameters in model optimization, where more base classes require larger stability and fewer base classes require larger plasticity. Therefore, designing the algorithm to handle CIL problems given any base classes [234] is also important to the real world.

CIL with Any Memory Budgets: Dynamic networks have obtained impressive performance in recent years [20], [21]. However, most of them require an extra memory budget to save the external backbones. In the real world, a good CIL algorithm should handle different budget restrictions. For example, an algorithm should be able to learn with high-performance computers or with edge devices (e.g., smartphones), and both scenarios are essential. Hence, deploying and comparing CIL models in the real world should take the memory budgets into consideration. The performance measures of AUC-A/L [22] are proper solutions to holistically compare different methods given any memory budgets. On the other hand, future CIL methods are encouraged to handle specific learning scenarios. For example, [235] addresses training CIL systems under resource-limited scenarios.

CIL with Pre-trained Models: Recently, pre-trained models have shown to work competitively with their strong transferability, especially for ViT-based methods [72], [73], [74], [75]. The pre-trained models provide generalizable features for the downstream tasks, enabling the incremental model to adjust with minimal cost [145]. CIL with pre-trained models is a proper way to handle real-world incremental applications from an excellent starting point. However, since the final target for incremental learning is to build a generalizable feature continually [236], some will argue that pre-trained models weaken the difficulty of incremental learning. From this perspective, designing proper algorithms to train the CIL model from scratch is more challenging. On the other hand, pre-trained language-vision models [147], [237] have shown powerful generalizability in recent years, and exploring the ensemble of various pre-trained models [238], [239] is also an exciting topic.

CIL with Bidirectional Compatibility: Compatibility is a design characteristic in software engineering community [240], [241], [242], [243], which was introduced to the machine learning community in [244], [245]. Backward compatibility allows for interoperability with an older legacy system. In contrast, forward compatibility allows a system to accept input intended for a later version of itself. In the incremental learning field, compatibility is also a core problem, where the new model is required to understand the features produced by the old model to classify old classes. Most methods aim to enhance backward compatibility by making the new model similar to the old one. However, FACT [106] addresses the forward compatibility in CIL, where the model should be prepared for future updates, acting like the pre-assigned interface. Furthermore, it will be interesting to address bidirectional compatibility [92] in the future.

Analyzing the Reason Behind Catastrophic Forgetting: Model rectification-based methods aim to reduce the inductive bias in the CIL model. Recently, more works have tried to analyze the reason for forgetting in CIL. [172], [246] find that batch normalization layers are biased when sequentially trained, which triggers the different activation of old and new classes. [138] finds that representations with large eigenvalues transfer better and suffer less forgetting, and [226] finds that eigenvalues can be enlarged by maximizing the mutual information between old and new features. [247] theoretically decomposes the CIL problem into within-task and task-id predictions. It further proves that good within-task and task-id predictions are necessary and sufficient for good CIL performances. [248] finds that applying data replay causes the newly added classes' representations to overlap significantly with the previous classes, leading to highly disruptive parameter updates. [249] empirically finds that the amount of forgetting correlates with the geometrical properties of the convergent points. It would be interesting to explore more reasons for catastrophic forgetting theoretically and empirically in the future.

6 Conclusion

Real-world applications are often faced with streaming data, with which the model should be incrementally updated without catastrophic forgetting. In this paper, we provide a comprehensive survey about deep class-incremental learning by categorizing them into three categories and six subcategories taxonomically and chronologically. Additionally, we provide a holistic comparison among different methods on several publicly available datasets.

With these results, we discuss the insights and summarize the common rules to inspire future research. Finally, we highlight an important factor in CIL comparison, namely memory budget, and advocate evaluating different methods holistically by emphasizing the effect of memory budgets. We provide a comprehensive evaluation of different methods given specific budgets as well as some new performance measures. We expect this survey to provide an effective way to understand current state-of-the-art and speed up the development of the class-incremental learning field.

REFERENCES

[1] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, 2020.

[2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[3] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[5] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *ACL*, pages 4885–4901, 2020.

[6] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *CSUR*, 50(2):1–36, 2017.

[7] Lukasz Golab and M Tamer Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[8] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, et al. Open challenges for data stream mining research. *KDD*, 16(1):1–10, 2014.

[9] Mohamed Medhat Gaber. Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):79–85, 2012.

[10] Mahawaga Arachchige Pathum Chamikara, Peter Bertók, Dongxi Liu, Seyit Camtepe, and Ibrahim Khalil. Efficient data perturbation for privacy preserving and accurate data stream mining. *Pervasive and Mobile Computing*, 48:1–19, 2018.

[11] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[12] Stephen T Grossberg. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media, 2012.

[13] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[14] Joao Gama, Indre Zliobaite, Albert Bifet, Mykol Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys*, 46(4):1–37, 2014.

[15] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

[16] Stephan Lewandowsky and Shu-Chen Li. Catastrophic interference in neural networks: Causes, solutions, and data. In *Interference and inhibition in cognition*, pages 329–361. Elsevier, 1995.

[17] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From n to n+ 1: Multiclass transfer incremental learning. In *CVPR*, pages 3358–3365, 2013.

[18] Qing Da, Yang Yu, and Zhi-Hua Zhou. Learning with augmented class by exploiting unlabeled data. In *AAAI*, pages 1760–1766, 2014.

[19] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[20] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021.

[21] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. *arXiv preprint arXiv:2204.04662*, 2022.

[22] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023.

[23] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pages 9285–9295, 2022.

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[25] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[26] Magdalena Marta Biesialska, Katarzyna Biesialska, and Marta Ruiz Costa-Jussà. Continual lifelong learning in natural language processing: a survey. In *COLING*, pages 6523–6541, 2020.

[27] Tyler L Hayes, Giri P Krishnan, Maxim Bazhenov, Hava T Siegelmann, Terrence J Sejnowski, and Christopher Kanan. Replay in deep learning: Current approaches and missing biological elements. *Neural Computation*, 33(11):2908–2950, 2021.

[28] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, pages 1–13, 2022.

[29] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[30] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

[32] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017.

[33] Jiangwei Xie, Shipeng Yan, and Xuming He. General incremental learning with domain-aware categorical representations. In *CVPR*, pages 14351–14360, 2022.

[34] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *arXiv preprint arXiv:2110.10031*, 2021.

[35] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, pages 8218–8227, 2021.

[36] Jihwan Bang, Hyunseo Koh, Seulki Park, Hwanjun Song, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on a contaminated data stream with blurry task boundaries. In *CVPR*, pages 9275–9284, 2022.

[37] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *CVPR*, pages 3589–3599, 2021.

[38] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *AAAI*, pages 9630–9638, 2021.

[39] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pages 532–547, 2018.

[40] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, pages 11816–11825, 2019.

[41] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *AAAI*, pages 3302–3309, 2018.

[42] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. In *NeurIPS*, pages 350–360, 2019.

[43] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI*, pages 6993–7001, 2020.

[44] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *ICCV*, pages 8250–8259, 2021.

[45] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4080–4090, 2017.

[46] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *ECCV*, pages 699–715, 2020.

[47] Hanbin Zhao, Hui Wang, Yongjian Fu, Fei Wu, and Xi Li. Memory efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[48] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *NIPS*, 30, 2017.

[49] Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *BMVC*, page 98, 2018.

[50] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*, 2019.

[51] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *ICLR*, 2018.

[52] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, pages 11321–11329, 2019.

[53] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *ICCV*, pages 6619–6628, 2019.

[54] Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *CVPR*, pages 5383–5392, 2021.

[55] Jian Jiang, Edoardo Cetin, and Oya Celiktutan. Ib-drr-incremental learning with information-back discrete representation replay. In *CVPRW*, pages 3533–3542, 2021.

[56] Wenju Sun, Qingyong Li, Jing Zhang, Danyu Wang, Wen Wang, and Yangli-ao Geng. Exemplar-free class incremental learning via discriminative and comparable one-class classifiers. *arXiv preprint arXiv:2201.01488*, 2022.

[57] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880, 2021.

[58] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. *arXiv preprint arXiv:2211.13131*, 2022.

[59] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, pages 6467–6476, 2017.

[60] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2018.

[61] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *CVPR*, pages 184–193, 2021.

[62] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[63] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *CVPR*, pages 9634–9643, 2021.

[64] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018.

[65] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *NeurIPS*, pages 899–908, 2018.

[66] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, pages 3925–3934, 2019.

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

[68] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, pages 3366–3375, 2017.

[69] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, pages 4528–4537, 2018.

[70] Hanbin Zhao, Yongjian Fu, Mintong Kang, Qi Tian, Fei Wu, and Xi Li. Mgsvf: Multi-grained slow vs. fast framework for few-shot class-incremental learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[71] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *NeurIPS*, 34:16131–16144, 2021.

[72] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pages 139–149, 2022.

[73] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2204.04799*, 2022.

[74] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. *arXiv e-prints*, pages arXiv–2211, 2022.

[75] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *arXiv preprint arXiv:2207.12819*, 2022.

[76] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. *arXiv preprint arXiv:2212.04842*, 2022.

[77] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

[78] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017.

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

[80] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, pages 11254–11263, 2019.

[81] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*, pages 4652–4662, 2017.

[82] Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, and Yuan Jiang. Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In *KDD*, pages 74–82, 2019.

[83] Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, Yuan Jiang, and Yang Jian. Cost-effective incremental deep model: Matching model capacity with the least sampling. *IEEE Transactions on Knowledge and Data Engineering*, page in press, 2021.

[84] Janghyeon Lee, Hyeong Gwon Hong, Donggyu Joo, and Junmo Kim. Continual learning with extended kronecker-factored approximate curvature. In *CVPR*, pages 9001–9010, 2020.

[85] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, pages 614–629. Springer, 2016.

[86] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017.

[87] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019.

[88] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, pages 437–452, 2018.

[89] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *ICCV*, pages 312–321, 2019.

[90] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *WACV*, pages 1131–1140, 2020.

[91] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *ICCV*, pages 9374–9384, 2021.

[92] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Co-transport for class-incremental learning. In *ACM MM*, pages 1645–1654, 2021.

[93] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019.

[94] Yichen Lu, Mei Wang, and Weihong Deng. Augmented geometric distillation for data-free incremental person reid. In *CVPR*, pages 7329–7338, 2022.

[95] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *ICCV*, pages 13698–13707, 2021.

[96] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *CVPR*, pages 5138–5146, 2019.

[97] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*, pages 16071–16080, 2022.

[98] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102, 2020.

[99] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. Distilling causal effect of data in class-incremental learning. In *CVPR*, pages 3957–3966, 2021.

