Continual Recognition with Adaptive Memory Update

XUANRONG YAO, XIN WANG, YUE LIU, and WENWU ZHU, Tsinghua University

Class incremental continual learning aims to improve the ability of modern classification models to continually recognize new classes without forgetting the previous ones. Prior art in the field has largely considered using a replay buffer. In this article, we start from an observation that the existing replay-based method would fail when the stored exemplars are not hard enough to get a good decision boundary between a previously learned class and a new class. To prevent this situation, we propose a method from the perspective of remedy after forgetting for the first time. In the proposed method, a set of exemplars is preserved as a working memory, which helps to recognize new classes. When the working memory is insufficient to distinguish between new classes, more discriminating samples would be swapped from a long-term memory, which is built up during the early training process, in an adaptive way. Our continual recognition model with adaptive memory update is capable of overcoming the problem of catastrophic forgetting with various new classes coming in sequence, especially for similar but different classes. Extensive experiments on different real-world datasets demonstrate that the proposed model is superior to existing state-of-the-art algorithms. Moreover, our model can be used as a general plugin for any replay-based continual learning algorithm to further improve their performance.

CCS Concepts: • **Computing methodologies** → **Lifelong machine learning**; *Image representations*; *Supervised learning by classification;*

Additional Key Words and Phrases: Continual learning, lifelong, incremental, replay

ACM Reference format:
Xuanrong Yao, Xin Wang, Yue Liu, and Wenwu Zhu. 2023. Continual Recognition with Adaptive Memory Update. *ACM Trans. Multimedia Comput. Commun. Appl.* 19, 3s, Article 134 (February 2023), 15 pages.
https://doi.org/10.1145/3573202

1 INTRODUCTION

The ultimate goal of Artificial Intelligence, particularly machine learning, is to empower machines with the abilities of humans. This work focuses on imitating humans' ability to continuously learn from experience—that is, *lifelong learning*. Humans can utilize the previously learned skills to accomplish new tasks and memorize the acquired knowledge as the basis for future learning without forgetting previous skills. However, current machine learning algorithms perform far less competitively than humans in this aspect, which has prompted fast-growing research on continual

This work was supported in part by the National Key Research and Development Program of China (no. 2020AAA0106300) and the National Natural Science Foundation of China (nos. 62250008, 62222209, and 62102222).
Authors' address: X. Yao, X. Wang (corresponding author), Y. Liu, and W. Zhu (corresponding author), Department of Computer Science and Technology, Tsinghua University, 30 Shuangqing Road, Beijing, China; emails: yaoxr17@mails.tsinghua.edu.cn, xin_wang@tsinghua.edu.cn, liuyuethu@163.com, wwzhu@tsinghua.edu.cn.

1551-6857/2023/02-ART134 $15.00
https://doi.org/10.1145/3573202

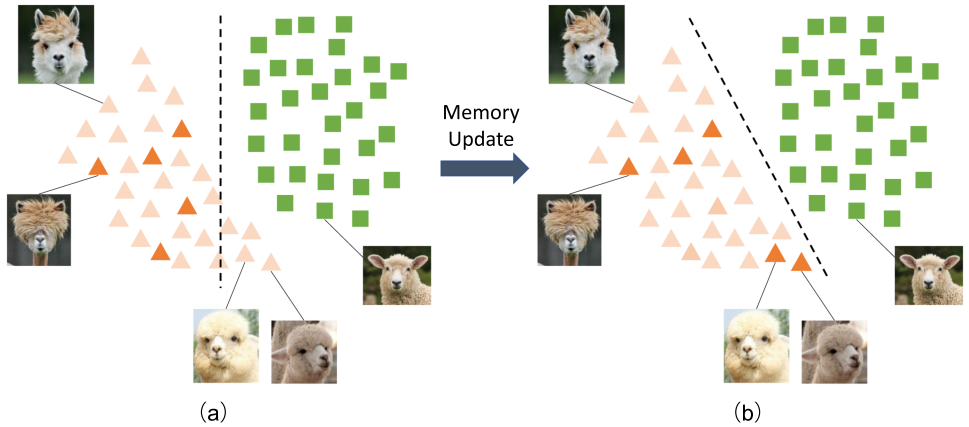

Fig. 1. Motivation of adaptive memory update. The triangles represent the samples from a previously learned class (alpaca), and the rectangles represent a new class (sheep). The dashed line is the learned decision boundary. (a) The stored exemplars (dark triangles) are not hard enough to obtain a good decision boundary, and thus some alpacas will be misclassified as sheep. (b) After some exemplars were replaced with samples more similar to sheep (harder exemplars), we can obtain more discriminative decision boundary.

recognition. Continual learning [8] aims to learn several tasks sequentially while maintaining good performances on all the tasks learned so far, which is quite challenging. This article focuses on class incremental continual recognition, where each task is classification on several classes, and task ID is unavailable during inference.

Some continual learning methods merge the data from new tasks and existing tasks and then train the model on them, which is quite time consuming in most cases. Others fine-tune the trained model on the new task, which suffers from the fact that knowledge obtained from old tasks will be overwritten upon training on a new task, resulting in poor performances on old tasks. This phenomenon is referred to as catastrophic forgetting. Existing works to solve the catastrophic forgetting problem on continual learning can be divided into three categories: (i) *parameter isolation*, dedicating different model parameters to each task, which is restricted to the task incremental setting; (ii) *regularization*, adding certain restrictions during model updates to prevent the model from "forgetting" the learned knowledge of old tasks, which, although it can mitigate the catastrophic forgetting problem to some extent, suffers from a performance drop as the number of tasks continuously increases; and (iii) *replay*, partially memorizing data from old tasks as exemplars and mixing them with the new data for training, which has the best performance among the existing literature but can still inevitably "forget" knowledge obtained previously. Since replay methods achieve the state of the art, we choose to design our owned model based on replay.

