Mnemonics Training: Multi-Class Incremental Learning without Forgetting

Yaoyao Liu$^{1,2*}$ Yuting Su$^1$† An-An Liu$^1$† Bernt Schiele$^2$ Qianru Sun$^3$

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

$\{\text{yaoyao.liu, schiele, qsun}\}@\text{mpi-inf.mpg.de}$

$\{\text{liuyaoyao, ytsu, liuanan}\}@\text{tju.edu.cn}$  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 different 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 [12, 20, 21, 24, 28]. 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 [2, 9, 16, 17, 25, 36]. Rebuffi et al. [25] 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 authors.
$^1$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 [35]) [2, 9, 25, 36], but turn out to be not particularly effective. For example, iCaRL [25] 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 becomes weaker 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. Using 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 using a bilevel optimization program (BOP) [19, 29] 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 baselines, e.g., 20% and 6.5% higher than herding-based iCaRL [25] and LUCIR [9], respectively, in the 25-phase setting on the ImageNet [25].

Our contributions include: (1) A novel mnemonics training framework that alternates the learning of exemplars and models in a global bilevel optimization program, where bilevel includes 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, 14, 22]. 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) [4, 10, 17, 26, 28], or in the multi-class setting (classes from the identical dataset) [2, 9, 25, 36]. 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. [17]. Rebuffi et al. [25] combined this idea with representation learning, for which a handful of herding exemplars are stored for replaying old knowledge. Herding [35] picks the nearest neighbors of the average sample per class [25]. 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. [36] 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 [11, 28, 34] 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 [30] 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-
Fig. 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 Fig. 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, 15, 32, 33, 37, 38] is another BOP in which a meta-learner is optimized subject to the optimality of the base-learner. Recently, MacKay et al. [19] 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 [25] to evaluate classification models incrementally learned using a sequence of data from different classes. Its uniform setting is used in related works [2, 9, 25, 36]. 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 the network training.

Denotations. Assume there are \( N + 1 \) phases (i.e., 1 initial phase and \( N \) incremental phases) in the MCIL system. In the initial (the 0-th) phase, we learn the model \( \Theta_0 \) on data \( D_0 \) using a conventional classification loss, e.g., cross-entropy loss, and then save \( \Theta_0 \) to the memory of the system. Due to the memory limitation, we cannot 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 \( \Theta_{i-1} \) 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 train \( \Theta_i \) initialized by \( \Theta_{i-1} \). During training, we use a classification loss and an MCIL-specific distillation loss [17, 25]. 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 [9, 25, 36].

Distillation Loss and Classification Loss. Distillation Loss was originally proposed in [8] and was applied to MCIL in [17, 25]. It encourages the new \( \Theta_i \) and previous \( \Theta_{i-1} \) 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 \( \Theta_{i-1} \) and \( \Theta_i \), respectively. The distillation loss is formulated as

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

\[
\hat{\pi}_k(x) = \frac{\exp(\hat{p}_k(x)/\tau)}{\sum_{j=1}^{K} \exp(\hat{p}_j(x)/\tau)}, \quad \pi_k(x) = \frac{\exp(p_k(x)/\tau)}{\sum_{j=1}^{K} \exp(p_j(x)/\tau)}, \tag{1b}
\]

where \( \tau \) is a temperature scalar set to be greater than 1 to assign larger weights to smaller values.

We use the softmax cross entropy loss as the Classification Loss \( \mathcal{L}_c \). Assume there are \( M \) classes in \( D_{0:i} \). This loss is formulated as

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

where \( y \) is the ground truth label of \( x \), and \( \delta_{y=k} \) is an indicator function.

4. Mnemonics Training

As illustrated in Fig. 2, the proposed mnemonics training alternates the learning of classification models and
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 provide the solutions in Section 4.2 and Section 4.3, respectively.

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. In this way, the optimality of model derives a constrain to optimizing the exemplars, and vice versa. 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, our MCIL system aims to learn a model $\Theta_i$ to approximate the ideal one named $\Theta_i^*$ which minimizes the classification loss $L_c$ on both $D_i$ and $D_{0:i-1}$, i.e.,

$$\Theta_i^* = \arg \min_{\Theta_i} L_c(\Theta_i; D_{0:i-1} \cup D_i).$$

Since $D_{0:i-1}$ was omitted (i.e., not accessible) and only $E_{0:i-1}$ is stored in memory, we approximate $E_{0:i-1}$ 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 follows,

$$\min_{\Theta_i} L_c(\Theta_i; E_{0:i-1} \cup D_i)$$

s.t. $E_{0:i-1} = \arg \min_{E_{0:i-1}} L_c(\Theta_{i-1}(E_{0:i-1}); E_{0:i-2} \cup D_{i-1})$,  