[100] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *CVPR*, pages 1591–1600, 2021.

[101] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-DFCIL: relation-guided representation learning for data-free class incremental learning. In *ECCV*, pages 423–439, 2022.

[102] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *AAAI*, pages 1255–1263, 2021.

[103] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *ECCV*, pages 254–270. Springer, 2020.

[104] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *CVPR*, pages 12183–12192, 2020.

[105] Yu Liu, Xiaopeng Hong, Xiaoyu Tao, Songlin Dong, Jingang Shi, and Yihong Gong. Model behavior preserving for class-incremental learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[106] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *CVPR*, pages 9046–9056, 2022.

[107] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, pages 6982–6991, 2020.

[108] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip HS Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *CVPR*, pages 16722–16731, 2022.

[109] Shibo Jie, Zhi-Hong Deng, and Ziheng Li. Alleviating representational shift for continual fine-tuning. In *CVPRW*, pages 3810–3819, 2022.

[110] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. In *NeurIPS*, pages 15579–15590, 2020.

[111] Pravendra Singh, Pratik Mazumder, Piyush Rai, and Vinay P Namboodiri. Rectification-based knowledge retention for continual learning. In *CVPR*, pages 15282–15291, 2021.

[112] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, pages 233–248, 2018.

[113] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *ICCV*, pages 583–592, 2019.

[114] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *CVPR*, pages 2544–2553, 2021.

[115] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020.

[116] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *ICCV*, pages 844–853, 2021.

[117] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo. Class-incremental learning with pre-allocated fixed classifiers. In *ICPR*, pages 6259–6266, 2021.

[118] Max Welling. Herding dynamical weights to learn. In *ICML*, pages 1121–1128, 2009.

[119] Anthony Robins. Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pages 65–68. IEEE, 1993.

[120] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. *NIPS*, 23, 2010.

[121] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 27, 2014.

[122] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[123] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pages 2256–2265. PMLR, 2015.

[124] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[125] Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *CVPR*, pages 4016–4025, 2019.

[126] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: learning to generate images from new categories without forgetting. In *NeurIPS*, pages 5966–5976, 2018.

[127] Mengyao Zhai, Lei Chen, and Greg Mori. Hyper-lifelonggan: Scalable lifelong learning for image conditioned generation. In *CVPR*, pages 2246–2255, 2021.

[128] Vinay Kumar Verma, Kevin J Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. In *CVPR*, pages 13865–13875, 2021.

[129] Lukasz Korycki and Bartosz Krawczyk. Class-incremental experience replay for continual learning under concept drift. In *CVPR*, pages 3649–3658, 2021.

[130] Andrea Maracani, Umberto Michieli, Marco Toldo, and Pietro Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. In *ICCV*, pages 7026–7035, 2021.

[131] Sungmin Cha Kim, Youngjoon Yoo, Taesup Moon, et al. Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. *arXiv preprint arXiv:2106.11562*, 2021.

[132] Sungmin Cha, YoungJoon Yoo, Taesup Moon, et al. Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. *NeurIPS*, 34:10919–10930, 2021.

[133] Andrés Villa, Kumail Alhamoud, Victor Escorcia, Fabian Caba, Juan León Alcázar, and Bernard Ghanem. vclimb: A novel video class incremental learning benchmark. In *CVPR*, pages 19035–19044, 2022.

[134] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *ICCV*, pages 9385–9394, 2021.

[135] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021.

[136] Lu Yang, He Jiang, Qing Song, and Jun Guo. A survey on long-tailed visual recognition. *International Journal of Computer Vision*, pages 1–36, 2022.

[137] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning. In *CVPR*, pages 10164–10173, 2022.

[138] Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *NeurIPS*, 34, 2021.

[139] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *CVPR*, pages 9296–9305, 2022.

[140] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[141] Zijie J Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Polo Chau. Cnn explainer: learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406, 2020.

[142] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[143] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[144] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[145] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727. Springer, 2022.

[146] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[147] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021.

[148] Lucien Le Cam. *Asymptotic methods in statistical decision theory*. Springer Science & Business Media, 2012.

[149] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *NeurIPS*, 32:13669–13679, 2019.

[150] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4548–4557. PMLR, 2018.

[151] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. In *NeurIPS*, pages 12669–12679, 2019.

[152] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *CVPR*, pages 3931–3940, 2020.

[153] Oleksiy Ostapenko, Pau Rodriguez, Massimo Caccia, and Laurent Charlin. Continual learning via local module composition. *NeurIPS*, 34:30298–30312, 2021.

[154] Beyza Ermis, Giovanni Zappella, Martin Wistuba, Aditya Rawal, and Cedric Archambeau. Memory efficient continual learning with transformers. In *NeurIPS*, 2022.

[155] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *ACL*, pages 487–503, 2021.

[156] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *NIPS*, 30, 2017.

[157] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015.

[158] Yi-Kai Zhang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Audio-visual generalized few-shot learning with prototype-based co-adaptation. In *Interspeech*, pages 531–535, 2022.

[159] Zhi-Hua Zhou and Yuan Jiang. Nec4. 5: neural ensemble based c4. 5. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):770–773, 2004.

[160] Zhi-Hua Zhou, Yuan Jiang, and Shi-Fu Chen. Extracting symbolic rules from trained neural network ensembles. *Ai Communications*, 16(1):3–15, 2003.

[161] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 35(11):2624–2637, 2013.

[162] Hanbin Zhao, Xin Qin, Shihao Su, Yongjian Fu, Zibo Lin, and Xi Li. When video classification meets incremental classes. In *ACM MM*, pages 880–889, 2021.

[163] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, pages 3967–3976, 2019.

[164] Guanglei Yang, Enrico Fini, Dan Xu, Paolo Rota, Mingli Ding, Moin Nabi, Xavier Alameda-Pineda, and Elisa Ricci. Uncertainty-aware contrastive distillation for incremental semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[165] Youngmin Oh, Donghyeon Baek, and Bumsub Ham. Alife: Adaptive logit regularizer and feature replay for incremental semantic segmentation. In *NeurIPS*, 2022.

[166] Fabio Cermelli, Dario Fontanel, Antonio Tavera, Marco Ciccone, and Barbara Caputo. Incremental learning in semantic segmentation from image labels. In *CVPR*, pages 4371–4381, 2022.

[167] Chang-Bin Zhang, Jia-Wen Xiao, Xialei Liu, Ying-Cong Chen, and Ming-Ming Cheng. Representation compensation networks for continual semantic segmentation. In *CVPR*, pages 7053–7064, 2022.

[168] Nan Pu, Wei Chen, Yu Liu, Erwin M Bakker, and Michael S Lew. Lifelong person re-identification via adaptive knowledge accumulation. In *CVPR*, pages 7901–7910, 2021.

[169] Zhipeng Huang, Zhizheng Zhang, Cuiling Lan, Wenjun Zeng, Peng Chu, Quanzeng You, Jiang Wang, Zicheng Liu, and Zheng-jun Zha. Lifelong unsupervised domain adaptive person re-identification with coordinated anti-forgetting and adaptation. In *CVPR*, pages 14288–14297, 2022.

[170] Chen He, Ruiping Wang, and Xilin Chen. A tale of two cils: The connections between class incremental learning and class imbalanced learning, and beyond. In *CVPR Workshop*, pages 3559–3569, 2021.

[171] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *ICCV*, pages 9516–9525, 2021.

[172] Quang Pham, Chenghao Liu, and HOI Steven. Continual normalization: Rethinking batch normalization for online continual learning. In *ICLR*, 2022.

[173] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456. PMLR, 2015.

[174] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *CVPR*, pages 9233–9242, 2020.

[175] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. Incremental few-shot object detection. In *CVPR*, pages 13846–13855, 2020.

[176] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, pages 3400–3409, 2017.

[177] KJ Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Towards open world object detection. In *CVPR*, pages 5830–5840, 2021.

[178] Jianren Wang, Xin Wang, Yue Shang-Guan, and Abhinav Gupta. Wanderlust: Online continual object detection in the real world. In *ICCV*, pages 10829–10838, 2021.

[179] Na Dong, Yongqiang Zhang, Mingli Ding, and Gim Hee Lee. Bridging non co-occurrence with unlabeled in-the-wild data for incremental object detection. *NeurIPS*, 34:30492–30503, 2021.

[180] KJ Joseph, Jathushan Rajasegaran, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Incremental object detection via meta-learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9209–9216, 2021.

[181] Tao Feng, Mang Wang, and Hangjie Yuan. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *CVPR*, pages 9427–9436, 2022.

[182] Minh H Vu, Gabriella Norman, Tufve Nyholm, and Tommy Löfstedt. A data-adaptive loss function for incomplete data and incremental learning in semantic image segmentation. *IEEE Transactions on Medical Imaging*, 41(6):1320–1330, 2021.

[183] Mengya Xu, Mobarakol Islam, Chwee Ming Lim, and Hongliang Ren. Class-incremental domain adaptation with smoothing and calibration for surgical report generation. In *MICCAI*, pages 269–278. Springer, 2021.

[184] Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. Pretrained language model in continual learning: A comparative study. In *ICLR*, 2022.

[185] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, KIM Gyeonghun, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *ICLR*, 2022.

[186] Chengwei Qin and Shafiq Joty. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *ICLR*, 2022.

[187] Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *arXiv preprint arXiv:2206.00761*, 2022.

[188] Shipeng Yan, Lanqing Hong, Hang Xu, Jianhua Han, Tinne Tuytelaars, Zhenguo Li, and Xuming He. Generative negative text replay for continual vision-language pretraining. In *ECCV*, pages 22–38. Springer, 2022.

[189] Tejas Srinivasan, Ting-Yun Chang, Leticia Leonor Pinto Alva, Georgios Chochlakis, Mohammad Rostami, and Jesse Thomason. Climb: A continual learning benchmark for vision-and-language tasks. *arXiv preprint arXiv:2206.09059*, 2022.

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

[191] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2:18, 2010.

[192] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017.

[193] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In *CVPR*, pages 2575–2584, 2020.

[194] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[195] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7, 2015.

[196] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[197] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *NeurIPS*, 34, 2021.

[198] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein,

Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019.

[199] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: A python toolbox for class-incremental learning. *arXiv preprint arXiv:2112.12533*, 2021.

[200] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2015.

[201] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*, pages 2286–2296. PMLR, 2021.

[202] Zhi-Hua Zhou. Open-environment machine learning. *National Science Review*, 9(8):nwac123, 2022.

[203] Da-Wei Zhou, Yang Yang, and De-Chuan Zhan. Learning to classify with incremental new class. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[204] Han-Jia Ye, Xiang-Rong Sheng, and De-Chuan Zhan. Few-shot learning with adaptively initialized task optimizer: a practical meta-learning approach. *Machine Learning*, 109(3):643–664, 2020.