Most of the preceding models focus on the training stage and design different methods to prevent forgetting. However, forgetting always happens, even for humans. The *Ebbinghaus forgetting curve* [26] shows that learners will forget about 90% of what they have learned within the first month. Due to the inevitability of forgetting, how to fix the model after forgetting happened needs to be considered. To the best of our knowledge, we are the first to remedy the machine learning model after forgetting happened. As Figure 1 shows, one major reason for forgetting from replay methods is that the stored exemplars are not sufficient to help discriminate between the coming classes and previously learned classes. As no information is provided about the future classes, we could only tackle this issue after we meet the new classes. Inspired by the baby who is born with the ability to learn and recognize continually [35], we assume that most learned samples are stored

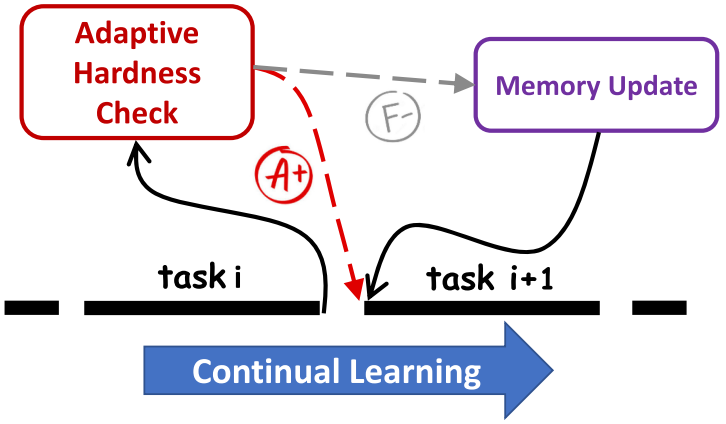

Fig. 2. Illustration of the learning process. After learning a task, our model conducts an adaptive forgetting check for each learned class. If the model performs well and no classes experience severely forgetting, the model could continue to learn the next task (red dashed line). If the model performs poorly and some classes are forgotten, the model needs to conduct memory update on the forgetting classes before learning the next task (gray dashed line).

in long-term memory. As forgetting or interference happen, they are retrieved efficiently to solve the issue.

Based on the preceding observation and assumption, we propose a continual learning method with adaptive memory update for class incremental continual learning. The proposed method only preserves a small number of exemplars in working memory for each task learned. By conducting *half-and-half* validations on exemplars, our method can adaptively determine which classes are experiencing severe forgetting or interference. By exchanging data points between current exemplars set and long-term memory in a non-trivial manner, our method can adaptively maintain a group of exemplars that best help memorize the forgetting classes. Meanwhile, the computational complexity of our model for a single task will not be increased. Moreover, our proposed method can serve as a general plugin for any replay-based approach to further improve their performance. We report our experimental results on comparisons between our method and several baselines on CIFAR-100 and mini-ImageNet, showing that our method can significantly outperform state-of-the-art methods. Figure 2 shows the process of our proposed continual learning method.

Our contributions are summarized as follows:

- We propose using *adaptive memory update* to remedy the machine learning model after forgetting, which is the first work with such mechanism to the best of our knowledge.
- We propose a novel continual recognition model, which is capable of adaptively discovering classes being severely forgotten or interfered with, and then conducting a memory update on these classes through exchanging data points between the current exemplars set and long-term memory.
- We conduct extensive experiments on several real-world datasets to validate our proposed method's superiority over existing baseline approaches.

The rest of the article is organized as follows. We discuss related works in Section 2 and explain the details of our proposed model in Section 3. In Sections 4 and 5, we conduct extensive experiments on various real-world datasets to show the proposed model's advantages against existing state-of-the-art approaches. We finally conclude the article and point out future research directions in Section 6.

2 RELATED WORK

Continual learning, also known as lifelong learning, is continuously acquiring, modifying, and transferring knowledge and skills. It remains a considerable challenge for machine learning and neural networks. In recent years, much work [4, 21, 27, 30, 36] has been proposed to address the catastrophic forgetting problem. The existing methodologies for continual learning can be divided into three categories: parameter isolation-based methods [1, 15, 22, 24], regularization-based methods [10, 18, 23], and replay-based methods [5, 7, 14, 31, 39]. Our work is based on replay.

2.1 Parameter Isolation Based Methods

Parameter isolation based methods [22, 24] dedicate different subsets of the model parameters to each task. When a new task arrives, this kind of method trains a new branch of network with parameters from the previous task (or sometimes entirely new). After the training of this task, the branch is determined to be used for the prediction of this task and can be reused in a layer-wise or neuron-wise way for future tasks. Some methods select a new branch from the original architecture [24], whereas others increase the model capacity to grow a new branch [22].

Most of these works require a task identifier to activate the corresponding branch during prediction, and it impairs the performance of the model significantly. The catastrophic forgetting problem still exists when the model capacity is limited.

