Mnemonics Training: Multi-Class Incremental Learning without Forgetting

Yaoyao Liu[1,2*] An-An Liu[2] Yuting Su[2] Bernt Schiele[1] Qianru Sun[3†]

[1]Max Planck Institute for Informatics, Saarland Informatics Campus
[2]School of Electrical and Information Engineering, Tianjin University
[3]School of Information Systems, Singapore Management University

{yaoyao.liu, schiele}@mpi-inf.mpg.de
{liuanan, ytsu}@tju.edu.cn qianrusun@smu.edu.sg

Abstract

Multi-Class Incremental Learning (MCIL) aims to learn new concepts by incrementally updating a model trained on previous concepts. However, there is an inherent trade-off to effectively learning new concepts without catastrophic forgetting of previous ones. To alleviate this issue, it has been proposed to keep around a few examples of the previous concepts but the effectiveness of this approach heavily depends on the representativeness of these examples. This paper proposes a novel and automatic framework we call mnemonics*, where we parameterize exemplars and make them optimizable in an end-to-end manner. We train the framework through bilevel optimizations, i.e., model-level and exemplar-level. We conduct extensive experiments on three MCIL benchmarks, CIFAR-100, ImageNet-Subset and ImageNet, and show that using* mnemonics *exemplars can surpass the state-of-the-art by a large margin. Interestingly and quite intriguingly, the* mnemonics *exemplars tend to be on the boundaries between classes.*[1]

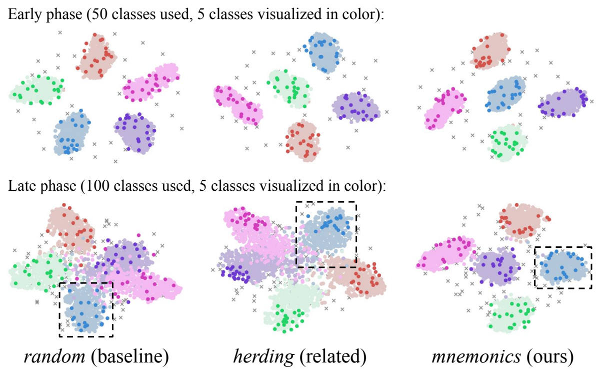


Figure 1. The t-SNE [17] results of three exemplar methods in two phases. The original data of 5 colored classes occur in the early phase. In each colored class, deep-color points are exemplars, and light-color ones show the original data as reference of the real data distribution. Gray crosses represent other participating classes, and each cross for one class. We have two main observations. (1) Our approach results in much clearer separation in the data, than *random* (where exemplars are randomly sampled in the early phase) and *herding* (where exemplars are nearest neighbors of the mean sample in the early phase) [24, 9, 34, 2]. (2) Our learned exemplars mostly locate on the boundaries between classes.

1. Introduction

Natural learning systems such as humans inherently work in an incremental manner as the number of concepts increases over time. They naturally learn new concepts while not forgetting previous ones. In contrast, current machine learning systems, when continuously updated using novel incoming data, suffer from catastrophic forgetting (or catastrophic interference), as the updates can override knowledge acquired from previous data [19, 20, 23, 27, 12]. This is especially true for multi-class incremental learning (MCIL) where one cannot replay all previous inputs. Catastrophic forgetting, therefore, becomes a major problem for MCIL systems.

Motivated by this, a number of works have recently emerged [24, 16, 9, 34, 2]. Rebuffi et al. [24] firstly defined a protocol for evaluating MCIL methods, i.e., to tackle the image classification task where the training data for different classes comes in sequential training phases. As it is neither desirable nor scaleable to retain all data from previous concepts, in their protocol, they restrict the number of exemplars that can be kept around per class: e.g., only 20 exemplars per class can be stored and passed to the subsequent training phases. These "20 exemplars" are important to MCIL as they are the key resource for the model

*This work was done during Yaoyao's internship supervised by Qianru.
†Corresponding author.
[1]Code: https://github.com/yaoyao-liu/mnemonics

to refresh its previous knowledge. Existing methods to extract exemplars are based on heuristically designed rules, e.g., nearest neighbors around the average sample in each class (named *herding* [33]) [24, 9, 34, 2], but turn out to be not particularly effective. For example, iCaRL [24] with *herding* sees an accuracy drop of around 25% in predicting 50 previous classes in the last phase (when the number of classes increases to 100) on CIFAR-100, compared to the upper-bound performance of using all examples. A t-SNE visualization of *herding* exemplars is given in Figure 1, and shows that the separation between classes is weak in later training phases.

In this work, we address this issue by developing an automatic exemplar extraction framework called *mnemonics* where we parameterize the exemplars (using image-size parameters) and then optimize them in an end-to-end scheme. By *mnemonics*, the MCIL model in each phase can not only learn the optimal exemplars from the new class data, but also adjust the exemplars of previous phases to fit the current data distribution. As demonstrated in Figure 1, *mnemonics* exemplars yield consistently clear separations among classes, from early to late phases. When inspecting individual classes (as e.g. denoted by the black dotted frames in Figure 1 for the "blue" class), we observe that the *mnemonics* exemplars (dark blue dots) are mostly located on the boundary of the class data distribution (light blue dots), which is essential to derive high-quality classifiers.

Technically, *mnemonics* has two models to optimize, i.e., the conventional model and the parameterized *mnemonics* exemplars. The two are not independent and can not be jointly optimized, as the exemplars learned in the current phase will act as the input data of later-phase models. We address this issue by a bilevel optimization program (BOP) [28, 18] that alternates the learning of two levels of models. We iterate this optimization through the entire incremental training phases. In particular, for each single phase, we perform a local BOP that aims to distill the knowledge of new class data into the exemplars. First, a temporary model is trained with exemplars as input. Then, a validation loss on new class data is computed and the gradients are back-propagated to optimize the input layer, i.e., the parameters of the *mnemonics* exemplars. Iterating these two steps allows to derive representative exemplars for later training phases. To evaluate the proposed *mnemonics* method, we conduct extensive experiments for four different baseline architectures and on three MCIL benchmarks – CIFAR-100, ImageNet-Subset and ImageNet. Our results reveal that *mnemonics* consistently achieves top performance compared to related works, e.g., 20% and 6.5% higher than *herding*-based iCaRL [24] and LUCIR [9], respectively, in the 25-phase setting on ImageNet [24].