[205] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? *NeurIPS*, 32:6838–6849, 2019.

[206] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, pages 15750–15758, 2021.

[207] Feilong Chen, Fandong Meng, Xiuyi Chen, Peng Li, and Jie Zhou. Multimodal incremental transformer with visual grounding for visual dialogue generation. *arXiv preprint arXiv:2109.08478*, 2021.

[208] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Learning placeholders for open-set recognition. In *CVPR*, pages 4401–4410, 2021.

[209] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[210] Da-Wei Zhou, Han-Jia Ye, Liang Ma, Di Xie, Shiliang Pu, and De-Chuan Zhan. Few-shot class-incremental learning by sampling multi-phase tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[211] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *CVPR*, pages 12455–12464, 2021.

[212] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *ECCV*, pages 411–428. Springer, 2020.

[213] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *ICML*, pages 1952–1961. PMLR, 2020.

[214] Xialei Liu, Yu-Song Hu, Xu-Sheng Cao, Andrew D Bagdanov, Ke Li, and Ming-Ming Cheng. Long-tailed class incremental learning. In *ECCV*, pages 495–512. Springer, 2022.

[215] Yanchao Li, Yongli Wang, Qi Liu, Cheng Bi, Xiaohui Jiang, and Shurong Sun. Incremental semi-supervised learning on streaming data. *Pattern Recognition*, 88:383–396, 2019.

[216] Amanda Rios, Nilesh Ahuja, Ibrahima Ndiour, Utku Genc, Laurent Itti, and Omesh Tickoo. incdfm: Incremental deep feature modeling for continual novelty detection. In *ECCV*, pages 588–604. Springer, 2022.

[217] KJ Joseph, Sujoy Paul, Gaurav Aggarwal, Soma Biswas, Piyush Rai, Kai Han, and Vineeth N Balasubramanian. Novel class discovery without forgetting. In *ECCV*, pages 570–586. Springer, 2022.

[218] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *CVPR*, pages 1893–1902, 2015.

[219] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, pages 7201–7211, 2022.

[220] Zhiqiu Lin, Deepak Pathak, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Learning with an evolving class ontology. *arXiv preprint arXiv:2210.04993*, 2022.

[221] Mohamed Abdelsalam, Mojtaba Faramarzi, Shagun Sodhani, and Sarath Chandar. Iirc: Incremental implicitly-refined classification. In *CVPR*, pages 11038–11047, 2021.

[222] Xiang Xiang, Yuwen Tan, Qian Wan, Jing Ma, Alan Yuille, and Gregory D Hager. Coarse-to-fine incremental few-shot learning. In *ECCV*, pages 205–222. Springer, 2022.

[223] Kai Wang, Xialei Liu, Luis Herranz, and Joost van de Weijer. Hcv: Hierarchy-consistency verification for incremental implicitly-refined classification. *arXiv preprint arXiv:2110.11148*, 2021.

[224] Longhui Yu, Zhenyu Weng, Yuqing Wang, and Yuesheng Zhu. Multi-teacher knowledge distillation for incremental implicitly-refined classification. *arXiv preprint arXiv:2202.11384*, 2022.

[225] Jian Yang, Kai Zhu, Kecheng Zheng, and Yang Cao. Uncertainty-aware hierarchical refinement for incremental implicitly-refined classification. In *NeurIPS*, 2022.

[226] Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *ICML*, pages 8109–8126. PMLR, 2022.

[227] Fei Ye and Adrian G Bors. Task-free continual learning via online discrepancy distance learning. *arXiv preprint arXiv:2210.06579*, 2022.

[228] Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Yunzhe Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. *arXiv preprint arXiv:2209.13917*, 2022.

[229] Alexander Ororbia, Ankur Mali, Daniel Kifer, and C Lee Giles. Lifelong neural predictive coding: Learning cumulatively online without forgetting. *arXiv preprint arXiv:1905.10696*, 2019.

[230] Zhen Wang, Liu Liu, Yajing Kong, Jiaxian Guo, and Dacheng Tao. Online continual learning with contrastive vision transformer. In *ECCV*, pages 631–650. Springer, 2022.

[231] Julien Pourcel, Ngoc-Son Vu, and Robert M French. Online task-free continual learning with dynamic sparse distributed memory. In *ECCV*, pages 739–756. Springer, 2022.

[232] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *NeurIPS*, 34, 2021.

[233] Zhenyi Wang, Li Shen, Le Fang, Qiuling Suo, Tiehang Duan, and Mingchen Gao. Improving task-free continual learning by distributionally robust memory evolution. In *ICML*, pages 22985–22998. PMLR, 2022.

[234] Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Online hyperparameter optimization for class-incremental learning. In *AAAI*, 2023.

[235] Zifeng Wang, Zheng Zhan, Yifan Gong, Geng Yuan, Wei Niu, Tong Jian, Bin Ren, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. Sparcl: Sparse continual learning on the edge. *arXiv preprint arXiv:2209.09476*, 2022.

[236] Sungmin Cha, Dongsub Shim, Hyunwoo Kim, Moontae Lee, Honglak Lee, and Taesup Moon. Is continual learning truly learning representations continually? *arXiv preprint arXiv:2206.08101*, 2022.

[237] Vishal Thengane, Salman Khan, Munawar Hayat, and Fahad Khan. Clip model is an efficient continual learner. *arXiv preprint arXiv:2210.03114*, 2022.

[238] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. *arXiv preprint arXiv:2202.09368*, 2022.

[239] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[240] Ovidiu Gheorghioiu, Alexandru Salcianu, and Martin Rinard. Interprocedural compatibility analysis for static object preallocation. In *POPL*, pages 273–284, 2003.

[241] Santosh Nagarakatte, Jianzhou Zhao, Milo MK Martin, and Steve Zdancewic. Softbound: Highly compatible and complete spatial memory safety for c. In *PLDI*, pages 245–258, 2009.

[242] Pradeep Varma, Rudrapatna K Shyamasundar, and Harshit J Shah. Backward-compatible constant-time exception-protected memory. In *FSE/ESEC*, pages 71–80, 2009.

[243] Wei Xu, Daniel C DuVarney, and R Sekar. An efficient and backwards-compatible transformation to ensure memory safety of c programs. In *FSE*, pages 117–126, 2004.

[244] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. In *AAAI*, pages 2429–2437, 2019.

[245] Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. An empirical analysis of backward compatibility in machine learning systems. In *KDD*, pages 3272–3280, 2020.

[246] Minghao Zhou, Quanziang Wang, Jun Shu, Qian Zhao, and Deyu Meng. Diagnosing batch normalization in class incremental learning. *arXiv preprint arXiv:2202.08025*, 2022.

[247] Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. A theoretical study on solving continual learning. *arXiv preprint arXiv:2211.02633*, 2022.

[248] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *arXiv preprint arXiv:2203.03798*, 2022.

[249] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *NeurIPS*, 33:7308–7320, 2020.

Appendix A Supplied Results in Benchmark Comparison

In this section, we report the full results of the benchmark comparison. With a bit of redundancy, we list the incremental performance of all dataset splits. Specifically, we first report the detailed results of CIFAR100 and ImageNet100 and then report the top-5 accuracy of the ImageNet comparison.

A.1 Detailed Results of CIFAR100

We first report the results of CIFAR100 where we select six dataset splits (*i.e.*, Base0 Inc5, Base0 Inc10, Base0 Inc20, Base50 Inc50, Base50 Inc10, Base50 Inc25). The results are shown in Figure 14, and we list the number of parameters, the average accuracy, and the last accuracy in Table 5.

As we can infer from these figures, L2P consistently obtains the best performance among all settings. However, it must be noted that L2P relies on the ImageNet-21K pre-trained Vision Transformer with 85M parameters. At the same time, typical methods train ResNet32 with 0.46M parameters from scratch. Hence, comparing pre-trained model based methods to typical methods is unfair. We also notice that dynamic networks, *e.g.*, DER and MEMO obtain better performance at the cost of more parameters. Therefore, we report Table 5 to point out that parameter cost is sometimes ignored in the benchmark comparison, which shall lead to unfair results. These comparison details also motivate us to design *fair* comparison protocols for class-incremental learning.

A.2 Detailed Results of ImageNet100

We report the results of ImageNet100 where we select six dataset splits (*i.e.*, Base0 Inc5, Base0 Inc10, Base0 Inc20, Base50 Inc50, Base50 Inc10, Base50 Inc25). The results are shown in Figure 15, and we list the number of parameters, the average accuracy, and the last accuracy in Table 6. The conclusions are consistent with former small-scale datasets, and dynamic networks perform better at the cost of more memory budget.

It must be noted that there are two methods not included in the ImageNet comparison, *i.e.*, GEM and L2P. GEM requires solving the QP problem and saving a large-scale matrix, which cannot be deployed with ImageNet. Besides, L2P utilizes an ImageNet-21K pre-trained backbone for class-incremental learning [72], which overlaps with the dataset to be learned. Hence, these two methods are not included in the comparison.

A.3 ImageNet Top-5 Results

In this section, we report the top-5 accuracy of different methods in the ImageNet100 comparison. As shown in Figure 16, the ranking of different methods remains the same as in the top-1 accuracy comparison. We also report the top-5 accuracy in the ImageNet1000 comparison in Figure 17.

Appendix B Supplied Visualization Results

In this section, we report the full visualization results in the CIFAR100 comparison. Specifically, we first show the confusion matrix of 15 methods in Figure 18. As we can infer from these figures, finetune and EWC forget most knowledge from former classes, while LwF shows less forgetting with knowledge distillation. When comparing LwF to EWC, we can find that knowledge distillation is a more suitable solution to resist catastrophic forgetting than parameter regularization without saving exemplars. When equipping the model with exemplars, the diagonal elements become brighter, implying that forgetting is alleviated. The performance of Replay shows more steady improvement than finetune, indicating the critical role of exemplars in CIL. However, when comparing iCaRL to GEM, we can find that data regularization shows inferior performance than knowledge distillation in the efficiency of using exemplars. Moreover, BiC and WA improve the performance of iCaRL by rectifying the bias in the model, indicating that these methods can be orthogonally equipped to other CIL methods for rectification. Lastly, dynamic networks (*i.e.*, DER, FOSTER, MEMO, and DyTox) still show the most competitive performance in these visualizations, indicating that the former backbones help the model retrieve old features and resist forgetting.