2.2 Regularization-Based Methods

Regularization-based methods limit how far the parameters can move from values that were optimal for previous tasks. This is usually implemented via additional terms in the loss function. Some methods discourage the updating of essential parameters for past tasks [18]. They determine essential parameters in the current task first and then penalize the change to these parameters in future training for the new tasks. In the training phase for a new task, they use the output probabilities for each image on the old task as a soft label, and the updated model's output is forced to be close to these soft labels [23].

To some extent, regularization-based methods are a simple way to mitigate the problem of catastrophic forgetting. However, the constraint is still insufficient to counter the accumulation of errors in old tasks, and the drop in performance is inevitable as the number of classes increases.

2.3 Replay-Based Continual Learning

Replay-based continual learning methods, also known as rehearsal methods, need a memory component to store samples from the previous task. The stored samples are often in their raw format, known as exemplars. These previous task samples are replayed while learning a new task to alleviate forgetting. This memory component plays a role like the hippocampus of the complementary learning theory [25].

With the stored samples, the replay-based methods perform pretty well in continual learning. Nevertheless, which samples to store remains a challenge. Several strategies [16, 31] have been proposed for selecting the samples.

Recently, some works have tried to use generative models to generate high-quality samples instead of storing them [34, 38]. However, this also leads to a challenge of training the generative model continually.

Most replay-based methods pick samples randomly from the exemplar set while training, which is not optimal. Aljundi et al. [2] use a selective replay technique that retrieves the most disturbed samples from the exemplar set each time. Shim et al. [33] try to retrieve those exemplars that would

be most helpful for learning. Inspired by game theory, they use the Shapley value to measure the extent to which the samples contributed to the learning.

How to store the exemplar set more efficiently is of equal interest. Riemer et al. [32] use a discretized variational self-encoder to compress the stored exemplar set to save storage costs. Caccia et al. [6] use a variational self-encoder with adaptive vector quantization to compress the exemplar set.

2.4 Class Incremental Scenario

There are different settings of continual learning [37]. In this article, we focus on class incremental continual learning, where the model does not have access to the task-ID at inference time and therefore must be able to distinguish between all classes from all tasks. It is a much more difficult scenario.

3 PROPOSED METHOD

In this section, we first formulate the class incremental continual learning problem in Section 3.1, and then we describe our main components in Sections 3.2 through 3.4.

3.1 Problem Formulation

In class incremental continual learning, a model experiences a sequence of classification tasks denoted by $\mathcal{T} = [(C_1, D_1), (C_2, D_2), \ldots, (C_T, D_T)]$, where each task t is represented by a set of classes $C_t = \{c_t^1, c_t^2, \ldots\}$ and training data D_t. T is the total number of tasks and is not *a priori*. Each training data D_t contains a number of input-target pairs (x_i^t, y_i^t), which is identically and independently drawn from an unknown distribution. Here, x_i^t represents the i-th input example in task t and $y_i^t \in C_t$ represents its class label. We use N_t to represent the set of total classes in all tasks up to and including task t: $N_t = \cup_{i=1}^{t} C_i$ Usually, different tasks contain different classes, and hence the model needs to recognize more and more classes during the training phase.

We denote our model as $M_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, composed by a feature extractor $f : \mathcal{X} \rightarrow \mathbb{R}^d$ and a classifier $g_\mathcal{W} : \mathbb{R}^d \rightarrow \mathcal{Y}$. Here, f can be any convolutional neural network, depending on the complexity of the dataset. The parameters $\mathcal{W}$ of g is a set of weight vectors $\{w_1, w_2, \ldots, w_k\}$, and k is the number of classes learned so far. When our model finishes training on one task and is ready for learning a new task, f and w_i would be temporarily saved as $\hat{f}$ and $\hat{w}_i$ for distillation loss. Meanwhile, classifier $\mathcal{W}$ will expand, and several new weight vectors will be added corresponding to the classes in the new task.

The model gets to learn tasks sequentially, and it is worth mentioning which tasks the samples belong to is not provided in the inference phase.

3.2 Long-Term Memory

When forgetting happens and we want to do something to remedy it, the first thing to determine is what has been forgotten. There are usually two situations for humans. One is that humans could tell what they have forgotten through their own perception. The other one is that humans become confused or make mistakes. The latter situation is quite similar to the catastrophic forgetting of our machine learning model. After determining what has been forgotten or interfered with, humans could review them from some learning resources like the library or the Internet, or even their memory if they have a good one. Since there are no such resources for our machine learning model, we choose to store the learning material, i.e., the training data, into a long-term memory component.

Someone may argue that most existing works follow a rule that the training data for previously learned tasks is unavailable. There are two main reasons for this rule: one is the privacy issue, and the other is storage constraints. However, Knoblauch et al. [19] claim that the optimal continual learning model needs perfect memory. Bartol et al. [3] estimate that the storage capacity per synapse is roughly 4.7 bits of information, and this implies that the total memory capacity of the brain, with its many trillions of synapses, is much larger than the size of our continual learning model.

These findings make us wonder whether this rule is still necessary in today's world of reliable encryption and anonymity technology, and low storage costs. In this article, we relax the rule in a way that the training data is stored in a long-term memory component and can be accessed depending on demand.

3.3 Exemplar Management

Our model maintains a collection of exemplars $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ during training as working memory. When the model finishes training for a task, a few exemplars of each new class will be selected and added to the exemplar set. The data from the collection will be involved in the training on future tasks later.