Our contributions include: (1) A novel *mnemonics* training framework that alternates the learning of exemplars and models in a global bilevel optimization program (including *model-level* and *exemplar-level*); (2) A novel local bilevel optimization program (including *meta-level* and *base-level*) that trains exemplars for new classes as well as adjusts exemplars of old classes in an end-to-end manner; (3) In-depth experiments, visualization and explanation of *mnemonics* exemplars in the feature space.

2. Related Work

Incremental learning has a long history in machine learning [3, 21, 14]. A uniform setting is that the data of different classes gradually come. Recent works are either in the multi-task setting (classes from different datasets) [16, 27, 10, 4, 25], or in the multi-class setting (classes from the identical dataset) [24, 9, 34, 2]. Our work is conducted on the benchmarks of the latter one called multi-class incremental learning (MCIL).

A classic baseline method is called knowledge distillation using a transfer set [8], first applied to incremental learning by Li et al. [16]. Rebuffi et al. [24] combined this idea with representation learning, for which a handful of *herding* exemplars are stored for replaying old knowledge. *Herding* [33] picks the nearest neighbors of the average sample per class [24]. With the same *herding* exemplars, Castro et al. [2] tried a balanced fine-tuning and temporary distillation to build an end-to-end framework; Wu et al. [34] proposed a bias correction approach; and Hou et al. [9] introduced multiple techniques also to balance classifiers. Our approach is closely related to these works. The difference lies in the way of generating exemplars. In the proposed *mnemonics* training framework, the exemplars are optimizable and updatable in an end-to-end manner, thus more effective than previous ones.

Using synthesizing exemplars is another solution that "stores" the old knowledge in generative models. Related methods [27, 11, 32] used Generative Adversarial Networks (GAN) [6] to generate old samples in each new phase for data replaying, and good results were obtained in the multi-task incremental setting. However, their performance strongly depends on the GAN models which are notoriously hard to train. Moreover, storing GAN models requires memory, so these methods might not be applicable to MCIL with a strict memory budget. Our *mnemonics* exemplars are optimizable, and can be regarded as synthesized, while our approach is based on the direct parameterization of exemplars without training extra models.

Bilevel optimization program (BOP) aims to solve two levels of problems in one framework where the A-level problem is the constraint to solve the B-level problem. It can be traced back to the Stackelberg competition [29] in the area of game theory. Nowadays, it is widely applied in the area of machine learning. For instance, Training GANs [6] can be formulated as a BOP with two opti-

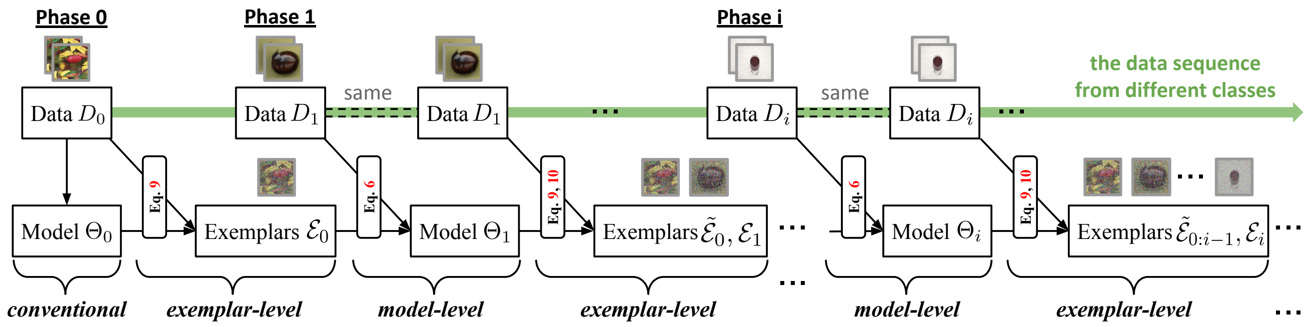


Figure 2. The computing flow of the proposed *mnemonics* training. It is a global BOP that alternates the learning of *mnemonics* exemplars (we call *exemplar-level* optimization) and MCIL models (*model-level* optimization). The *exemplar-level* optimization within each phase is detailed in Figure 3. $\tilde{\mathcal{E}}$ denotes the old exemplars adjusted to the current phase.

mization problems: maximizing the reality score of generated images and minimizing the real-fake classification loss. Meta-learning [5, 31, 35, 15] is another BOP in which a meta-learner is optimized subject to the optimality of the base-learner. Recently, MacKay et al. [18] formulated the hyperparameter optimization as a BOP where the optimal model parameters in a certain time phase depend on hyperparameters, and vice versa. In this work, we introduce a global BOP that alternatively optimizes the parameters of the MCIL models and the *mnemonics* exemplars across all phases. Inside each phase, we exploit a local BOP to learn (or adjust) the *mnemonics* exemplars specific to the new class (or the previous classes).

3. Preliminaries

Multi-Class Incremental Learning (MCIL) was proposed in [24] to evaluate classification models incrementally learned using a sequence of data from different classes. Its uniform setting is used in related works [24, 9, 34, 2]. It is different from the conventional classification setting – where training data for all classes are available from the start – in three aspects: (i) the training data come in as a stream where the sample of different classes occur in different time phases; (ii) in each phase, MCIL classifiers are expected to provide a competitive performance for all seen classes so far; and (iii) the machine memory is limited (or at least grows slowly), so it is impossible to save all data to replay network training.

Denotations. Assume there are $N+1$ phases in the MCIL system (one initial phase and N incremental phases). In the initial phase (i.e., the 0-th phase), we learn the model Θ_0 on data D_0 using a conventional classification loss, e.g. cross-entropy loss, and then save Θ_0 to the memory of the system. Due to the memory limitation, we can not keep the entire D_0, but instead we select and store a handful of exemplars $\mathcal{E}_0$ (evenly for all classes) as a replacement of D_0 with $|\mathcal{E}_0| \ll |D_0|$. In the i-th incremental phase, we denote the previous exemplars $\mathcal{E}_0 \sim \mathcal{E}_{i-1}$ shortly as $\mathcal{E}_{0:i-1}$. We load Θ_{i-1} and $\mathcal{E}_{0:i-1}$ from the memory, and then use $\mathcal{E}_{0:i-1}$ and the new class data D_i to learn Θ_i initialized by Θ_{i-1}. During learning, we use the conventional classification loss and the MCIL-specific distillation loss [16, 24]. After each phase the model is evaluated on unseen data for all classes (observed by the system so far). We report the average accuracy over all $N+1$ phases as the final evaluation, following [24, 34, 9].