Additionally, we visualize the weight norm of the fully-connected layers in Figure 19 to show the bias introduced by incremental learning. As we can infer from these figures, when sequentially finetuning the model, the weight norm of new classes will dominate the learning process due to data imbalance. As shown in Figure 19(a), the weight norm of finetune has a stair-step shape. Since the features are passed through the ReLU layer to avoid negative ones, the larger weight norm of new classes will result in an imbalance in the prediction results. These results are consistent with the confusion matrix in Figure 18, and we can find that most CIL methods can relieve the imbalance in the fully connected layers.

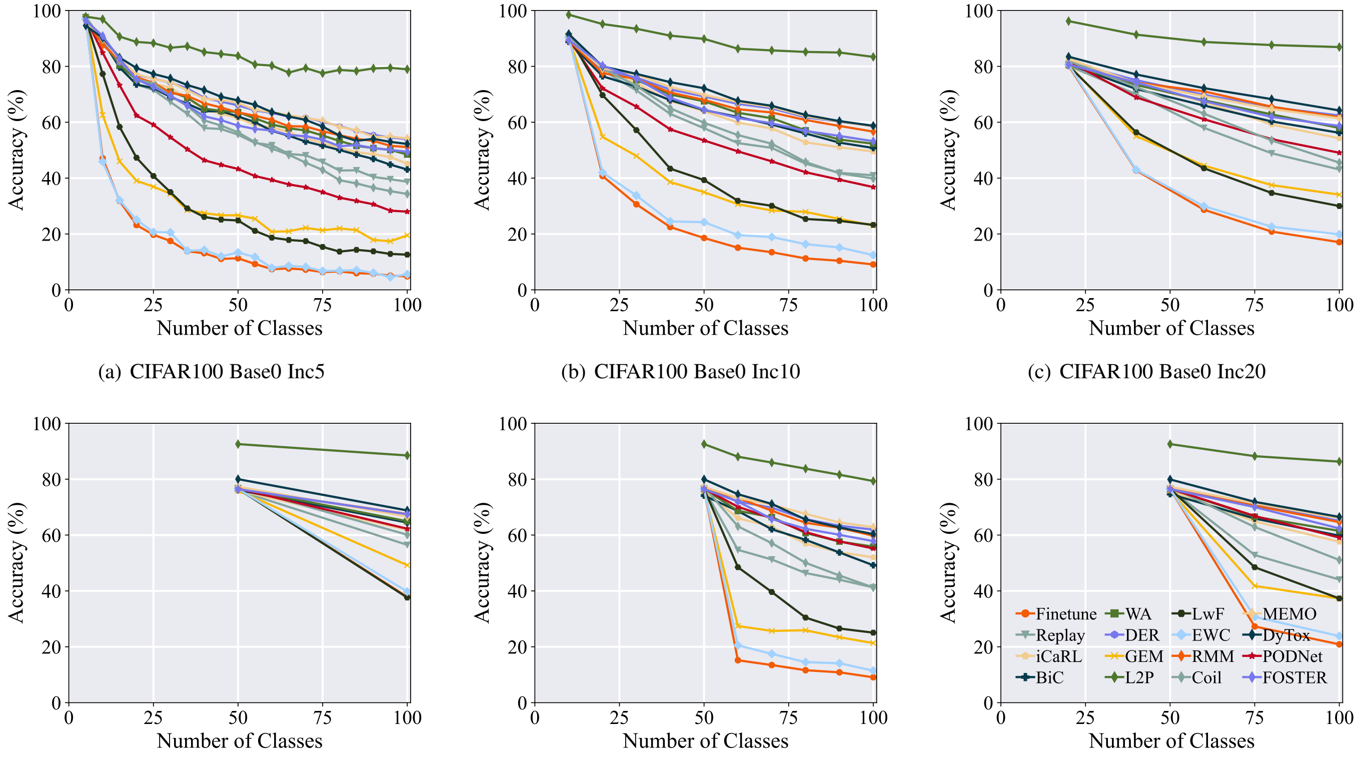
Appendix C Supplied Results in Memory-Aligned Comparison

In the main paper, we report the incremental performance curve with aligned memory to DER. In this section, we report the detailed setting of those figures, including the number of exemplars, the number of parameters, the total memory size, and the detailed performance in Table 7. Specifically, since DER and MEMO require an extra model budget than other methods, we equip more exemplars to other methods to make sure the total memory size is aligned to the same scale.

As we can infer from the table, the gap between different methods becomes smaller when the total budget is aligned to the same scale. Specifically, the typical baseline method iCaRL shows the best average accuracy in the CIFAR100 Base0 Inc5 setting. In other words, saving exemplars can be more memory-efficient than saving backbones when fairly compared.

Appendix D Details About the AUC-A/L Measure

In this section, we report the details in the memory-agnostic measure, *i.e.*, AUC-A/L for class-incremental learning. We first



(a) CIFAR100 Base0 Inc5

(b) CIFAR100 Base0 Inc10

(c) CIFAR100 Base0 Inc20

(d) CIFAR100 Base50 Inc50

(e) CIFAR100 Base50 Inc10

(f) CIFAR100 Base50 Inc25

Figure 14: Incremental accuracy of different methods on CIFAR100. Legends are shown in (f).

Table 5: Average and last accuracy performance comparison on CIFAR100. '#P' represents the number of parameters (million).

Method	Base0 Inc5			Base0 Inc10			Base0 Inc20			Base50 Inc10		
	#P	$\bar{A}$	$\mathcal{A}_B$	#P	$\bar{A}$	$\mathcal{A}_B$	#P	$\bar{A}$	$\mathcal{A}_B$	#P	$\bar{A}$	$\mathcal{A}_B$
Finetune	0.46	17.59	4.83	0.46	26.25	9.09	0.46	37.90	17.07	0.46	22.79	9.09
EWC	0.46	18.42	5.58	0.46	29.73	12.44	0.46	39.19	19.87	0.46	25.77	11.47
LwF	0.46	30.93	12.60	0.46	43.56	23.25	0.46	48.96	30.00	0.46	41.12	25.06
GEM	0.46	31.73	19.48	0.46	40.18	23.03	0.46	50.33	34.09	0.46	33.28	21.33
Replay	0.46	58.20	38.69	0.46	59.31	41.01	0.46	60.03	43.08	0.46	52.37	41.26
RMM	0.46	65.72	51.10	0.46	68.54	56.64	0.46	70.64	61.81	0.46	67.53	59.75
iCaRL	0.46	63.51	45.12	0.46	64.42	49.52	0.46	67.00	54.23	0.46	61.29	52.04
PODNet	0.46	47.88	27.99	0.46	55.22	36.78	0.46	62.96	49.08	0.46	64.45	55.21
Coil	0.46	57.68	34.33	0.46	60.27	39.85	0.46	63.33	45.54	0.46	55.71	41.24
WA	0.46	64.65	48.46	0.46	67.09	52.30	0.46	68.51	57.97	0.46	64.32	55.85
BiC	0.46	62.38	43.08	0.46	65.08	50.79	0.46	67.03	56.22	0.46	61.01	49.19
FOSTER	0.46	63.38	49.42	0.46	66.49	53.21	0.46	68.60	58.67	0.46	65.73	57.82
DER	9.27	67.99	53.95	4.60	69.74	**58.59**	2.30	**70.82**	**62.40**	2.76	68.24	61.94
MEMO	7.14	**68.10**	**54.23**	3.62	**70.20**	58.49	1.87	70.43	61.39	2.22	**69.39**	**62.83**
DyTox	10.7	68.06	52.23	10.7	71.07	58.72	10.7	73.05	64.22	10.7	69.07	60.35
L2P	85.7	84.00	78.96	85.7	89.35	83.39	85.7	90.17	86.91	85.7	85.21	79.34

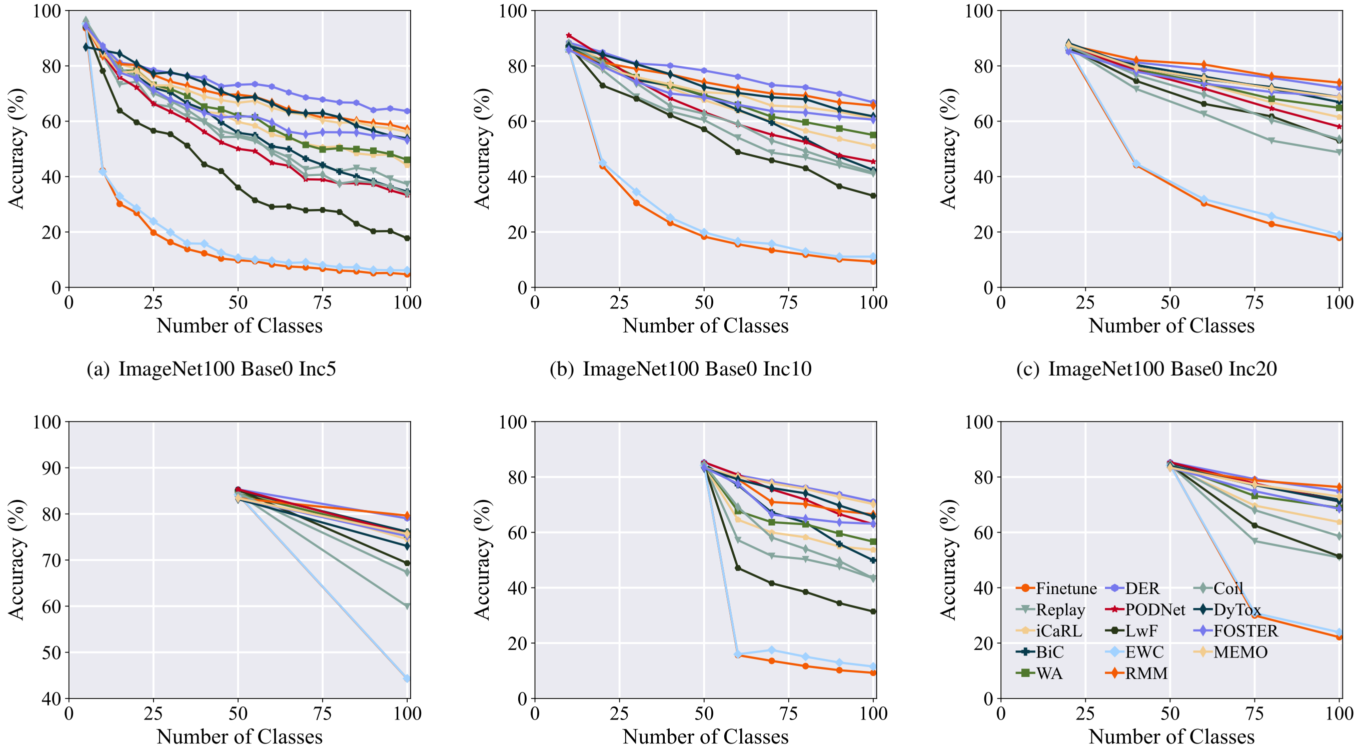
highlight the importance of this measure and then give the implementation details of the curve, including the per-node performance and memory alignment details. We will start with CIFAR100 [31] and then discuss ImageNet100 [4].