Some methods use an exemplar set of a fixed size. Every time new exemplars are added, the number of old class exemplars needs to be reduced to meet the fixed size limit. Since the number of tasks is unknown, it is difficult to determine a proper size for the exemplar set at the very beginning. Our model maintains the exemplar set without the constraint on fixed memory size and uses a constant number of exemplars for each class instead. In this way, the exemplar set's size will increase linearly as the number of classes increases. Since the number of exemplars for each class is a small constant, the problem of increasing size is affordable compared to the improvement brought by the exemplar set.

We use herding selection to select exemplars. *Herding selection* [31] is a greedy algorithm for selecting new exemplars for one class. This algorithm iteratively selects exemplars from training data and makes the mean feature vectors of exemplars close to the mean feature vectors of training data until the exemplar set size is met.

3.4 Adaptive Memory Update

To conduct a memory update, there are two fundamental problems to be solved. One is when to update, and the other is how to update.

As we will see later, the drop in model performance varies significantly from class to class, telling us that the probability of a class being forgotten is quite different. Here we let the unit for memory update be one single class rather than all classes in one task. In the following discussion, we are going to focus on a particular class c.

When to Update. The time for our model to update is when the model finds itself unable to distinguish a class from other classes—that is, performing poorly on the data from this class. However, it is difficult for a model to obtain this kind of ability and evaluate its own performance autonomously. We tackle this issue through conducting a half and half validation procedure on our exemplar set.

Before training a new task, the data from each class's exemplar set is randomly divided into two parts. One part is then used as training data to participate in model training together with the data from the new task. After the training is completed, our model would be tested on the other part, which we call *half and half validation*.

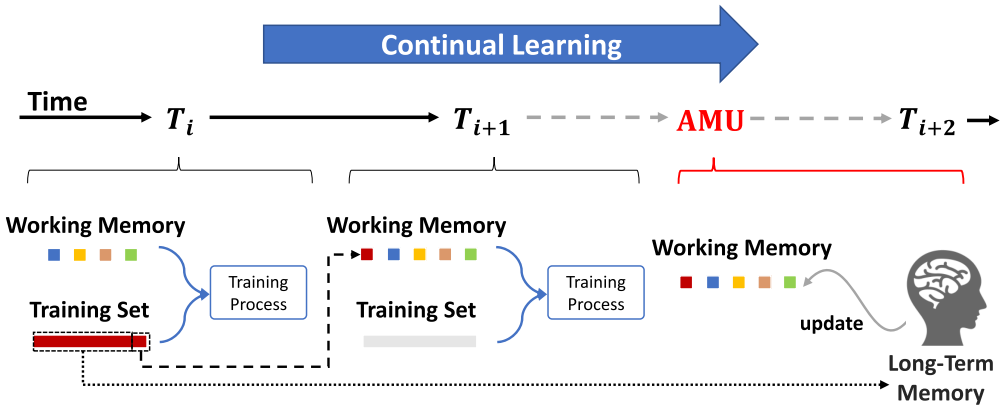

Fig. 3. Adaptive memory update (AMU). After training on the task i, some samples will store in the exemplar set (dashed line), whereas others will be stored in long-term memory (dotted line). In the memory update stage, the same number of samples as the exemplar set are taken from long-term memory to replace the original exemplars.

If our model performs poorly on the other part for a particular class, our model will update the exemplar set for this class. The criterion we adopt for poor performance is whether the recall score is lower than a threshold λ. Any other reasonable criterion can also be used.

Based on the preceding mechanism, our model could determine whether a class c needs a memory update.

How to Update. After training a task, we assume that all the training data is not discarded but instead is stored in a long-term memory component. A simple and brute-force update approach is to take all the data from class c into the training procedure. This method, however, requires a lot of access operations on long-term memory, and the training complexity is greatly increased, especially when there are many classes that need an update.

To address this difficulty, we propose a simple and elegant update approach. We replace the exemplar set with the same number of data samples that are randomly selected from long-term memory. In this way, the long-term memory component only needs to support random access operation by class. Besides, the training complexity with the memory update is not increased at all. Figure 3 illustrates the process of the adaptive memory update.

3.5 Loss Function

Our loss function contains two terms: distillation loss $L_{distill}$ and classification loss L_{clf}.

Distillation Loss. Distillation loss was originally proposed to transfer knowledge between different neural networks [13]. Here we use it to maintain the output of our model on the old tasks while training the model on a new task. When our model is training on task t, the distillation loss $L_{distill}$ is computed as follows:

$$\mathcal{L}_{distill} = -\sum_{(x_i, y_i \in D_t)} \sum_{j \in N_t} q_{ij} \, log \, p_{ij} + (1 - q_{ij}) \, log \, (1 - p_{ij})$$

And q_{ij} is computed as:

$$q_{ij} = \begin{cases} h_j(x_i) & j \in N_{t-1} \\ y_i & j \in N_t - N_{t-1} \end{cases}$$

Here, p_{ij} represents the probability of the i-th sample belonging to class j, and $h_j(x_i)$ is the output from the old model before training on task t:

$$p_{ij} = sigmoid\left(\frac{w_j^\top}{||w_j||_2} f(x_i)\right)$$

$$h_j(x_i) = sigmoid\left(\frac{\hat{w}_j^\top}{||\hat{w}_j||_2} \hat{f}(x_i)\right)$$

Here we adopt a $l2$-normalized form of weight vector to produce logits, which is useful and practical for solving the class imbalance issue in continual learning [14].