Distillation Loss and Classification Loss. Distillation Loss was originally proposed in [8] and was applied to MCIL in [16, 24]. It encourages the new Θ_i and the previous Θ_{i-1} to maintain the same prediction ability on old classes. Assume there are K classes in $D_{0:i-1}$. Let x be an image in D_i. $\hat{p}_k(x)$ and $p_k(x)$ denote the prediction logits of the k-th class from Θ_{i-1} and Θ_i, respectively. The distillation loss is formulated as

$$\mathcal{L}_d(\Theta_i;\Theta_{i-1};x) = -\sum_{k=1}^{K} \hat{\pi}_k(x)\log\pi_k(x), \qquad (1a)$$

$$\hat{\pi}_k(x) = \frac{e^{\hat{p}_k(x)/\tau}}{\sum_{j=1}^{K} e^{\hat{p}_j(x)/\tau}}, \quad \pi_k(x) = \frac{e^{p_k(x)/\tau}}{\sum_{j=1}^{K} e^{p_j(x)/\tau}}, \qquad (1b)$$

where τ is a temperature scalar set to be greater than 1 to assign larger weights to smaller values.

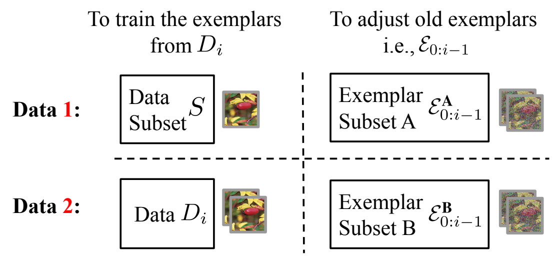
For classification loss $\mathcal{L}_c$, we use softmax cross entropy loss. Assume there are M classes in $D_{0:i}$,. The classification loss is formulated as

$$\mathcal{L}_c(\Theta_i;x) = -\sum_{k=1}^{K+M} \delta_{y=k}\log p_k(x), \qquad (2)$$

where y is the ground truth label for x, and $\delta_{y=k}$ is the indicator function.

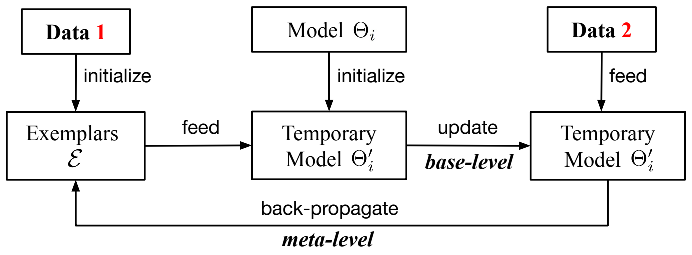
4. Mnemonics Training

As illustrated in Figure 2, the proposed *mnemonics* training alternates the learning of classification models and



To train the exemplars from D_i
To adjust old exemplars i.e., $\mathcal{E}_{0:i-1}$
Data 1:
Data Subset S
Exemplar Subset A $\mathcal{E}^{\mathbf{A}}_{0:i-1}$
Data 2:
Data D_i
Exemplar Subset B $\mathcal{E}^{\mathbf{B}}_{0:i-1}$

(a) data splits in two cases



(b) uniform computing flow

Figure 3. The proposed local BOP framework that uses a uniform computing flow in (b) to handle two cases of *exemplar-level* learning: training new class exemplars $\mathcal{E}_i$ from D_i; and adjusting old exemplars $\mathcal{E}_{0:i-1}$, with the data respectively given in (a). Note that (1) $\mathcal{E}^{\mathbf{A}}_{0:i-1}$ and $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ are used as the validation set alternately for each other when adjusting $\mathcal{E}_{0:i-1}$; (2) $\mathcal{E}$ in (b) denote the *mnemonics* exemplars which are $\mathcal{E}_i$, $\mathcal{E}^{\mathbf{A}}_{0:i-1}$, and $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ in Eq. 9, 10a and 10b, respectively.

mnemonics exemplars across all phases, where *mnemonics* exemplars are not just data samples but can be optimized and adjusted online. We formulate this alternative learning with a global *Bilevel Optimization Program (BOP)* composed of *model-level* and *exemplar-level* problems (Section 4.1), and offer the solutions in Section 4.2 and Section 4.3, respectively. We summarize the entire algorithm in Section 4.4.

4.1. Global BOP

In MCIL, the classification model is incrementally trained in each phase on the union of new class data and old class *mnemonics* exemplars. In turn, based on this model, the new class *mnemonics* exemplars (i.e., the parameters of the exemplars) are trained before omitting new class data. Therefore, the optimality of the exemplars and the model are a constraint to each other. We propose to formulate this relationship with a global BOP in which each phase uses the optimal model to optimize exemplars, and vice versa.

Specifically in the i-th phase, the general MCIL system aims to learn a model Θ_i to approximate the *ideal* one Θ_i^* which minimizes the classification loss $\mathcal{L}_c$ on D_i and $D_{0:i-1}$ (the latter not being accessible in the i-th phase), i.e,

$$\Theta_i^* = \arg\min_{\Theta_i} \mathcal{L}_c(\Theta_i; D_{0:i-1} \cup D_i). \tag{3}$$

Since $D_{0:i-1}$ was omitted and only $\mathcal{E}_{0:i-1}$ is stored in memory, we approximate $\mathcal{E}_{0:i-1}$ towards the optimal replacement of $D_{0:i-1}$ as much as possible. We formulate this with the global BOP (where "global" means operating through all phases) as:

$$\min_{\Theta_i} \mathcal{L}_c(\Theta_i; \mathcal{E}^*_{0:i-1} \cup D_i) \tag{4a}$$

$$\text{s.t. } \mathcal{E}^*_{0:i-1} = \arg\min_{\mathcal{E}_{0:i-1}} \mathcal{L}_c\big(\Theta_{i-1}(\mathcal{E}_{0:i-1}); \mathcal{E}_{0:i-2} \cup D_{i-1}\big), \tag{4b}$$

where $\Theta_{i-1}(\mathcal{E}_{0:i-1})$ denotes that Θ_{i-1} was fine-tuned on $\mathcal{E}_{0:i-1}$ to reduce the bias caused by the imbalance sample numbers between new class data D_{i-1} and old exemplars $\mathcal{E}_{0:i-2}$ in the $i-1$-th phase. Please refer to the last paragraph of Section 4.3 for more details. In the following, Problem 4a and Problem 4b are called *model-level* and *exemplar-level* problems, respectively.