D.1 Necessity of AUC in CIL

We report the full performance-memory curve in Figure 20, including the average accuracy-memory curve and last accuracy-memory curve. In each figure, the X-coordinate indicates the memory budget of the specific model, and the Y-coordinate represents the corresponding performance of the model. To highlight the importance of the AUC measure, we begin with a simple question:

> Given *the same memory budget*, which algorithm should
> I choose for class-incremental learning? iCaRL or DER?

It is a simple question that aims to explore a better algorithm with the same memory budget. However, when we compare the performance of iCaRL and DER in Figure 20(a), we find an intersection among these methods. Specifically, we find iCaRL has better accuracy when the memory size is 7.4MB, while DER works better when the memory size is 23.5MB. This results in a *contradictory* conclusion, and we cannot find a model that suits all



(a) ImageNet100 Base0 Inc5

(b) ImageNet100 Base0 Inc10

(c) ImageNet100 Base0 Inc20

(d) ImageNet100 Base50 Inc50

(e) ImageNet100 Base50 Inc10

(f) ImageNet100 Base50 Inc25

Figure 15: Incremental top-1 accuracy of different methods on ImageNet100. Legends are shown in (f).

Table 6: Average and last top-1 accuracy performance comparison on ImageNet100. '#P' represents the number of parameters (million).

Method	Base0 Inc5			Base0 Inc10			Base0 Inc20			Base50 Inc10		
	#P	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	#P	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	#P	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	#P	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	11.17	17.06	4.70	11.17	26.19	9.30	11.17	40.20	17.86	11.17	24.12	9.26
EWC	11.17	18.78	6.14	11.17	27.78	11.10	11.17	41.54	18.98	11.17	26.21	11.54
LwF	11.17	41.76	17.74	11.17	55.50	33.10	11.17	68.43	53.00	11.17	46.24	31.42
Replay	11.17	56.37	37.32	11.17	59.21	41.00	11.17	64.53	48.76	11.17	55.73	43.38
RMM	11.17	69.70	57.16	11.17	74.07	65.66	11.17	**80.08**	**73.96**	11.17	73.02	66.52
iCaRL	11.17	62.36	44.10	11.17	67.11	50.98	11.17	73.57	61.50	11.17	62.56	53.68
PODNet	11.17	53.70	33.34	11.17	64.03	45.40	11.17	71.99	58.04	11.17	73.83	62.94
Coil	11.17	56.21	34.00	11.17	61.91	41.50	11.17	69.49	53.54	11.17	59.80	43.40
WA	11.17	62.96	46.06	11.17	68.60	55.04	11.17	74.44	64.84	11.17	65.81	56.64
BiC	11.17	58.03	34.56	11.17	65.13	42.40	11.17	76.29	66.92	11.17	66.36	49.90
FOSTER	11.17	64.45	53.18	11.17	69.36	60.58	11.17	75.27	68.88	11.17	69.85	63.12
DER	223.4	**73.79**	**63.66**	111.7	**77.08**	**66.84**	55.85	78.56	72.10	67.02	**77.57**	**71.10**
MEMO	170.6	68.19	56.10	86.72	71.00	60.96	44.75	76.59	68.64	53.14	76.66	70.22
DyTox	11.00	69.57	53.82	11.00	73.40	61.78	11.00	76.81	68.78	11.00	74.65	65.76

memory budgets.

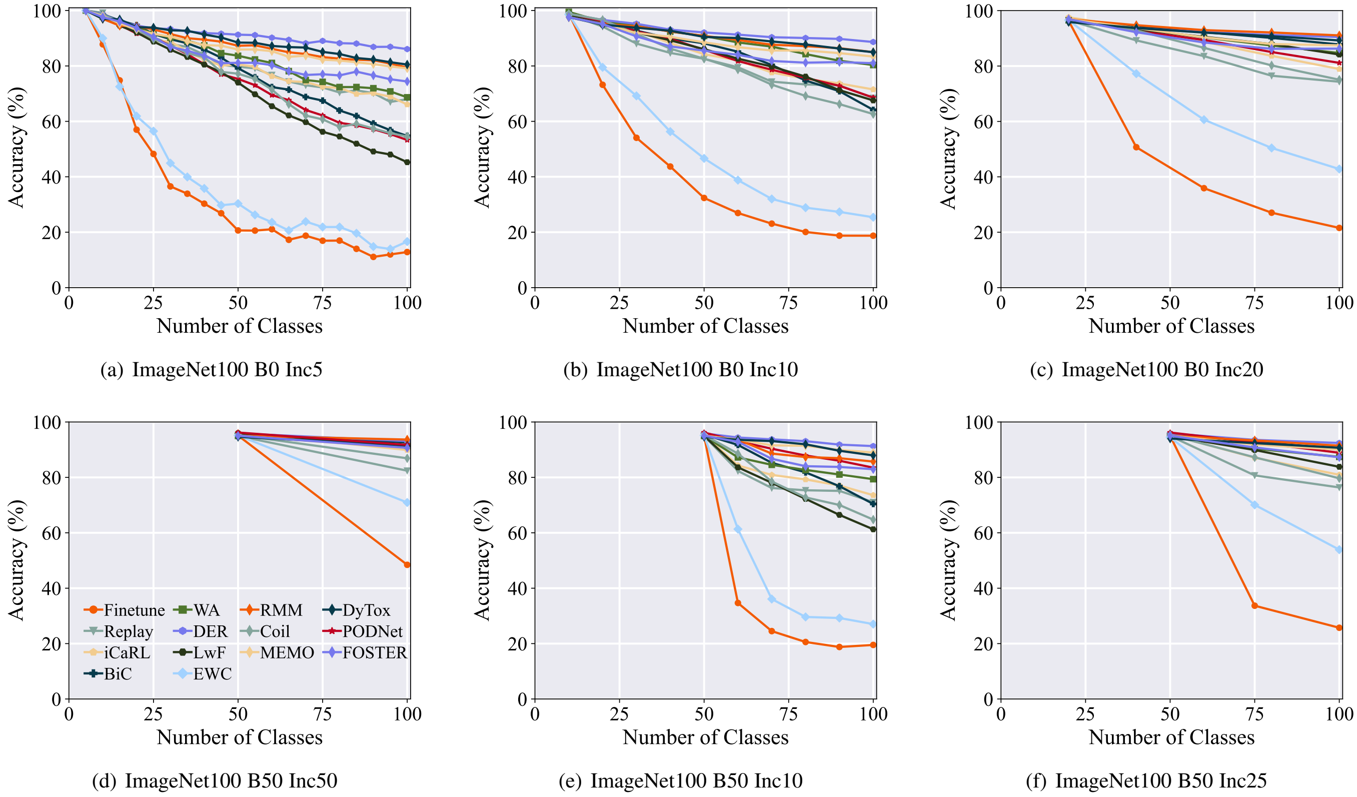
To this end, calculating AUC does not rely on the specific budget, and we can tell from the AUC table that DER has a better AUC performance, which means it has better expandability. In other words, accuracy can only measure the performance given a specific X-coordinate. In contrast, AUC measures the area under the incremental performance curve holistically.

D.2 Implementation Details

We give the implementation details in plotting the performance-memory curve in this section. In the following discussions, we use $\mathcal{E}$ to represent the exemplar set. $\#\mathcal{E}$ denotes the number of exemplars, and $S(\mathcal{E})$ represents the memory size (in MB) that saving these exemplars consume. Following the benchmark implementation in the main paper, 2,000 exemplars are saved for every method for CIFAR100 and ImageNet100. Hence, the exemplar size of each method is denoted as $\#\mathcal{E}= 2000 + E$, where E corresponds to the extra exemplars exchanged from the model size, as discussed in the main paper. We use '#P' to represent the number of parameters and 'MS' to represent the memory budget (in MB) it costs to save this model in memory. The total memory size (*i.e.*, numbers on the X coordinate) is the summation of exemplars and the models:

$$\text{Memory Size} = \text{Model Size} + \text{Exemplar Size}\,, \tag{18}$$

which should be aligned for fair comparison as we advocated.



(a) ImageNet100 B0 Inc5

(b) ImageNet100 B0 Inc10

(c) ImageNet100 B0 Inc20

(d) ImageNet100 B50 Inc50

(e) ImageNet100 B50 Inc10

(f) ImageNet100 B50 Inc25

Figure 16: Incremental top-5 accuracy of different methods on ImageNet100. Legends are shown in (d).

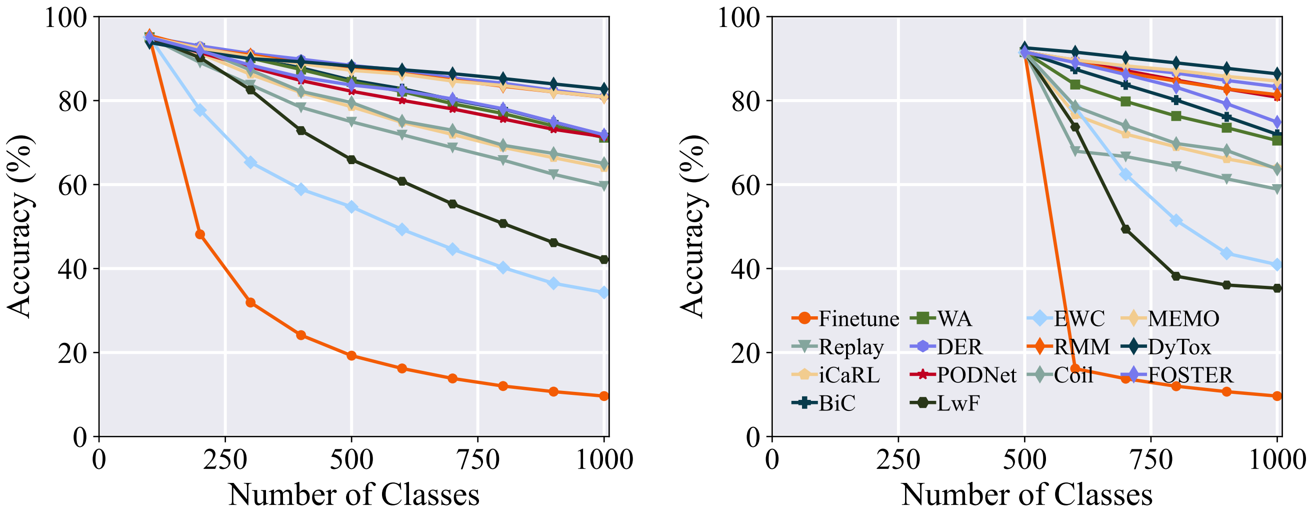
Table 7: Average and last accuracy performance comparison with aligned memory cost. ‘#P’ represents the number of parameters (million). ‘#$\mathcal{E}$’ denotes the number of exemplars, and ‘MS’ denotes the memory size (MB).