Classification Loss. In the traditional multi-class classification task, cross-entropy loss with softmax activation is the most commonly used loss. When considering distillation loss, we find that binary cross-entropy with sigmoid is a better choice than cross-entropy with softmax, because the soft label in distillation loss is more analogous to a multi-label target than a multi-class one. As we can see, the form of distillation loss is binary cross-entropy with sigmoid activation. For consistency, the binary cross-entropy loss is directly performed on the exemplar set.

$$\mathcal{L}_{clf} = -\sum_{(x_i, y_i \in \mathcal{E})} \sum_{j \in N_t} \delta_{y_i=j} \; log \; p_{ij} + \delta_{y_i \neq j} \; log \; (1 - p_{ij})$$

δ is an indicator function. Finally, our loss function is calculated as:

$$\mathcal{L} = \mathcal{L}_{distill} + \mathcal{L}_{clf}$$

The overall incremental training process with the proposed method is presented in Algorithm 1.

4 EXPERIMENTS

4.1 Datasets

We compare the proposed method with several baselines on two widely used image datasets for class incremental continual learning: CIFAR-100 and ImageNet-Subset.

CIFAR-100 [20] consists of 60,000 samples of 32×32 color images in 100 classes, with 600 images per class. The official training set is used as our training data, and the rest is for testing. In this way, each class has 500 training and 100 test samples.

ImageNet-Subset is a subset of ImageNet [9] with only 100 classes, randomly sampled from the original 1000. We also use the official split of training and test, where each class has about 1,300 training and 50 test samples of 224×224 color images.

4.2 Baseline Methods

For a fair comparison, we choose two state-of-the-art methods that use exemplars as baselines. *iCaRL* [31] is a widely used baseline, and it uses a distillation loss and a nearest-mean-of-exemplars classification strategy. *BiC* [39] aims to solve the imbalance problem and proposes a bias correction layer. Besides, we report the performance of the *Base* method that only uses exemplar sets and cross-entropy loss. More specifically, we respectively report the results of CNN predictions and nearest-mean-of-exemplars classification, denoted as CNN and NME for the suffix. Some papers denote it as NCM (nearest-class-mean) instead of NME.

4.3 Protocol

Some work starts with a network trained on a large number of classes and then learns several classes per task incrementally. This setting might give an added advantage to scaling/bias

ALGORITHM 1: IncrementalTrain

Input: $D_t = (X^t, Y^t)$ // Training data for task t
Input: $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ // Exemplar set
Input: $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ // External database
Input: M_θ // Current model parameters
Output: $\mathcal{E}, M_\theta$ // New exemplar set & model

$\mathcal{E}_{train}, \mathcal{E}_{validate} \leftarrow \emptyset$ // Split the exemplar set
for $i \in N_{t-1}$ **do**
  $e_i^t, e_i^v \leftarrow \text{RandomSplit}(e_i)$
  $\mathcal{E}_{train} \leftarrow \mathcal{E}_{train} \cup e_i^t$
  $\mathcal{E}_{validate} \leftarrow \mathcal{E}_{validate} \cup e_i^v$
$M_\theta \leftarrow \text{BackPropagation}(M_\theta, D_t, \mathcal{E}_{train})$ // Train

// Adaptive Memory Update
for $i \in N_{t-1}$ **do**
  if $\mathit{MeetCriterion}(M_\theta, e_i^t)$ **then**
    $e_i \leftarrow \text{RandomSelect}(P_i)$
  else
    $e_i = \text{Join}(e_i^t, e_i^v)$
for $i \in (N_t - N_{t-1})$ **do**
  $e_i = \text{HerdingSelect}(X_i^t, Y_i^t)$
$\mathcal{E} = \{e_1, e_2, \ldots, e_{C_t}\}$ // New exemplar set
return $\mathcal{E}, M_\theta$

correction methods [29]. So in our article, we divide the classes into several tasks of equal size.

For CIFAR-100, we preserve 50 exemplars for each class, and for ImageNet-Subset, we preserve 100 exemplars per class. The class order plays an important role, and hence we run experiments five times on a random but fixed class order. After each task, the resulting classifier is evaluated on the test data of the dataset, considering only classes that have already been trained. The result of the evaluation is a curve of the classification accuracies after each task. The average of these accuracies is also reported as *incremental average accuracy*.

4.4 Implementation Details

All compared models are implemented with PyTorch [28] and trained on TITAN-X GPUs. We adopt ResNet [12] as the convolutional network backbone to extract features for all models. The training images are randomly flipped and cropped as data augmentation. The threshold λ we set for the update is 0.7. For CIFAR-100 and ImageNet-Subset, our model is trained by Adam [17] for 70 epochs with the batch size 128 and the initial learning rate 0.001 while other baselines are fine-tuned to gain their optimal performance. It is worth mentioning that all results are from our implementation for reproducibility.

5 RESULTS AND ANALYSIS

We run our experiments on 5, 10, and 20 tasks with 20, 10, and 5 classes per task. Five random class orders are used, and the mean of incremental average accuracy is reported.