4.2. Model-level problem

As illustrated in Figure 2, in the i-th phase, we first solve the *model-level* problem: *mnemonics* exemplars $\mathcal{E}_{0:i-1}$ as part of the input; and previous Θ_{i-1} as the initialization. In terms of the loss, we additionally consider the distillation loss $\mathcal{L}_d$ introduced in Eq 1. According to Problem 4, the objective function can be expressed as

$$\mathcal{L}_{\text{all}} = \lambda\mathcal{L}_c(\Theta_i; \mathcal{E}_{0:i-1} \cup D_i) + (1-\lambda)\mathcal{L}_d(\Theta_i; \Theta_{i-1}; \mathcal{E}_{0:i-1} \cup D_i), \tag{5}$$

where λ is a scalar manually set to balance two loss terms. Let α_1 be the learning rate, Θ_i is updated with gradient descent as follows,

$$\Theta_i \leftarrow \Theta_i - \alpha_1 \nabla_\Theta \mathcal{L}_{\text{all}}. \tag{6}$$

Then, Θ_i will be used to train the parameters of the *mnemonics* exemplars, i.e., to solve the *exemplar-level* problem in Section 4.3.

4.3. Exemplar-level problem

Typically, the number of exemplars (i.e., $\mathcal{E}_i$) is set to be greatly smaller than that of the original data (i.e., D_i). Existing methods [24, 9, 34, 2] are always based on the assumption that the models trained on the few exemplars also minimize its loss on the original data. However, there is no guarantee particularly when these exemplars are heuristically chosen. In contrast, our approach explicitly aims to ensure a feasible approximation of that assumption, thanks to the differentiability of our *mnemonics* exemplars.

To achieve this, we train a temporary model Θ'_i on $\mathcal{E}_i$ to maximize the prediction on D_i, for which we use D_i to compute a validation loss to penalize this temporary training with respect to the parameters of $\mathcal{E}_i$. The entire problem is thus formulated in a local BOP (where "local" means within a single phase):

$$\min_{\mathcal{E}_i} \mathcal{L}_c\big(\Theta'_i(\mathcal{E}_i); D_i\big) \tag{7a}$$

$$\text{s.t. } \Theta'_i(\mathcal{E}_i) = \arg\min_{\Theta_i} \mathcal{L}_c(\Theta_i; \mathcal{E}_i). \tag{7b}$$

We name the temporary training in Problem. 7b as *base-level* optimization and the validation in Problem. 7a as *meta-level* optimization, similar to the meta-learning naming applied to learning few-shot tasks [5].

Training $\mathcal{E}_i$. The training flow is detailed in Figure 3(b) with the data split on the left of Figure 3(a). First, the image-size parameters of $\mathcal{E}_i$ are initialized by a random sample subset S of D_i. Second, we initialize a temporary model Θ'_i by Θ_i and train Θ'_i on $\mathcal{E}_i$ (represented uniformly as $\mathcal{E}$ in 3(b)), for a few iterations by gradient descent:

$$\Theta'_i \leftarrow \Theta'_i - \alpha_2 \nabla_{\Theta'} \mathcal{L}_c(\Theta'_i; \mathcal{E}_i), \tag{8}$$

where α_2 is the learning rate to fine-tune temporary models. Finally, as the Θ'_i and $\mathcal{E}_i$ are both differentiable, we are able to compute the loss of Θ'_i on D_i (i.e., the validation loss) and back-propagate it to optimize $\mathcal{E}_i$,

$$\mathcal{E}_i \leftarrow \mathcal{E}_i - \beta_1 \nabla_{\mathcal{E}} \mathcal{L}_c\big(\Theta'_i(\mathcal{E}_i); D_i\big), \tag{9}$$

where β_1 is the learning rate. In this step, we basically need to back-propagate the validation gradients till the input layer, through unrolling all training gradients of Θ'_i. It involves a gradient through a gradient. Computationally, this operation requires an additional backward pass through $\mathcal{L}_c(\Theta'_i; \mathcal{E}_i)$ to compute Hessian-vector products, which is supported by standard numerical computation libraries like TensorFlow [1] and PyTorch [30].

Adjusting $\mathcal{E}_{0:i-1}$. The *mnemonics* exemplars of a previous class were trained when this class occurred. It is desirable to adjust them to the changing data distribution online. However, the original data $D_{0:i-1}$ is not accessible, so it is not feasible to directly apply Eq. 9. Instead, we propose to split $\mathcal{E}_{0:i-1}$ into two subsets and subject to $\mathcal{E}_{0:i-1} = \mathcal{E}^{\mathbf{A}}_{0:i-1} \cup \mathcal{E}^{\mathbf{B}}_{0:i-1}$. We use one of them, e.g. $\mathcal{E}^{\mathbf{B}}_{0:i-1}$, as the validation set (i.e., a replacement of $D_{0:i-1}$) to optimize the other one, e.g., $\mathcal{E}^{\mathbf{A}}_{0:i-1}$ as shown on the right of Figure 3(a). Alternating the input and target data in Figure 3(b), we can adjust all old exemplars in two steps:

$$\mathcal{E}^{\mathbf{A}}_{0:i-1} \leftarrow \mathcal{E}^{\mathbf{A}}_{0:i-1} - \beta_2 \nabla_{\mathcal{E}^{\mathbf{A}}} \mathcal{L}_c\big(\Theta'_i(\mathcal{E}^{\mathbf{A}}_{0:i-1}); \mathcal{E}^{\mathbf{B}}_{0:i-1}\big), \tag{10a}$$

$$\mathcal{E}^{\mathbf{B}}_{0:i-1} \leftarrow \mathcal{E}^{\mathbf{B}}_{0:i-1} - \beta_2 \nabla_{\mathcal{E}^{\mathbf{B}}} \mathcal{L}_c\big(\Theta'_i(\mathcal{E}^{\mathbf{B}}_{0:i-1}); \mathcal{E}^{\mathbf{A}}_{0:i-1}\big), \tag{10b}$$

where β_2 is the learning rate. $\Theta'_i(\mathcal{E}^{\mathbf{B}}_{0:i-1})$ and $\Theta'_i(\mathcal{E}^{\mathbf{A}}_{0:i-1})$ are trained by replacing $\mathcal{E}_i$ in Eq. 8 with $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ and $\mathcal{E}^{\mathbf{A}}_{0:i-1}$, respectively. We denote the adjusted exemplars as $\tilde{\mathcal{E}}_{0:i-1}$. Additionally, we can also split $\mathcal{E}_{0:i-1}$ into more than 2 subsets, each subset is optimized with its complement as the replacement of $D_{0:i-1}$ by the same strategy in Eq. 10.