Method	CIFAR100 Base0 Inc5					CIFAR100 Base0 Inc10					ImageNet100 Base0 Inc10				
	#P	#$\mathcal{E}$	MS	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	#P	#$\mathcal{E}$	MS	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	#P	#$\mathcal{E}$	MS	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
GEM	0.46	13466	41.22	20.30	7.99	0.46	7431	23.5	27.03	10.72	-	-	-	-	-
Replay	0.46	13466	41.22	74.25	61.09	0.46	7431	23.5	69.97	55.61	11.17	4671	713.2	67.46	53.72
iCaRL	0.46	13466	41.22	**75.17**	61.88	0.46	7431	23.5	70.94	58.52	11.17	4671	713.2	67.16	51.58
PODNet	0.46	13466	41.22	59.03	42.98	0.46	7431	23.5	60.80	45.38	11.17	4671	713.2	68.43	53.44
Coil	0.46	13466	41.22	74.39	60.78	0.46	7431	23.5	70.69	54.40	11.17	4671	713.2	69.97	54.24
WA	0.46	13466	41.22	74.07	**63.18**	0.46	7431	23.5	69.55	59.26	11.17	4671	713.2	72.62	62.12
BiC	0.46	13466	41.22	71.27	60.98	0.46	7431	23.5	70.69	59.60	11.17	4671	713.2	72.83	61.98
FOSTER	0.46	13466	41.22	75.06	62.82	0.46	7431	23.5	72.28	59.39	11.17	4671	713.2	71.55	59.86
DER	9.27	2000	41.22	70.41	57.85	4.60	2000	23.5	71.47	60.26	111.7	2000	713.2	**75.70**	**67.14**
MEMO	7.14	4771	41.22	73.57	62.97	3.62	3312	23.5	**72.37**	**61.98**	86.72	2652	713.2	72.55	64.08

Specifically, *we can divide the selected methods into two groups*. The first group contains GEM [59], iCaRL [32], Replay [196], WA [115], PODNet [98], Coil [92], BiC [87] and FOSTER [21], whose use a single backbone for incremental learning in the inference stage. Hence, they use the same network backbone with the same model size and equal memory sizes, and we denote them as **SingleNet**. The second group contains DER [20] and MEMO [22], which require more memory budget to save the extra model during inference. Specifically, DER sacrifices the memory size to store the backbone from history, which consumes the largest model size. On the other hand, compared to DER, MEMO does not keep the duplicated generalized blocks from history and saves much memory size to change into exemplars.

Discussion about selected methods: In this part, we choose ten methods for the comparison, *i.e.*, Finetune, EWC, LwF, RMM, DyTox, and L2P are not included in the evaluation. Firstly, non-exemplar-based methods (Finetune, EWC, LwF) are unsuitable for the evaluation since the extendability relies on adding extra exemplars. Secondly, RMM advocates another evaluation protocol in terms of the *memory*, which is incompatible with ours. Lastly, DyTox and L2P rely on the vision transformer as the backbone, which have a much larger memory scale than benchmark backbones. Directly comparing ResNet to ViT may be unfair since these backbones have different characteristics in model optimization.

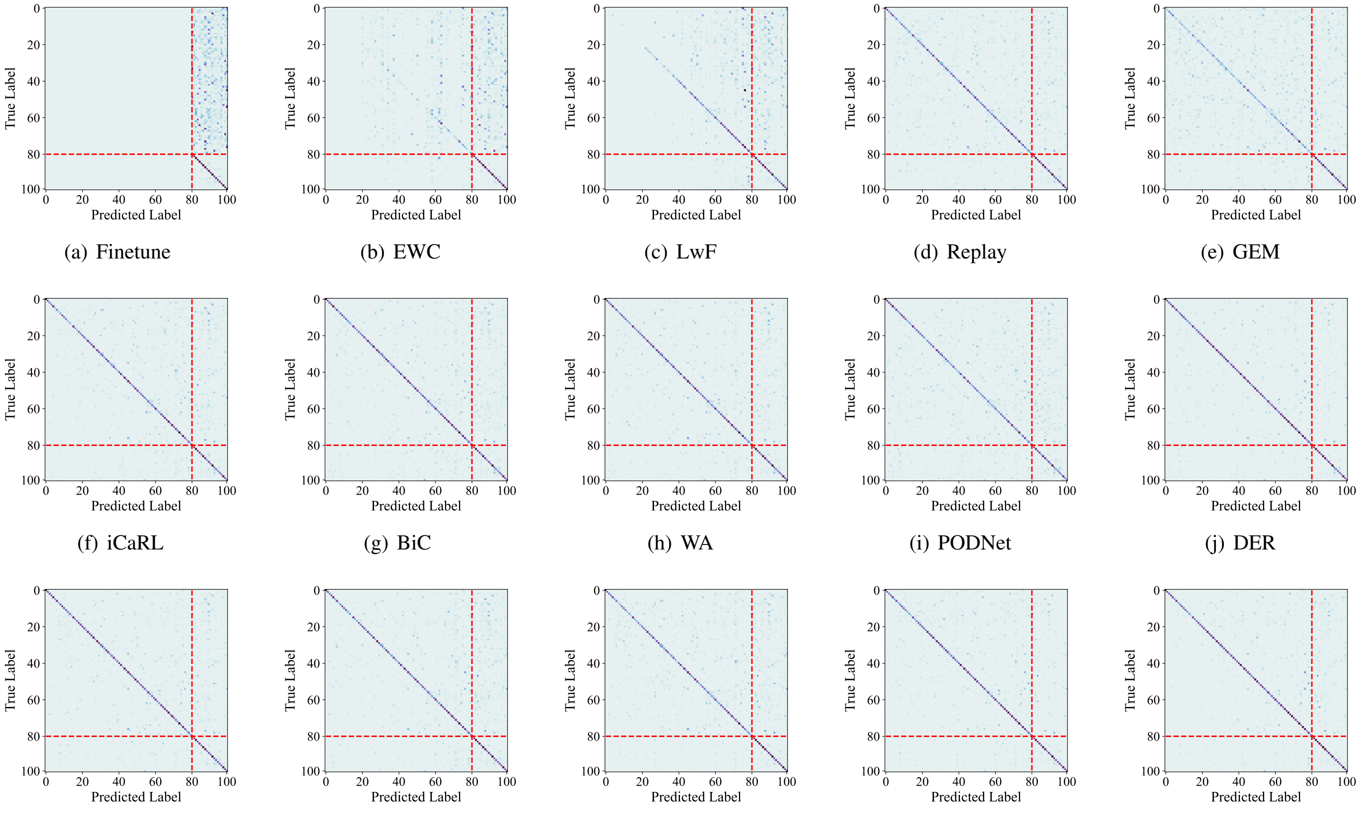
How to choose the budget list? As we can infer from Figure 20, there are several selected memory sizes in the figures, which formulate the X-coordinate. Firstly, the start point corresponds to the memory size of the first group, *i.e.*, a single backbone with $2,000$ exemplars. It is a relatively small budget, which can be seen as the budget for edge devices. Correspondingly, since DER has the largest model size among all compared methods, we set the endpoint to the memory size of DER. After selecting the start and



(a) ImageNet1000 B0 Inc100

(b) ImageNet1000 B500 Inc100

Figure 17: Incremental top-5 accuracy of different methods on ImageNet1000. Legends are shown in (b).



(a) Finetune

(b) EWC

(c) LwF

(d) Replay

(e) GEM

(f) iCaRL

(g) BiC

(h) WA

(i) PODNet

(j) DER

(k) RMM+FOSTER

(l) Coil

(m) FOSTER

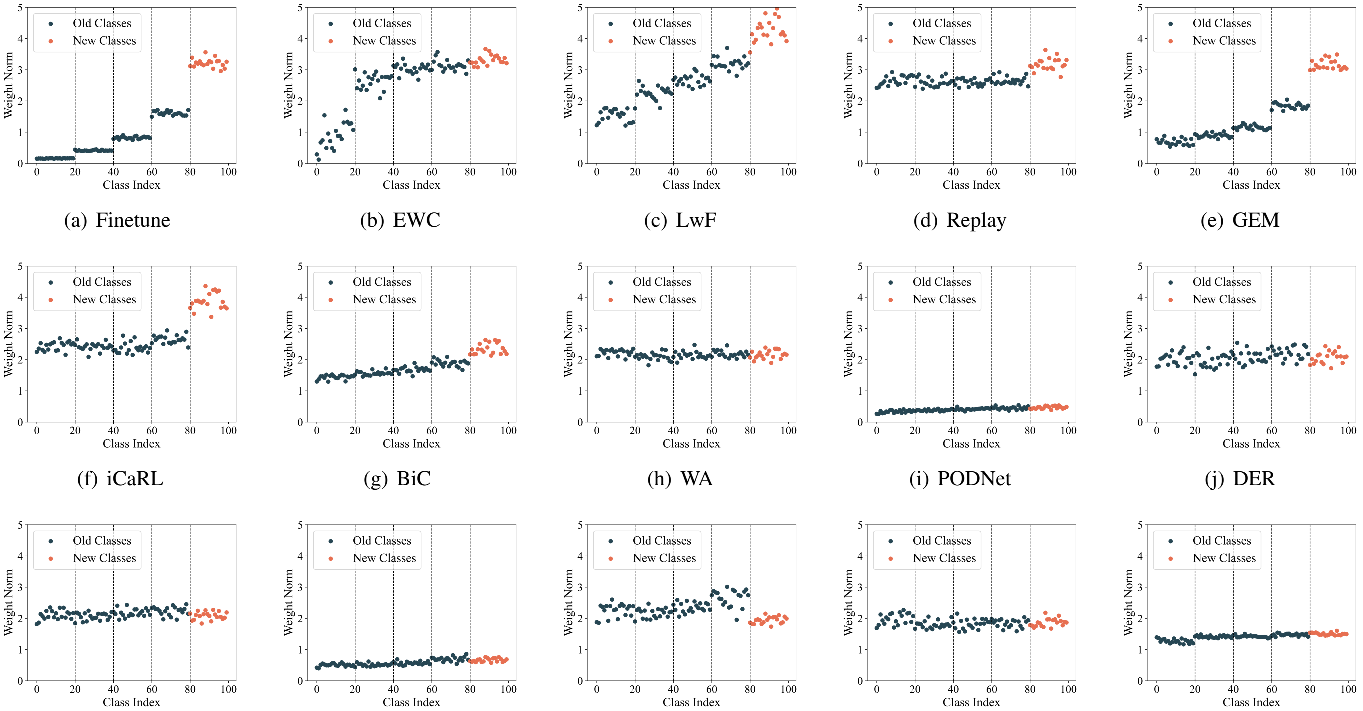
(n) MEMO

(o) Dytox

Figure 18: Confusion matrix of different methods on CIFAR100 Base0 Inc20 setting after the last incremental stage.

end points, we choose several intermediate budgets to formulate the budget list. The intermediate budgets are set according to the parameter size of DER and MEMO, as shown below.