Figures 4, 5, 6, and 7 show the results of CIFAR-100 and ImageNet-Subset using NME inference and CNN inference, respectively. One can see that our proposed method significantly beats other

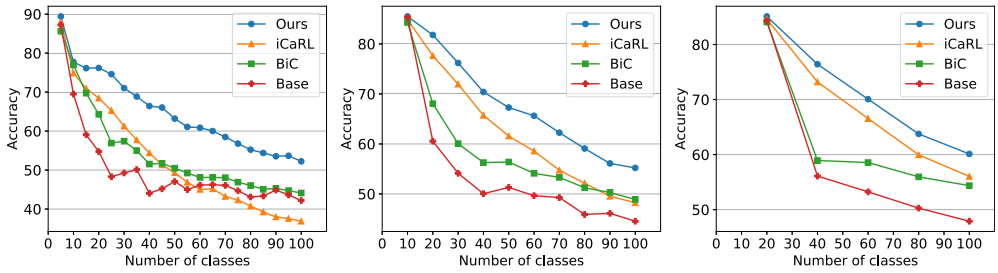

Fig. 4. Evaluation on CIFAR100 using CNN inference.

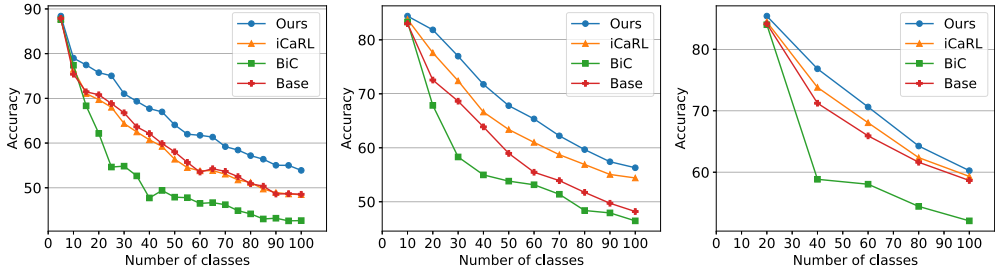

Fig. 5. Evaluation on CIFAR100 using NME inference.

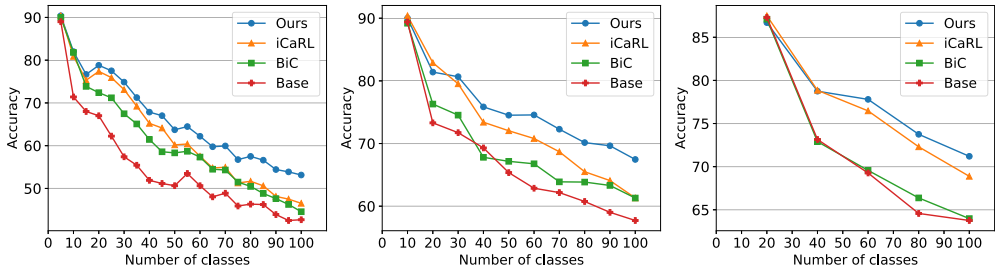

Fig. 6. Evaluation on ImageNet-Subset using CNN inference.

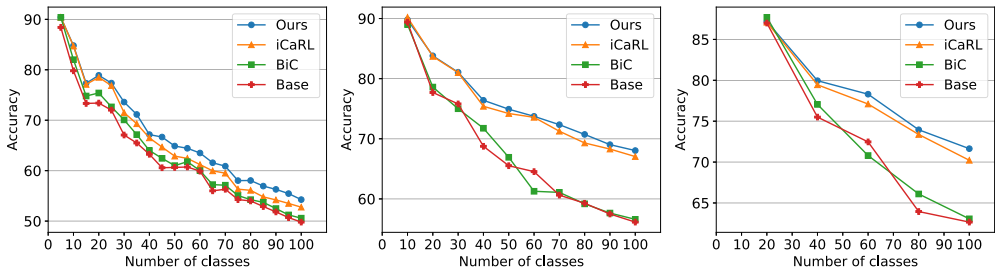

Fig. 7. Evaluation on ImageNet-Subset using NME inference.

approaches, and the larger number of tasks we have, the more ours outperforms other baselines, indicating that our model is more adaptable for real lifelong learning.

General Plugin. Table 3 shows the effect of using the adaptive memory update as a plugin among different models. Ours benefits from this update and obtains the best improvement when using

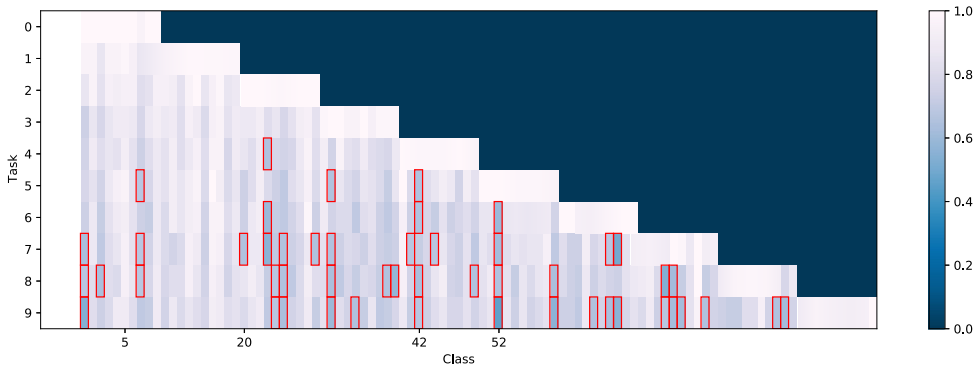

Fig. 8. The recall of different classes after each task. Each small rectangle located at (i, j) in the figure represents the recall of class j after finishing training task i, and the darker color indicates a lower recall value. The red box located at (i, j) represents that class j needs update after task i. The drop in recall varies significantly from class to class, as the model is trained on more and more tasks. Some classes, such as class 5, remain high recall since the beginning, whereas recalls of other classes drop rapidly to low levels after one or two tasks, such as classes 42 and 52. We observe that recalls of some classes, including class 20, return to a high level after memory update, whereas other classes, such as class 52, may require repeated updates. This figure is generated from 10 classes per task experiment on CIFAR100.