(4b)

where $\Theta_{i-1}(E_{0:i-1})$ denotes that $\Theta_{i-1}$ was fine-tuned on $E_{0:i-1}$ to reduce the bias caused by the imbalanced sample numbers between new class data $D_{i-1}$ and old exemplars $E_{0:i-2}$, in the $i-1$-th phase. Please refer to the last paragraph in Section 4.3 for more details. In the following paper, 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 with the mnemonics exemplars $E_{0:i-1}$ as part of the input and previous $\Theta_{i-1}$ as the model initialization. According to Problem 4, the objective function can be expressed as

$$L_{all} = \lambda L_c(\Theta_i; E_{0:i-1} \cup D_i) + (1-\lambda) L_d(\Theta_i; \Theta_{i-1}; E_{0:i-1} \cup D_i),$$

(5)

where $\lambda$ is a scalar manually set to balance between $L_d$ and $L_c$ (introduced in Section 3). Let $\alpha_1$ be the learning rate, $\Theta_i$ is updated with gradient descent as follows,

$$\Theta_i \leftarrow \Theta_i - \alpha_1 \nabla_{\Theta_i} L_{all}.$$  

Then, $\Theta_i$ 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 $E_i$ is set to be greatly smaller than that of the original data $D_i$. Existing methods [2, 9, 25, 36] 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 $\Theta_i'$ on $E_i$ to maximize the prediction on $D_i$, for which we use $D_i$ to compute a validation loss to penalize this temporary train-
ing 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, as

$$\min_{\mathbf{E}_i} \mathcal{L}_c(\Theta'_i(\mathbf{E}_i); D_i)$$

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

(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 naming in meta-learning applied to tackling 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 $\Theta'_i$ using $\Theta_i$ and train $\Theta'_i$ on $\mathbf{E}_i$ (denoted uniformly as $\mathcal{E}$ in 3(b)), for a few iterations by gradient descent:

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

(8)

where $\alpha_2$ is the learning rate of fine-tuning temporary models. Finally, as the $\Theta'_i$ and $\mathbf{E}_i$ are both differentiable, we are able to compute the loss of $\Theta'_i$ on $D_i$, and back-propagate this validation loss to optimize $\mathbf{E}_i$,

$$\mathbf{E}_i \leftarrow \mathbf{E}_i - \beta_1 \nabla_{\mathbf{E}_i} \mathcal{L}_c(\Theta'_i; \mathbf{E}_i),$$

(9)

where $\beta_1$ 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 $\Theta'_i$. This operation involves a gradient through a gradient. Computationally, it requires an additional backward pass through $\mathcal{L}_c(\Theta'_i; \mathbf{E}_i)$ to compute Hessian-vector products, which is supported by standard weight overwritten computation libraries such as TensorFlow [1] and PyTorch [31].

**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, old class data $D_{0:i-1}$ are 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 $\mathcal{E}_{0:i-1} = \mathcal{E}^A_{0:i-1} \cup \mathcal{E}^B_{0:i-1}$. We use one of them, e.g. $\mathcal{E}^B_{0:i-1}$, as the validation set (i.e., a replacement of $D_{0:i-1}$) to optimize the other one, e.g., $\mathcal{E}^A_{0:i-1}$, as shown on the right of Figure 3(a). Alternating the input and target data in Figure 3(b), we adjust all old exemplars in two steps:

$$\mathcal{E}^A_{0:i-1} \leftarrow \mathcal{E}^A_{0:i-1} - \beta_2 \nabla_{\mathcal{E}_i} \mathcal{L}_c(\Theta''^A(\mathcal{E}^A_{0:i-1}); \mathcal{E}^B_{0:i-1}),$$

(10a)

$$\mathcal{E}^B_{0:i-1} \leftarrow \mathcal{E}^B_{0:i-1} - \beta_2 \nabla_{\mathcal{E}_i} \mathcal{L}_c(\Theta''^B(\mathcal{E}^B_{0:i-1}); \mathcal{E}^A_{0:i-1}),$$

(10b)

where $\beta_2$ is the learning rate. $\Theta''^A(\mathcal{E}^A_{0:i-1})$ and $\Theta''^B(\mathcal{E}^B_{0:i-1})$ are trained by replacing $\mathbf{E}_i$ in Eq. 8 with $\mathcal{E}^A_{0:i-1}$ and $\mathcal{E}^A_{0:i-1}$, respectively. We denote the adjusted exemplars as $\mathcal{E}'_{0:i-1}$. Note that we can also split $\mathcal{E}_{0:i-1}$ into more than 2 subsets, and optimize each subset using its complement as the validation data, following the same strategy in Eq. 10.