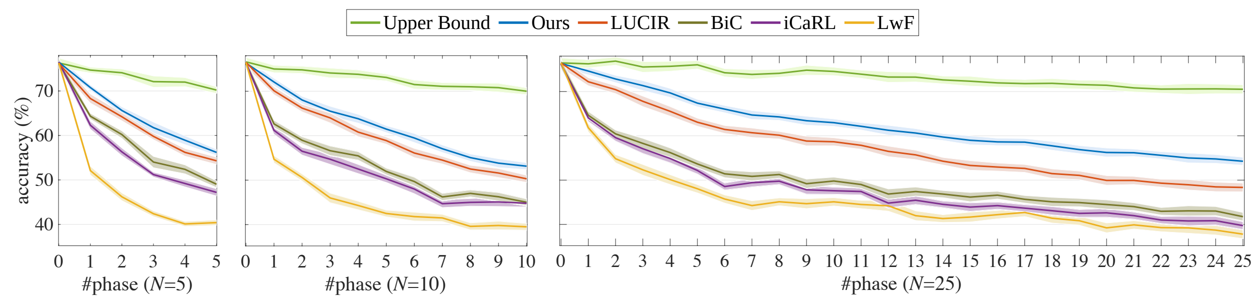
Fine-tuning models on only exemplars. The model Θ_i has been previously trained on $D_i \cup \mathcal{E}_{0:i-1}$, and may suffer from the classification bias caused by the imbalance sample numbers, e.g., 1000 *versus* 20, between the classes in D_i and $\mathcal{E}_{0:i-1}$. In order to alleviate this bias, we fine-tune Θ_i on $\mathcal{E}_i \cup \tilde{\mathcal{E}}_{0:i-1}$ in which each class has the same number of samples.

Algorithm 1: Mnemonics Training

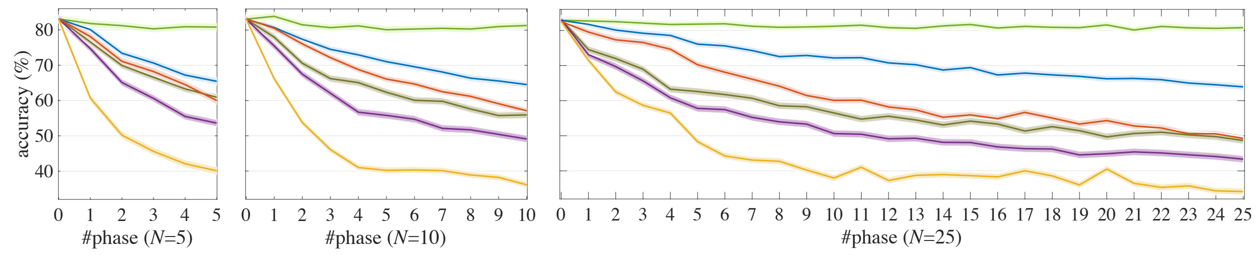
```
   Input: Data flow $\{D_i\}_{i=0}^{N}$.
   Output: MCIL models $\{\Theta_i\}_{i=0}^{N}$, and mnemonics
           exemplars $\{\mathcal{E}_i\}_{i=0}^{N}$.
 1 for $i$ in $\{0, 1, ..., N\}$ do
 2     Get $D_i$;
 3     if $i = 0$ then
 4         Randomly initialize $\Theta_0$ and train it on $D_0$;
 5     else
 6         Get $\mathcal{E}_{0:i-1}$ from memory;
 7         Initialize $\Theta_i$ with $\Theta_{i-1}$;
 8         Train $\Theta_i$ on $\mathcal{E}_{0:i-1} \cup D_i$ by Eq. 6;
 9     end
10     Sample $S$ from $D_i$ to initialize $\mathcal{E}_i$;
11     Train $\mathcal{E}_i$ using $\Theta_i$ by Eq. 9;
12     while $i \geq 1$ do
13         Split $\mathcal{E}_{0:i-1}$ into subsets $\mathcal{E}^{\mathbf{A}}_{0:i-1}$ and $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ ;
14         Optimize $\mathcal{E}^{\mathbf{A}}_{0:i-1}$ and $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ by Eq. 10;
15         Get the adjusted old exemplars $\tilde{\mathcal{E}}_{0:i-1}$
16     end
17     (Optional) delete part of the exemplars in $\tilde{\mathcal{E}}_{0:i-1}$;
18     Finetune $\Theta_i$ on $\mathcal{E}_i \cup \tilde{\mathcal{E}}_{0:i-1}$;
19     Run test and record the results;
20     Update $\mathcal{E}_{0:i} \leftarrow \mathcal{E}_i \cup \tilde{\mathcal{E}}_{0:i-1}$ in memory.
21 end
```

4.4. Algorithm

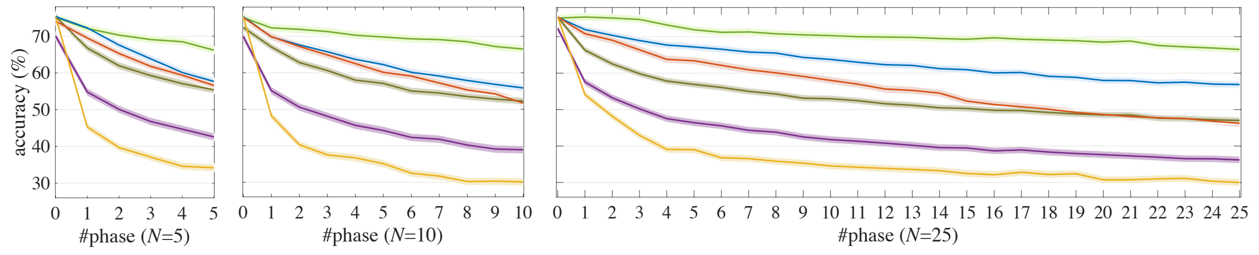
In Algorithm 1, we summarize the overall process of the proposed *mnemonics* training. Step 1-16 show the alternative learning of classification models and *mnemonics* exemplars, corresponding to Sections 4.1- 4.3. Specifically in each phase, Step 8 executes the *model-level* training, while Step 11 and 14 are the *exemplar-level*. Step 17 is optional due to different MCIL settings regarding the memory budget. We conduct experiments in two settings: (1) each class has a fixed number (e.g., 20) of exemplars, and (2) the system consistently keeps a fixed memory budget in all phases,



(a) CIFAR-100 (100 classes). In the 0-th phase, Θ_0 is trained on 50 classes, the remaining classes are given evenly in the subsequent phases.