Table 1. Using Different Update Strategies

	CNN	**NME**
Herding	68.27	68.65
Random	67.95	68.47
Testing	67.37	66.40
Kmeans	67.58	66.34
GNG	67.81	66.21
Nearest	66.89	65.73

NME to make the inference. However, iCaRL-NME's performance drops a little bit after using this plugin. This is because iCaRL-NME's distillation loss uses the model's previous output as targets for the exemplar set, resulting in a steady accumulation of old classes' errors and the failure of the plugin. Other methods, more or less, have been improved by the plugin.

Memory Size. Table 2 shows the effect of the different memory sizes. *Fixed 2000* means using an exemplar set with fixed capacity, and in this way, the more classes stored, the fewer exemplars reserved for each old class. All methods demonstrate a significant increase in incremental average accuracy using a larger size of the exemplar set. Ours consistently performs best under different memory sizes.

Criterion for Update. Figure 8 shows that the drop in model performance varies significantly from class to class, indicating that the probability of forgetting a class is different. In our proposed model, the criterion according to which a class needs an update is that the recall of the model for the target task is below 0.7. Figure 9 (left) shows how this threshold influences the performance. We conduct experiments on CIFAR100 for 10 tasks with 50 exemplars for each class. We can see the slope of this curve is going up, indicating that as the number of updates increases (a larger threshold tends to increase the probability that a class needs update), the model's gain from the update increases. If our access to the long-term memory is fast enough, we can update all the classes we have learned to achieve better performance.

Table 2. Effect of the Memory Size

	Fixed 2000	20	50	100	150
Ours-NME	**67.73**	**64.93**	**68.47**	**70.65**	**72.74**
Ours-CNN	67.03	64.00	67.94	70.47	72.45
iCaRL-NME	65.20	62.51	64.99	67.02	68.88
iCaRL-CNN	54.45	49.90	62.51	58.53	60.25
BiC-NME	58.84	50.62	56.55	59.51	60.60
BiC-CNN	57.21	47.29	58.29	60.24	61.80
Base-NME	54.74	47.41	60.40	60.20	61.89
Base-CNN	53.29	44.29	53.68	59.40	61.74

Fixed 2000 in the header means using an exemplar set with a fixed capacity of 2,000 samples for all classes. Others in the header mean the number of samples the model stored for each class.

Table 3. Effect of Using Adaptive Memory Update (AMU) as a Plugin

	w/o AMU	w/ AMU
Ours-NME	67.00	**68.22**
Ours-CNN	67.19	**67.76**
iCaRL-NME	**64.73**	64.54
iCaRL-CNN	61.56	**62.15**
BiC-NME	56.55	**58.80**
BiC-CNN	58.29	**59.28**

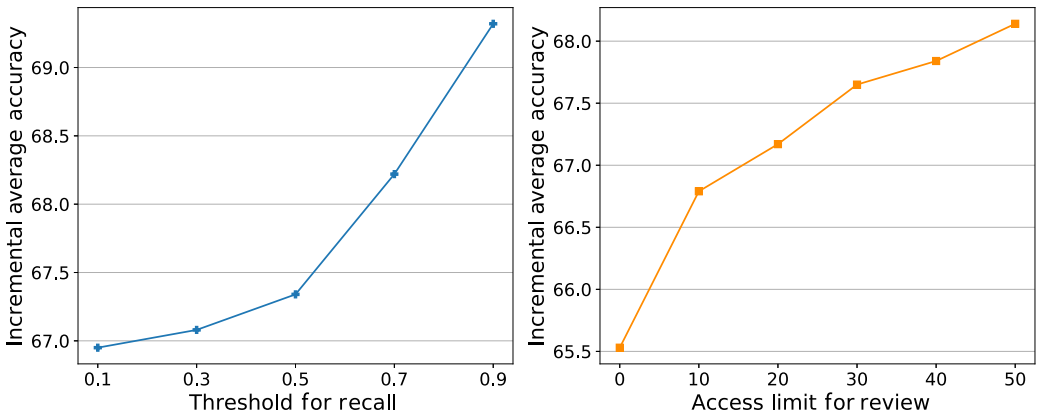

Fig. 9. The effect of threshold.

Access Overheads. However, there is always an overhead for accessing the long-term memory. Sometimes this becomes a non-negligible constraint on our method. Figure 9 (right) shows the performance when we limit the access times. The access limit means how many classes we can update after one task. As we can see, the performance first rises rapidly and then slowly as the access limit increases, which shows that it is enough to update a few classes for impressive improvement.