How to set the exemplar and model size? For SingleNet, we can easily extend them by adding the number of exemplars. Since the model size is fixed for them, adding exemplars enables these methods to extend to a larger scale. However, we cannot use the same backbone as SingleNet for DER and MEMO when the memory size is small. Hence, we divide the model parameters into ten equal parts (since there are ten incremental tasks in

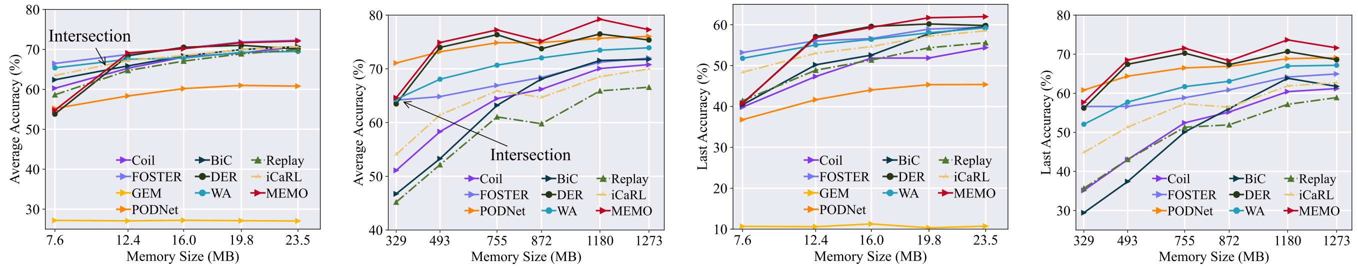


(a) Finetune (b) EWC (c) LwF (d) Replay (e) GEM

(f) iCaRL (g) BiC (h) WA (i) PODNet (j) DER

(k) RMM+FOSTER (l) Coil (m) FOSTER (n) MEMO (o) Dytox

Figure 19: Weight norm of different methods on CIFAR100 Base0 Inc20 setting. We visualize the norm after the last incremental stage.



(a) Average accuracy curve of CIFAR100 Base0 Inc10. (b) Average accuracy curve of ImageNet100 Base50 Inc5. (c) Last accuracy curve of CIFAR100 Base0 Inc10. (d) Last accuracy curve of ImageNet100 Base50 Inc5.

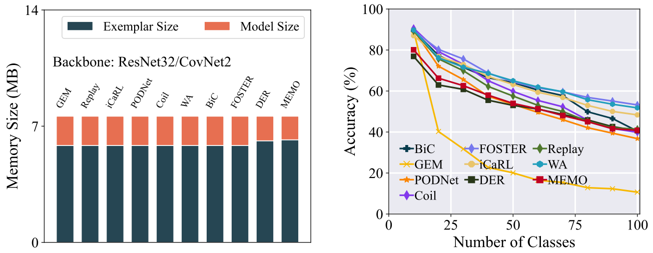
Figure 20: Performance-memory curve of different methods with different datasets.

the CIFAR100 Base0 Inc10 setting) and look for a backbone with similar parameter numbers. For example, we separately use ConvNet, ResNet14, ResNet20, and ResNet26 as the backbone for these methods to match different memory scales. Specifically, we use the same backbone for DER and MEMO and another same backbone for SingleNet to keep the total memory budget at the same scale. We annotate the backbone type in the figures of configurations, *e.g.*, 'ResNet32/ResNet14' means we use ResNet32 for SingleNet and ResNet14 for DER and MEMO.

In the following discussions, we give the memory figures and tables to illustrate the implementation of different methods and report their incremental performance. We start with CIFAR100 and then discuss ImageNet100.

D.2.1 CIFAR100 Implementations

There are five X coordinates in the curve of CIFAR100, *e.g.*, 7.6, 12.4, 16.0, 19.8, and 23.5 MB. Following, we show the detailed implementation of different methods at these scales.



(a) Memory Usage (b) Performance Curve

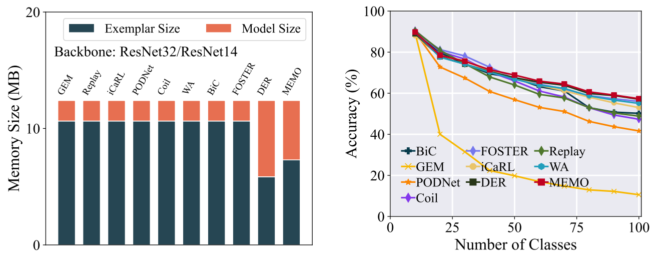
Figure 21: Implementation details and performance curve of CIFAR100 when memory size=7.6 MB.

Table 8: Numerical details when memory size=7.6 MB.

7.6MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	2000	5.85MB	ResNet32	0.46M	1.76MB
DER	2096	6.14MB	ConvNet2	0.38M	1.48MB
MEMO	2118	6.20MB	ConvNet2	0.37M	1.42MB

CIFAR100 with 7.6 MB Memory Size: The implementations are shown in Figure 21, and the memory size (7.6 MB) is relatively small. Since we need to align the total budget of these methods, we are only able to use small backbones for DER and MEMO. These small backbones, *i.e.*, *ConvNet with two convolutional layers*, have much fewer parameters than ResNet32 and saving 10 ConvNets matches the memory size of a single ResNet32 (1.48MB versus 1.76MB). We can infer that DER and MEMO are restricted by the learning ability of these small backbones, which perform poorly in the base session. These results are consistent with the conclusions in Figure 20 that dynamic networks are inferior to SingleNet given a small memory budget.

We report the implementation details, including the number of exemplars, size of exemplars, type of backbones, number of parameters, and size of models in Table 8. The total memory budget, *i.e.*, 7.6MB, is the summation of the exemplar size and model size.



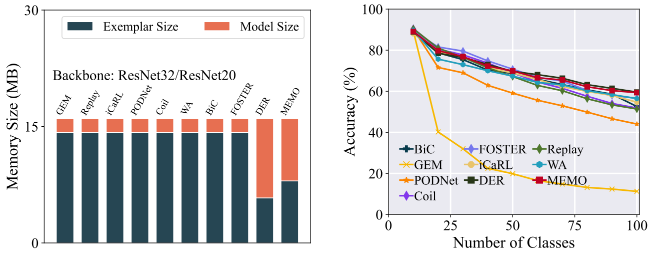
(a) Memory Usage (b) Performance Curve

Figure 22: Implementation details and performance curve of CIFAR100 when memory size=12.4 MB.

Table 9: Numerical details when memory size=12.4 MB.

12.4MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	3634	10.64MB	ResNet32	0.46M	1.76MB
DER	2000	5.85MB	ResNet14	1.70M	6.55MB
MEMO	2495	7.32MB	ResNet14	1.33M	5.10MB

CIFAR100 with 12.4 MB Memory Size: The implementations are shown in Figure 22. By raising the total memory cost to 12.4 MB, SingleNet can utilize the extra memory size to exchange 1634 exemplars. At the same time, dynamic networks can switch to more powerful backbones, *i.e.*, ResNet14, to get better representation ability. We can infer that DER and MEMO show competitive results with stronger backbones and outperform other methods in this setting. These results are consistent with the conclusions in Figure 20 that the *intersection* between these two groups of methods exists near the start point.



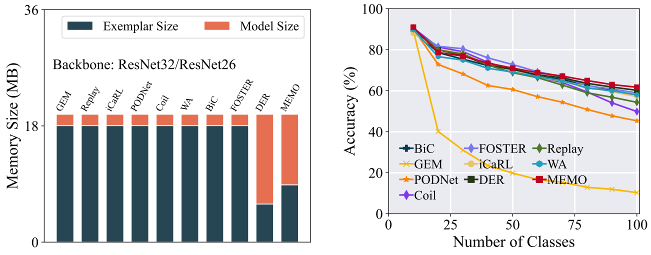
(a) Memory Usage (b) Performance Curve

Figure 23: Implementation details and performance curve of CIFAR100 when memory size=16.0 MB.

Table 10: Numerical details when memory size=16.0 MB.

16.0MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	4900	14.3MB	ResNet32	0.46M	1.76MB
DER	2000	5.85MB	ResNet20	2.69M	10.2MB
MEMO	2768	8.10MB	ResNet20	2.1M	8.01MB

CIFAR100 with 16.0 MB Memory Size: The implementations are shown in Figure 23. By raising the total memory cost to 16.0 MB, SingleNet can utilize the extra memory size to exchange 2900 exemplars, and dynamic networks can switch to larger backbones to get better representation ability. We use ResNet20 for DER and MEMO in this setting. The results are consistent with the former setting, where we can see that DER and MEMO show competitive results with stronger backbones and outperform other methods.



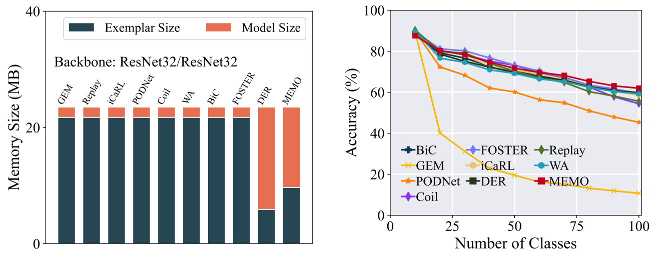
(a) Memory Usage (b) Performance Curve

Figure 24: Implementation details and performance curve of CIFAR100 when memory size=19.8 MB.

Table 11: Numerical details when memory size=19.8 MB.