(b) ImageNet-Subset (100 classes). In the 0-th phase, Θ_0 is trained on 50 classes, the remaining classes are given evenly in the subsequent phases.



(c) ImageNet (1000 classes). In the 0-th phase, Θ_0 on is trained on 500 classes, the remaining classes are given evenly in the subsequent phases.

Figure 4. Phase-wise accuracies (%). Light-color ribbons are visualized to show the 95% confidence intervals. Comparing methods: Upper Bound (the results of joint training with all previous data accessible in each phase); LUCIR (2019) [9]; BiC (2019) [34]; iCaRL (2017) [24]; and LwF (2016) [16]. For ours, we show our best results using "LUCIR *w/* ours". The average accuracy of each curve is given in Table 1.

therefore, the system in earlier phases can store more exemplars per class and needs to discard old exemplars in later phases gradually. Step 18 fine-tunes the model on adjusted and balanced examples. It is helpful to reduce the previous model bias (Step 8) caused by the imbalance samples numbers between new class data D_i and old exemplars $\mathcal{E}_{0:i-1}$. Step 19 is to evaluate the learned model Θ_i in the current phase, and the average over all phases will be reported as the final evaluation. Step 20 updates the memory to include new exemplars.

5. Experiments

We evaluate the proposed *mnemonics* training approach on two popular datasets (CIFAR-100 [13] and ImageNet [26]) for four different baseline architectures [16, 24, 34, 9], and achieve consistent improvements. Below we describe the datasets and implementation details (Section 5.1), followed by results and analyses (Section 5.2), including comparisons to the state-of-the-art, ablation studies and visualization results.

5.1. Datasets and implementation details

Datasets. We conduct MCIL experiments on two datasets, CIFAR-100 [13] and ImageNet [26], which are widely used in related works [24, 2, 34, 9]. **CIFAR-100** [13] contains $60,000$ samples of 32×32 color images from 100 classes. Each class has 500 training and 100 test samples. **ImageNet** (ILSVRC 2012) [26] contains around 1.3 million samples of 224×224 color images from $1,000$ classes. Each class has about $1,300$ training and 50 test samples. ImageNet is typically used in two MCIL settings [9, 24]: based on only a subset of 100 classes or the entire $1,000$ classes. The 100-class data in **ImageNet-Subeset** are randomly sampled from ImageNet with an identical random seed (1993) by NumPy, following [24, 9].

The architectures of Θ. Following the uniform setting [24,

Metric	Method	*CIFAR-100*			*ImageNet-Subset*			*ImageNet*		
		N=5	10	25	5	10	25	5	10	25
Average acc. (%) $\uparrow$ $\bar{\mathcal{A}}=\frac{1}{N+1}\sum_{i=0}^{N}\mathcal{A}_i$	LwF$^\diamond$ (2016) [16]	49.59	46.98	45.51	53.62	47.64	44.32	44.35	38.90	36.87
	LwF *w/* ours	54.43	52.67	51.75	61.23	59.24	59.71	52.70	50.37	50.79
	iCaRL (2017) [24]	57.12	52.66	48.22	65.44	59.88	52.97	51.50	46.89	43.14
	iCaRL *w/* ours	59.88	57.53	54.30	72.55	70.29	67.12	60.61	58.62	53.46
	BiC (2019) [34]	59.36	54.20	50.00	70.07	64.96	57.73	62.65	58.72	53.47
	BiC *w/* ours	60.67	58.11	55.51	73.16	71.37	68.41	64.63	62.71	60.20
	LUCIR (2019) [9]	63.17	60.14	57.54	70.84	68.32	61.44	64.45	61.57	56.56
	LUCIR *w/* ours	**64.95**	**63.25**	**63.70**	**73.30**	**72.17**	**71.50**	**66.15**	**63.12**	**63.08**
Forgetting rate (%) $\downarrow$ $\mathcal{F}=\mathcal{A}_N^Z-\mathcal{A}_0^Z$	LwF$^\diamond$ (2016) [16]	43.36	43.58	41.66	55.32	57.00	55.12	48.70	47.94	49.84
	LwF *w/* ours	38.38	36.66	33.50	39.56	40.44	39.99	37.46	38.42	37.95
	iCaRL (2017) [24]	31.88	34.10	36.48	43.40	45.84	47.60	26.03	33.76	38.80
	iCaRL *w/* ours	25.28	27.02	28.22	20.00	24.36	29.32	20.26	24.04	17.49
	BiC (2019) [34]	31.42	32.50	34.60	27.04	31.04	37.88	25.06	28.34	33.17
	BiC *w/* ours	22.42	24.50	25.52	14.52	17.40	23.96	18.32	19.72	20.50
	LUCIR (2019) [9]	18.70	21.34	26.46	31.88	33.48	35.40	24.08	27.29	30.30
	LUCIR *w/* ours	**11.64**	**10.90**	**9.96**	**10.20**	**9.88**	**11.76**	**13.63**	**13.45**	**14.40**

$^\diamond$ Using *herding* exemplars as [9, 24, 34] for fair comparison.

Table 1. Average accuracies $\bar{\mathcal{A}}$ (%) and forgetting rates $\mathcal{F}$ (%) for the state-of-the-art [9] and other baseline architectures [16, 24, 34] with and without our *mnemonics* training approach as a plug-in module. Let D_i^{test} be the test data corresponding to D_i in the i-th phase. $\mathcal{A}_i$ denotes the average accuracy of $D_{0:i}^{\text{test}}$ by Θ_i. $\mathcal{A}_i^Z$ is the average accuracy of D_0^{test} by Θ_i in the i-th phase. Note that the weight transfer operations are applied in "*w/* ours" methods.