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

**5. Experiments**

We evaluate the proposed mnemonics training approach on two popular datasets (CIFAR-100 [13] and ImageNet [27]) for four different baseline architectures [9, 17, 25, 36], 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 [27], which are widely used in related works [2, 9, 25, 36]. 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) [27] 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, 25]: one based on only a subset of 100 classes and the other based on the entire 1,000 classes. The 100-class data in ImageNet-Sub100 is randomly sampled from ImageNet with an identical random seed (1993) by NumPy, following [9, 25].

**The architectures of $\Theta$.** Following the uniform setting [9, 25, 36], we use a 32-layer ResNet [7] for CIFAR-100 and an 18-layer ResNet for ImageNet. We deploy the weight transfer operations [23, 33] to train the network, rather than using standard weight over-writing. This helps to reduce forgetting between adjacent models (i.e., $\mathcal{E}_{i-1}$ and $\Theta_i$). Please refer to the supplementary document for the detailed formulation of weight transfer.

**The architecture of $\mathcal{E}$.** It depends on the size of image and the number of exemplars we need. On the CIFAR-100, each mnemonics exemplar is a 32 × 32 × 3 tensor. On the ImageNet, it is a 224 × 224 × 3 tensor. The number of exemplars is set in two manners [9]. (1) 20 samples are uniformly used for every class. Therefore, the parameter size of the exemplars per class is equal to tensor×20. 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.
(a) CIFAR-100 (100 classes). In the 0-th phase, $\Theta_0$ 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, $\Theta_0$ is trained on 50 classes, the remaining classes are given evenly in the subsequent phases.

(c) ImageNet (1000 classes). In the 0-th phase, $\Theta_0$ 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) [36]; iCaRL (2017) [25]; and LwF (2016) [17]. We show Ours results using “LUCIR w/ ours”. Please refer to the average accuracy of each curve in Table 1.

It thus saves more exemplars per class in earlier phases and discard old exemplars afterwards. Due to page limits, the results in this setting are presented in the supplementary document. 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 with little computational or parametrization overheads.

**Model-level hyperparameters.** The SGD optimizer is used to train $\Theta$. Momentum and weight decay parameters are set to 0.9 and 0.0005, respectively. In each (i.e. $i$-th) phase, the learning rate $\alpha_i$ is initialized as 0.1. On the CIFAR-100 (ImageNet), $\Theta_i$ is trained in 160 (90) epochs for which $\alpha_i$ is reduced to its $\frac{1}{10}$ after 80 (30) and then 120 (60) epochs. In Eq. 5, the scalar $\lambda$ and temperature $\tau$ are set to 0.5 and 2, respectively, following [9, 25].

**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 $\beta_1$ and $\beta_2$ are initialized as 0.01 uniformly and reduced to their half after every 10 epochs. Gradient descent is applied to update the temporary model $\Theta'$ in 50 epochs (as in Eq. 8). The learning rate $\alpha_2$ is set to 0.01. We deploy the same set of hyperparameters for fine-tuning $\Theta_i$ on $\mathcal{E}_i \cup \tilde{\mathcal{E}}_{0:i-1}$.

**Benchmark protocol.** This work follows the protocol in the most recent work — LUCIR [9]. We also implement all other methods [2, 25, 36] on this protocol for fair comparison. Given a dataset, the model ($\Theta_0$) is firstly trained on half of the classes. Then, the model ($\Theta_i$) learns the remaining classes evenly in the subsequent phases. Assume an MCIL system has 1 initial phase and $N$ incremental phases. The total number of incremental phases $N$ is set to be 5, 10 or 25 (for each the setting is called “$N$-phase” setting). At the end of each individual phase, the learned $\Theta_i$ is evaluated on the test data $D_{0:i}^{test}$ where “$0 : i$” denote all seen classes so far. The average accuracy $\bar{A}$ (over all phases) is reported.
Table 1. Average accuracies $\bar{\alpha}$ (%) and forgetting rates $\mathcal{F}$ (%) for the state-of-the-art [9] and other baseline architectures [17, 25, 36] with and without our mnemonics training approach as a plug-in module. Let $D_{\text{test}}^i$ be the test data corresponding to $D_i$ in the $i$-th phase. $\mathcal{A}_i$ denotes the average accuracy of $D_{\text{test}}^i$ by $\Theta_i$. $\mathcal{A}_N^i$ is the average accuracy of $D_{\text{test}}^i$ by $\Theta_i$ in the $i$-th phase. Note that the weight transfer operations are applied in “w/ ours” methods.

\begin{table}
\begin{tabular}{lccccccccc}
\hline
Metric & Method & CIFAR-100 & & & ImageNet-Subset & & & ImageNet & \\
& & N=5 & 10 & 25 & & & 5 & 10 & 25 & \\
\hline
\multirow{2}{*}{Average acc. (%) $\uparrow$} & LwF (2016) [17] & 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 & \\
\multirow{2}{*}{Forgetting rate (%) $\downarrow$} & iCaRL (2017) [25] & 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) [36] & 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 & \\
\hline
\end{tabular}
\end{table}

\begin{itemize}
\item [\textsuperscript{o}] Using herding exemplars as [9, 25, 36] for fair comparison.
\end{itemize}

as the final evaluation [9, 25]. In addition, we propose a forgetting rate, denoted as $\mathcal{F}$, by calculating the difference between the accuracies of $\Theta_0$ and $\Theta_N$ on the same initial test data $D_{\text{test}}^0$. The lower forgetting rate is better.