Update Strategy. In this work, we use a simple and elegant memory update strategy. We replace exemplars with randomly selected samples in the long-term memory. In addition to this method, we tried other strategies based on different sample selection methods. The results are shown in Table 1. Random is our proposed strategy. Nearest strategy means selecting those samples that are close to the misclassified exemplars in feature space. Herding strategy uses herding selection

to select samples. Testing strategy means selecting those samples from the external database that are misclassified. To select the most representative samples, Kmeans strategy selects the cluster centers after conducting the standard Kmeans clustering algorithm. GNG strategy selects samples via growing neural gas algorithm [11], which helps maximize coverage of the feature space.

Exemplars play a crucial role in replay-based methods. The model needs the exemplars to find the best distinguishing feature for the classes learned before. Furthermore, when the model uses NME to infer, exemplars need to approximate the means of classes in feature space. That is why Herding performs best, followed by Random strategy by a narrow margin; Nearest performs worst in both settings; Testing performs poorly using NME but is not bad using CNN. It is noted that these methods, in addition to Random, require additional computation to obtain samples' features, classification results, or cluster centers, but finally come to a comparable result or worse. So we choose Random as our update strategy for efficiency.

6 CONCLUSION

In this article, we proposed a novel method using an adaptive memory update mechanism and a novel loss to tackle the catastrophic forgetting problem in class incremental continual learning. As far as we know, it is the first time that the concept of remedy has been brought into the field of continual learning. Experimental results on CIFAR-100 and ImageNet-Subset demonstrate that the proposed method achieves better performance than existing state-of-the-art continual learning algorithms. This work also starts a discussion on bringing humans' mnemonic methods and designing human-like models to solve catastrophic forgetting in continual learning.

REFERENCES

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. 2020. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3931–3940.

[2] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. 2019. Online continual learning with maximal interfered retrieval. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19)*. 11872–11883.

[3] Thomas M. Bartol Jr., Cailey Bromer, Justin Kinney, Michael A. Chirillo, Jennifer N. Bourne, Kristen M. Harris, and Terrence J. Sejnowski. 2015. Nanoconnectomic upper bound on the variability of synaptic plasticity. *eLife* 4 (2015), e10778.

[4] Eden Belouadah and Adrian Popescu. 2018. DeeSIL: Deep-shallow incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*.

[5] Eden Belouadah and Adrian Popescu. 2019. IL2M: Class incremental learning with dual memory. In *Proceedings of the IEEE International Conference on Computer Vision*. 583–592.

[6] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. 2020. Online learned continual compression with adaptive quantization modules. In *Proceedings of the International Conference on Machine Learning*. 1240–1250.

[7] F. M. Castro, M. Marín-Jiménez, Nicolás Guil Mata, C. Schmid, and Alahari Karteek. 2018. End-to-end incremental learning. In *Computer Vision—ECCV 2018*. Lecture Notes in Computer Science, Vol. 11216. Springer, 241–257.

[8] Zhiyuan Chen and Bing Liu. 2018. *Lifelong Machine Learning* (2nd ed.). Synthesis Lectures on Artificial Intelligence and Machine Learning.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*.

[10] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. 2019. Learning without memorizing. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 5138–5146.

[11] Bernd Fritzke et al. 1994. A growing neural gas network learns topologies. In *Proceedings of the 7th International Conference on Neural Information Processing Systems (NIPS'94)*. 625–632.

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 770–778.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *Proceedings of the NIPS Deep Learning and Representation Learning Workshop*.

[14] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*.
[15] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. 2019. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems*. 13669–13679.
[16] David Isele and Akansel Cosgun. 2018. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980 (2015).
[18] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526.
[19] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. 2020. Optimal continual learning has perfect memory and is NP-hard. In *Proceedings of the International Conference on Machine Learning*.
[20] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.
[21] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. 2019. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*. 312–321.
[22] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. 2019. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the International Conference on Machine Learning*.
[23] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2017), 2935–2947.
[24] Arun Mallya and S. Lazebnik. 2018. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7765–7773.
[25] James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review* 102, 3 (1995), 419.
[26] Jaap M. J. Murre and Joeri Dros. 2015. Replication and analysis of Ebbinghaus' forgetting curve. *PloS One* 10, 7 (2015), e0120644.
[27] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. 2019. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 11321–11329.
[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19)*. 8026–8037.
[29] Ameya Prabhu, Philip Torr, and Puneet Dokania. 2020. GDumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision (ECCV'20)*.
[30] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. 2020. iTAML: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13588–13597.
[31] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 5533–5542. https://doi.org/10.1109/CVPR.2017.587
[32] Matthew Riemer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. 2019. Scalable recollections for continual lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1352–1359.
[33] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. 2021. Online class-incremental continual learning with adversarial Shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9630–9638.
[34] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 2990–2999.
[35] Linda Smith and Michael Gasser. 2005. The development of embodied cognition: Six lessons from babies. *Artificial Life* 11, 1-2 (2005), 13–29.
[36] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. 2020. Topology-preserving class-incremental learning. In *Computer Vision—ECCV 2020*. Lecture Notes in Computer Science, Vol. 12364. Springer, 254–270.
[37] Gido M. van de Ven and Andreas S. Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (2019).

[38] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, and Bogdan Raducanu. 2018. Memory replay GANs: Learning to generate new categories without forgetting. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. 5962–5972.

[39] Y. Wu, Yan-Jia Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 374–382.

Received 10 May 2022; revised 17 October 2022; accepted 20 November 2022