34, 9], we use a 32-layer ResNet [7] for CIFAR-100 and an 18-layer ResNet for ImageNet. We deploy the weight transfer operations [31, 22] to train the network, rather than using standard weight over-writing. This helps to reduce *forgetting* between adjacent models (i.e., Θ_{i-1} and Θ_i). Detailed formulations are in the supplementary.

The architecture of $\mathcal{E}$. It depends on the size of image and the number of exemplars we need. On CIFAR-100, each *mnemonics* exemplar is a $32\times32\times3$ tensor. On ImageNet, it is a $224\times224\times3$ tensor. The number of exemplars is set in two manners [9]. (1) 20 samples are uniformly used for every class. So the parameter size of the exemplars per class is equal to tensor$\times20$. *This setting is used in the main paper*. (2) The system keeps a fixed memory budget, e.g. at most $2,000$ exemplars in total, in all phases. It thus saves more exemplars per class in earlier phases and discard old exemplars afterwards. *Results are given in the supplementary due to page limits*. In both settings, we have the consistent finding that *mnemonics* training is the most efficient approach, surpassing the state-of-the-art by large margins.

***Model-level* hyperparameters.** The SGD optimizer is used to train Θ. Momentum and weight decay parameters are set to 0.9 and 0.0005, respectively. In each (i.e. i-th) phase, the learning rate α_1 is initialized as 0.1. On the CIFAR-100 (ImageNet), Θ_i is trained in 160 (90) epochs for which α_1 is reduced to its $\frac{1}{10}$ after 80 (30) and then 120 (60) epochs. In Eq. 5, scalar λ and temperature τ are set to 0.5 and 2, respectively, following [24, 9].

***Exemplar-level* hyperparameters.** An SGD optimizer is used to update *mnemonics* exemplars $\mathcal{E}_i$ and adjust $\mathcal{E}_{0:i-1}$ (as in Eq. 9 and Eq. 10 respectively) in 50 epochs. In each phase, the learning rates β_1 and β_2 are initialized as 0.01 uniformly and reduced to their half each 10 epochs. Gradient descent is applied to update the temporary model Θ' in 50 epochs (as in Eq. 8). The learning rate α_2 is set to 0.01. Note that the same hyperparameters with the temporary model updating are applied when fine-tuning Θ_i on $\mathcal{E}_i\cup\tilde{\mathcal{E}}_{0:i-1}$ before testing in each phase.

Benchmark protocol. This work uses the protocol given in the the most recent work — LUCIR [9]. For fair comparison, we implement all other methods [24, 2, 34] on the same protocol. Given a dataset, Θ_0 is firstly trained on half of the classes. Then the remaining classes are evenly learned by the model in the subsequent phases. An MCIL system has one initial phase and N incremental phases. The total number of incremental phases N is set to be 5, 10 or 25 (denoted as "N-phase" setting). At the end of each phase, the model Θ_i is evaluated on the test data $D_{0:i}^{\text{test}}$ where "$0:i$" denote all seen classes so far. The average accuracy $\bar{A}$ (over all phases) is reported as the final evaluation [24, 9]. In addition, we propose a forgetting rate, denoted as $\mathcal{F}$, by calculating the difference between the accuracies of Θ_0 and Θ_N on the same initial test data D_0^{test}.

5.2. Results and analyses

Exemplar		*CIFAR-100*			*ImagNet-Subset*		
		N=5	10	25	5	10	25
random w/o adj.		63.06	62.30	62.06	71.34	70.02	68.24
random		63.51	62.47	61.59	71.67	70.31	68.02
herding w/o adj.		63.39	61.50	60.95	71.22	69.67	67.45
herding	↑	63.56	61.79	61.05	72.01	70.02	68.00
ours *w/o* adj.		63.97	62.34	62.31	72.45	70.57	70.78
ours		**64.95**	**63.26**	**63.70**	**73.30**	**72.17**	**71.50**
random w/o adj.		19.38	15.90	13.91	21.67	17.89	16.38
random		17.24	16.01	13.23	17.05	15.76	13.27
herding w/o adj.	↓	21.02	21.18	20.76	21.53	18.15	17.96
herding		17.02	19.76	16.87	21.93	16.32	15.91
ours *w/o* adj.		13.78	12.35	10.65	20.76	16.47	12.68
ours		**11.64**	**10.90**	**9.96**	**10.20**	**9.88**	**11.76**

Table 2. Ablation study. The top and the bottom blocks present average accuracies $\bar{\mathcal{A}}$ (%) and forgetting rates $\mathcal{F}$ (%), respectively. "*w/o* adj." means without old exemplar adjustment. Note that the weight transfer operations are applied in all these experiments.

Table 1 shows the comparisons with the state-of-the-art [9] and other baseline architectures [16, 24, 34] with and without our *mnemonics* training as plug-in module. Note that "without" in [16, 24, 34, 9] means using *herding* exemplars (we add *herding* exemplars to [16] for fair comparison). Figure 4 particularly shows the phase-wise results of our best model, i.e. LUCIR [9] *w/* ours, and the baselines. Table 2 shows the ablation study for evaluating two key components: training *mnemonics* exemplars; and adjusting old *mnemonics* exemplars. Figure 5 visualizes the differences between *herding* and *mnemonics* exemplars in the data space.

Compared to the state-of-the-art. Table 1 shows that taking our *mnemonics* training as a plug-in module on the state-of-the-art [9] and other baseline architectures consistently improves their performance. In particular, LUCIR [9] *w/* ours achieves the highest average accuracy and lowest forgetting rate, e.g. respectively 63.08% and 14.40% on the most challenging 25-phase ImageNet. The overview on forgetting rates $\mathcal{F}$ reveals that our approach is greatly helpful to reduce forgetting problems for every method. For example, LUCIR (*w/* ours) sees its $\mathcal{F}$ reduced to around the third and the half on the 25-phase CIFAR-100 and ImageNet, respectively.

Different total phases (N **= 5, 10, 25).** Table 1 and Figure 4 demonstrate that the boost by our *mnemonics* training becomes larger in more-phase settings, e.g. on CIFAR-100, LUCIR *w/* ours gains 1.78% on 5-phase while 6.16% on 25-phase. When checking the ending points of the curves from $N = 5$ to $N = 25$ in Figure 4, we find related methods, LUCIR, BiC, iCaRL and LwF, all suffer from the performance drop. The possible reason is that their models get more and more seriously overfitted to the *herding* exemplars which are heuristically chosen and fixed. In contrast, our best model (LUCIR *w/* ours) does not have such problem, thanks for our *mnemonics* exemplars being given both *strong optimizability* and *flexible adaptation ability*. In particular, its ending point on $N = 25$ (56.52%) goes even higher than that on $N = 5$ (56.19%) on the CIFAR-100.