19.8MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	6165	18.06MB	ResNet32	0.46M	1.76MB
DER	2000	5.85MB	ResNet26	3.60M	13.9MB
MEMO	3040	8.91MB	ResNet26	2.86M	10.92MB

CIFAR100 with 19.8 MB Memory Size: The implementations are shown in Figure 24. By raising the total memory cost to 19.8 MB, SingleNet can utilize the extra memory size to exchange 4165 exemplars, and dynamic networks can switch to larger backbones to get better representation ability. We use ResNet26 for DER and MEMO in this setting. The results are consistent with the former setting, where we can infer that dynamic networks show competitive results with stronger backbones and outperform other methods.



(a) Memory Usage (b) Performance Curve

Figure 25: Implementation details and performance curve of CIFAR100 when memory size=23.5 MB.

Table 12: Numerical details when memory size=23.5 MB.

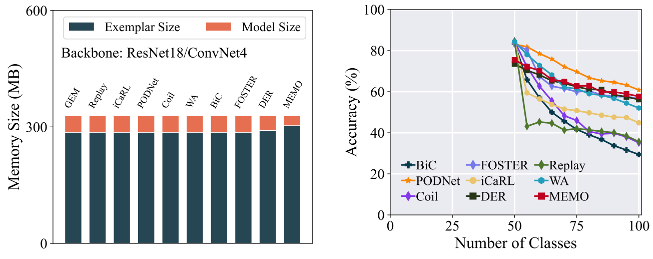
23.5MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	7431	21.76MB	ResNet32	0.46M	1.75MB
DER	2000	5.86MB	ResNet32	4.63M	17.68MB
MEMO	3312	9.7MB	ResNet32	3.62M	13.83MB

CIFAR100 with 23.5 MB Memory Size: The implementations are shown in Figure 25. By raising the total memory cost to 23.5 MB, SingleNet can utilize the extra memory size to exchange 5431 exemplars. In addition, dynamic networks can switch to larger backbones, *i.e.*, ResNet32, to get better representation ability.

D.2.2 ImageNet100 Implementations

Similar to CIFAR100, we can conduct an exchange between the model and exemplars with the ImageNet100 dataset. For example, saving a ResNet18 model costs $11,176,512$ parameters (float), while saving an ImageNet image costs $3 \times 224 \times 224$ integer numbers (int). The budget for saving a backbone is equal to saving $11,176,512$ floats $\times 4$ bytes/float $\div (3 \times 224 \times 224)$ bytes/image ≈ 297 images for ImageNet. We conduct the experiment with the ImageNet100 Base50 Inc5 setting, as discussed in the main paper.

There are six X coordinates in the curve of ImageNet100, *e.g.*, 329, 493, 755, 872, 1180 and 1273 MB. Following, we show the detailed implementation of different methods at these scales.



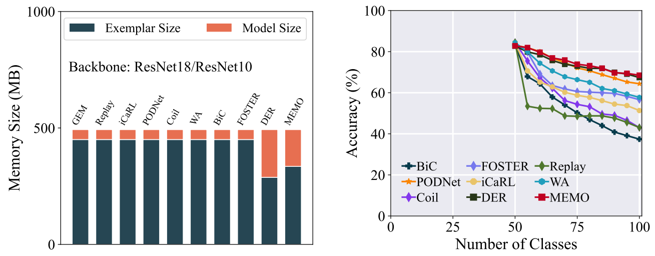
(a) Memory Usage (b) Performance Curve

Figure 26: Implementation details and performance curve of ImageNet100 when memory size=329 MB.

Table 13: Numerical details when memory size=329 MB.

329MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	2000	287MB	ResNet18	11.17M	42.6MB
DER	2032	291MB	ConvNet4	9.96M	38.0MB
MEMO	2115	303MB	ConvNet4	6.81M	26.0MB

ImageNet100 with 329 MB Memory Size: The implementations are shown in Figure 26. 329 MB is a relatively small memory size. Since we need to align the total budget of these methods, we are only able to use small backbones for DER and MEMO. These small backbones, *i.e.*, *ConvNet with four convolutional layers*, have much fewer parameters than ResNet18 and saving 10 ConvNets matches the memory size of a single ResNet18. We can infer from the figure that DER and MEMO are restricted by the learning ability of the inferior backbones, which perform poorly in the base task.



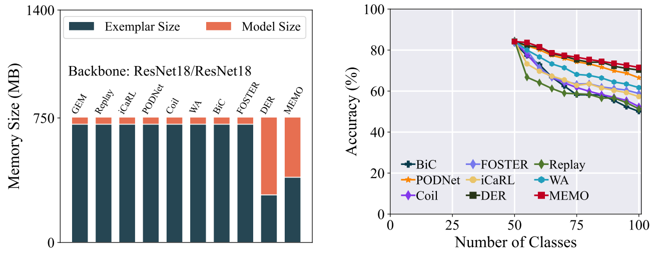
(a) Memory Usage (b) Performance Curve

Figure 27: Implementation details and performance curve ImageNet100 when memory size=493 MB.

Table 14: Numerical details when memory size=493 MB.

493MB	$\#\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	3136	450MB	ResNet18	11.17M	42.6MB
DER	2000	287MB	ResNet10	53.96M	205MB
MEMO	2327	334MB	ResNet10	41.63M	158MB

ImageNet100 with 493 MB Memory Size: The implementations are shown in Figure 27. By raising the total memory cost to 493 MB, SingleNet can utilize the extra memory size to exchange 1136 exemplars. In addition, dynamic networks can switch to larger backbones, *i.e.*, ResNet10, to get better representation ability. We can infer from the figure that dynamic networks show competitive results with stronger backbones and outperform other methods in this setting.



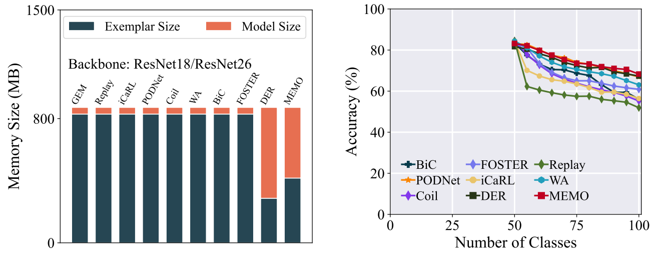
(a) Memory Usage (b) Performance Curve

Figure 28: Implementation details and performance curve ImageNet100 when memory size=755 MB.

Table 15: Numerical details when memory size=755 MB.

755MB	#$\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	4970	713MB	ResNet18	11.17M	42.6MB
DER	2000	287MB	ResNet18	122.9M	468MB
MEMO	2739	393MB	ResNet18	95.11M	362MB

ImageNet100 with 755 MB Memory Size: The implementations are shown in Figure 28. By raising the total memory cost to 755 MB, SingleNet can utilize the extra memory size to exchange 2970 exemplars. At the same time, dynamic networks can switch to larger backbones, *i.e.*, ResNet18, to get better representation ability.



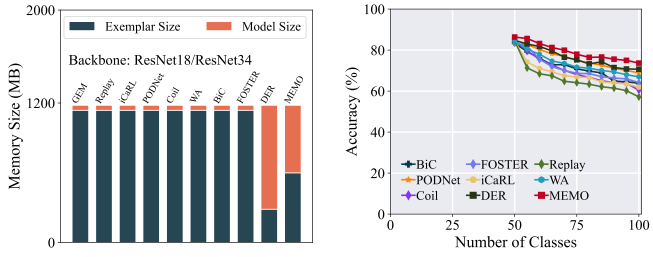
(a) Memory Usage (b) Performance Curve

Figure 29: Implementation details and performance curve ImageNet100 when memory size=872 MB.

Table 16: Numerical details when memory size=872 MB.

872MB	#$\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	5779	829MB	ResNet18	11.17M	42.6MB
DER	2000	287MB	ResNet26	153.4M	585.2MB
MEMO	2915	417MB	ResNet26	119.0M	453MB

ImageNet100 with 872 MB Memory Size: The implementations are shown in Figure 29. By raising the total memory cost to 872 MB, SingleNet can utilize the extra memory size to exchange 3779 exemplars. In addition, dynamic networks can switch to larger backbones, *i.e.*, ResNet26 for better representation ability.



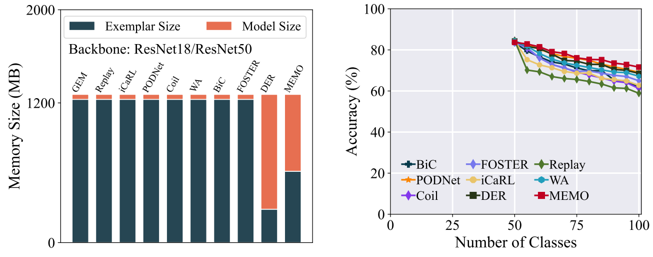
(a) Memory Usage (b) Performance Curve

Figure 30: Implementation details and performance curve ImageNet100 when memory size=1180 MB.

Table 17: Numerical details when memory size=1180 MB.

1180MB	#$\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	7924	1137MB	ResNet18	11.17M	42.6MB
DER	2000	287MB	ResNet34	234.1M	893MB
MEMO	4170	598MB	ResNet34	152.4M	581MB

ImageNet100 with 1180 MB Memory Size: The implementations are shown in Figure 30. By raising the total memory cost to 1180 MB, non-dynamic networks can utilize the extra memory size to exchange 5924 exemplars. At the same time, dynamic networks can switch to larger backbones, *i.e.*, ResNet34, to get better representation ability.



(a) Memory Usage (b) Performance Curve

Figure 31: Implementation details and performance curve ImageNet100 when memory size = 1273 MB.

Table 18: Numerical details when memory size=1273 MB.

1273MB	#$\mathcal{E}$	$S(\mathcal{E})$	Model Type	#P	Model Size
SingleNet	8574	1230MB	ResNet18	11.17M	42.6MB
DER	2000	287MB	ResNet50	258.6M	986MB
MEMO	4270	612MB	ResNet50	173.2M	660MB

ImageNet100 with 1273 MB Memory Size: The implementations are shown in Figure 31. By raising the total memory cost to 1273 MB, SingleNet can utilize the extra memory size to exchange 6574 exemplars, and dynamic networks can switch to larger backbones to get better representation ability. We use ResNet50 for DER and MEMO in this setting.

Discussion about Backbones: It should be noted that ResNet18 is the benchmark backbone for ImageNet, and DER consumes about

755 MB memory budget under the benchmark setting. Hence, the last three points in the X-coordinate, *i.e.*, 872, 1180, and 1273 MB, are larger than the benchmark setting. There are two main reasons for the budget list design. First, handling large-scale image inputs requires more convolutional layers, and it is hard to find typical models with small memory budgets. Second, we would like to investigate the performance when the model is large enough to see whether the improvement of stronger backbones will converge. The empirical results successfully verify our assumptions.