5.2. Results and analyses

Table 1 shows the comparisons with the state-of-the-art [9] and other baseline architectures [17, 25, 36], with and without our mnemonics training approach as a plug-in module. Note that “without” in [9, 17, 25, 36] means using herding exemplars (we add herding exemplars to [17] for fair comparison). Figure 4 shows the phase-wise results of our best model, i.e., LUCIR [9] w/ ours, and those of the baselines. Table 2 demonstrates 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 1.75\% 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 performance drop. The possible reason is that their models get more and more seriously overfitted to 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 through BOP. In particular, its ending point on $N=25$ (56.52\%) goes even higher than that on $N=5$ (56.19\%) on the CIFAR-100.

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. Dynamically adjusting old exemplars brings consistent improvements, i.e., average 1\% on both datasets. In terms of forgetting rates, our results are the lowest (best). It is interesting that random
achieves lower (better) performance than herding. Random selects exemplars both on the center and boundary of the data space (for each class), but herding considers the center data only which strongly relies on the data distribution in the current phase but can not take any risk of distribution change in subsequent phases. This weakness is further revealed through the visualization of exemplars in the data space, e.g., in Figure 5. Note that the weight transfer operations are applied in all these experiments.

**Visualization results.** Figure 5 demonstrates the t-SNE results for herding (deep-colored) and our 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 yield high-quality classifiers. Comparing the Phase-4 results of two datasets (i.e., among the sub-figures on the rightmost column), 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 is able to yield stronger intra-class compactness and inter-class separation. In the supplementary, we present more visualization figures about the changes during the mnemonics training from initial examples to learned 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.

**Acknowledgments**

We would like to thank all reviewers for their constructive comments. This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant, Max Planck Institute for Informatics, the National Natural Science Foundation of China (61772359, 61572356, 61872267), the grant of Tianjin New Generation Artificial Intelligence Major Program (19ZXZNGX00110, 18ZXZNGX00150), the Open Project Program of the State Key Lab of CAD and CG, Zhejiang University (Grant No. A2005), the grant of Elite Scholar Program of Tianjin University (2019XRX-0035).

<table>
<thead>
<tr>
<th>Exemplar</th>
<th>CIFAR-100</th>
<th>ImagNet-Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N=5</td>
<td>10</td>
</tr>
<tr>
<td>random w/o adj.</td>
<td>63.06</td>
<td>62.30</td>
</tr>
<tr>
<td>random</td>
<td>63.51</td>
<td>62.47</td>
</tr>
<tr>
<td>herding w/o adj.</td>
<td>63.39</td>
<td>61.50</td>
</tr>
<tr>
<td>herding</td>
<td>63.56</td>
<td>61.79</td>
</tr>
<tr>
<td>ours w/o adj.</td>
<td>63.97</td>
<td>62.34</td>
</tr>
<tr>
<td>ours</td>
<td>64.95</td>
<td>63.26</td>
</tr>
<tr>
<td>random w/o adj.</td>
<td>19.38</td>
<td>15.90</td>
</tr>
<tr>
<td>random</td>
<td>17.24</td>
<td>16.01</td>
</tr>
<tr>
<td>herding w/o adj.</td>
<td>21.02</td>
<td>21.18</td>
</tr>
<tr>
<td>herding</td>
<td>17.02</td>
<td>19.76</td>
</tr>
<tr>
<td>ours w/o adj.</td>
<td>13.78</td>
<td>12.35</td>
</tr>
<tr>
<td>ours</td>
<td>11.64</td>
<td>10.90</td>
</tr>
</tbody>
</table>

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

Figure 5. The t-SNE [18] results of herding and our mnemonics exemplars on two datasets. $N=5$. In each colored class, deep-color points are exemplars, and light-color ones are original data referring to the real data distribution. The total number of classes (used in the training) is given in the top-left corner of each sub-figure. For 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.
References


[34] Ragav Venkatesan, Hemanth Venkateswara, Sethuraman Panchanathan, and Baoxin Li. A strategy for an uncomromising incremental learner. arXiv, 1705.00744, 2017. 2
[36] 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. 1, 2, 3, 4, 5, 6, 7