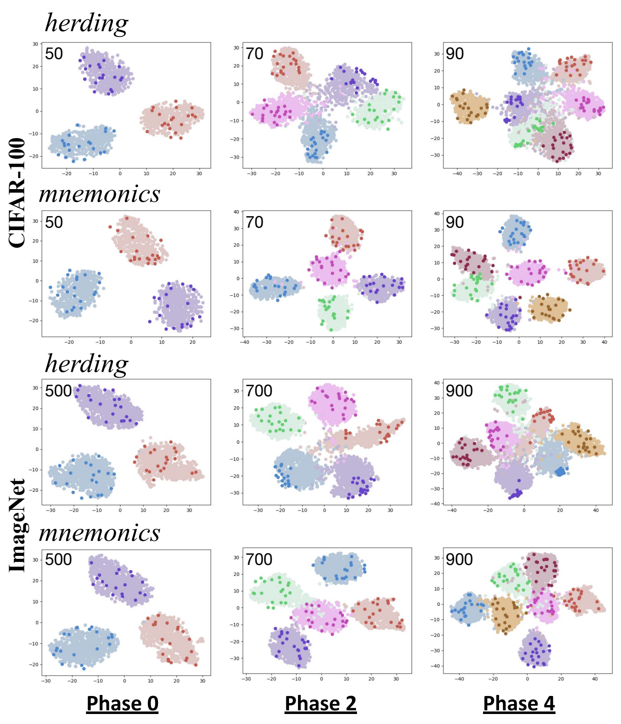


Figure 5. The t-SNE [17] results of *herding* and our *mnemonics* on two datasets. $N = 5$. In each colored class, deep-color points are exemplars, and light-color ones show the original data as reference of the real data distribution. Total number of classes (used in real training) is given in the top-left corner of the sub-figure. For a clear visualization, Phase-0 randomly picks 3 classes from 50 (500) classes on CIFAR-100 (ImageNet). Phase-2 and Phase-4 increases to 5 and 7 classes, respectively.

Ablation study. Table 2 concerns four ablative settings and compares the efficiencies between our *mnemonics* training approach (*w/* and *w/o* adjusting old exemplars) and two baselines: *random* and *herding* exemplars. Concretely, our approach achieves the highest average accuracies and the lowest forgetting rates in all settings. Our online operation of adjusting old exemplars derives consistent improvements, i.e., average 1% on both datasets, even though without the original data. In terms of forgetting rates, our results are the lowest (best). It is interesting that *random* achieves lower (better) performance than *herding*. *Random* actually selects exemplars both on the center and boundary of the data space (of each class), but *herding* considers the center only which strongly relies on the current data distribution and is thus weak to take any risk of change in the subsequent phase. This weakness is further verified in the visualization of exemplars. We also supply more ablative results on other components (e.g. distillation loss and transferring weights). These results are given in the supplementary.

Visualization results. Figure 5 demonstrates the t-SNE results for both *herding* [24, 9, 34, 2] and *mnemonics* exemplars (deep-colored) in the data space (light-colored). We have two main observations. (1) Our *mnemonics* approach results in much clearer separation in the data than *herding*. (2) Our *mnemonics* exemplars are optimized to mostly locate on the boundaries between classes, which is essential to derive high-quality classifiers. Comparing the Phase-4 results of two datasets, we can see that learning more classes (i.e., on the ImageNet) clearly causes more confusion among classes in the data space, while our approach still yields strong intra compactness and inter separation, as shown in the rightmost bottom sub-figure. Besides, the changes of average distances between exemplars and initial samples are provided in the supplementary to demonstrate the evolution of exemplars.

6. Conclusions

In this paper, we develop a novel *mnemonics* training framework for tackling multi-class incremental learning tasks. Our main contribution is the *mnemonics* exemplars which are not only efficient data samples but also flexible, optimizable and adaptable parameters contributing a lot to the flexibility of online systems. Quite intriguingly, our *mnemonics* training approach is *generic* that it can be easily applied to existing methods to achieve large-margin improvements. Extensive experimental results on four different baseline architectures validate the high efficiency of our approach, and the in-depth visualization reveals the essential reason is that our *mnemonics* exemplars are automatically learned to be the optimal replacement of the original data, which can yield high-quality classification models.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv*, 1603.04467, 2016. 5

[2] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018. 1, 2, 3, 4, 6, 7, 9

[3] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *NIPS*, 2001. 2

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 2

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 3, 5

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

[8] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv*, 1503.02531, 2015. 2, 3

[9] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, 2019. 1, 2, 3, 4, 6, 7, 8, 9

[10] 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. 2

[11] Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv*, 1710.10368, 2017. 2

[12] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2018. 1

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[14] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From N to N+1: multiclass transfer incremental learning. In *CVPR*, 2013. 2

[15] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. In *NeurIPS*, 2019. 3

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 1, 2, 3, 6, 7, 8

[17] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 1, 8

[18] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *ICLR*, 2019. 2, 3

[19] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning

problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[20] K. McRae and P. Hetherington. Catastrophic interference is eliminated in pre-trained networks. In *CogSci*, 1993. 1

[21] Thomas Mensink, Jakob J. Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013. 2

[22] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 7

[23] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. 97:285–308, 1990. 1

[24] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017. 1, 2, 3, 4, 6, 7, 8, 9

[25] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019. 2

[26] 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. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[27] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NIPS*, 2017. 1, 2

[28] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2018. 2

[29] Heinrich von Stackelberg et al. Theory of the market economy. 1952. 2

[30] Benoit Steiner, Zachary DeVito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 5

[31] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019. 3, 7

[32] Ragav Venkatesan, Hemanth Venkateswara, Sethuraman Panchanathan, and Baoxin Li. A strategy for an uncompromising incremental learner. *arXiv*, 1705.00744, 2017. 2

[33] Max Welling. Herding dynamical weights to learn. In *ICML*, 2009. 2

[34] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019. 1, 2, 3, 4, 6, 7, 8, 9

[35] Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In *CVPR*, 2019. 3