Learning with Selective Forgetting

Takashi Shibata , **Go Irie** , **Daiki Ikami** and **Yu Mitsuzumi**
NTT Communication Science Laboratories, NTT Corporation, Japan
{t.shibata, goirie}@ieee.org, {daiki.ikami.ef, yu.mitsuzumi.ae}@hco.ntt.co.jp

Abstract

Lifelong learning aims to train a highly expressive model for a new task while retaining all knowledge for previous tasks. However, many practical scenarios do not always require the system to remember all of the past knowledge. Instead, ethical considerations call for selective and proactive forgetting of undesirable knowledge in order to prevent privacy issues and data leakage. In this paper, we propose a new framework for lifelong learning, called *Learning with Selective Forgetting*, which is to update a model for the new task with forgetting only the selected classes of the previous tasks while maintaining the rest. The key is to introduce a class-specific synthetic signal called *mnemonic code*. The codes are "watermarked" on all the training samples of the corresponding classes when the model is updated for a new task. This enables us to forget arbitrary classes later by only using the mnemonic codes without using the original data. Experiments on common benchmark datasets demonstrate the remarkable superiority of the proposed method over several existing methods.

1 Introduction

Deep learning often suffers from a phenomenon called catastrophic forgetting. When a network is updated for a new task, its performance on previous tasks dramatically degrades. To mitigate this harmful effect, lifelong learning (or continual learning) has been explored, in which the network is updated to adapt to a new task (e.g., a new set of classes and a new instance) without forgetting the results of past learning. Major methods can be categorized into memory-replay-based [Rebuffi *et al.*, 2017; Lopez-Paz and Ranzato, 2017], parameter-freezing-based [Mallya and Lazebnik, 2018; Mallya *et al.*, 2018], and regularization-based [Kirkpatrick *et al.*, 2017; Li and Hoiem, 2017; Zenke *et al.*, 2017; Aljundi *et al.*, 2018] approaches. Most existing methods have been designed to learn a highly expressive model for the new task while preserving all of the knowledge for the previous tasks.

Meanwhile, artificial intelligence is currently facing a new type of problem; as artificial intelligence has become more practical and connected to our everyday lives, various ethical issues such as privacy protection and data leakage prevention have become critical topics. This has brought new challenges to the field, covering learning from encrypted data [Gilad-Bachrach *et al.*, 2016], preventing learning of unintended information [Wang *et al.*, 2019], privacy preserving localization [Speciale *et al.*, 2019a; Speciale *et al.*, 2019b], just to name a few. Even lifelong learning cannot avoid this issue either. Retaining the complete knowledge of all previous tasks is a double-edged sword – it possibly leads to the risk of data leakage and invasion of privacy. Moreover, it is not always necessary to have the complete knowledge of the previous tasks so desirable to have a mechanism for forgetting knowledge no longer needed. For example, a face recognition system at an office entrance gate would not need to remember the faces of staff who have transferred to other departments.

These observations motivate us to propose a new lifelong learning framework called *Learning with Selective Forgetting (LSF)*, which aims to avoid catastrophic forgetting of previous tasks while selectively forgetting only specified sets of past classes. To the best of our knowledge, our study is the first to introduce the new forgetting problem to lifelong learning and proposes a solution to it. In this paper, we focus on task-incremental learning. The challenge is to forget only the specified classes while preventing catastrophic forgetting for the rest without using the original data of the previous tasks. Our method solves this issue by performing a special type of data augmentation that embeds a class-specific signal, called *mnemonic code*, in all the samples of the corresponding class when updating the model. This makes the class information tightly linked to the corresponding code, making it possible to forget arbitrary classes later on simply by discarding the codes corresponding to the classes. Experiments on common benchmark datasets demonstrate the remarkable superiority of our proposed method to existing approaches.

2 Related Work

2.1 Lifelong Learning

We briefly review three mainstream approaches in lifelong learning, memory-replay-based, parameter-freezing-based, and regularization-based, and we highlight the contributions of our work.

Memory-replay: The memory-replay-based approach uses a set of original samples of previous tasks when updat-

ing the model for a new task [Chaudhry *et al.*, 2018b; Rebuffi *et al.*, 2017; Lopez-Paz and Ranzato, 2017]. Instead of using the original samples, some methods train deep generative models to generate pseudo samples [Wu *et al.*, 2018; Shin *et al.*, 2017]. Several recent papers have proposed algorithms to solve the problem of data imbalance between the current task and the previous tasks [Zhao *et al.*, 2020; Wu *et al.*, 2019; Liu *et al.*, 2020].

Parameter-freezing: The basic idea of this approach is to use different model parameters for each task. Several strategies have been proposed, such as networks switching the nodes or branches to be used depending on the tasks [Mallya and Lazebnik, 2018; Mallya *et al.*, 2018] or adding new nodes or branches every time a new task is learned [Rusu *et al.*, 2016; Aljundi *et al.*, 2017]. A hybrid version of these approaches has also been proposed [Hung *et al.*, 2019].

Regularization: This approach leverages the previous tasks' knowledge implicitly by introducing additional regularization terms. This approach can be grouped into data-driven-based [Li and Hoiem, 2017; Hou *et al.*, 2018; Dhar *et al.*, 2019] and weight-constrain-based [Kirkpatrick *et al.*, 2017; Zenke *et al.*, 2017; Aljundi *et al.*, 2018; Chaudhry *et al.*, 2018a; Lee *et al.*, 2017; Yu *et al.*, 2020]. The former utilizes the knowledge distillation, while the latter introduces a prior on the model parameters.

Our method is categorized into the regularization-based approach. To summarize, the existing algorithms are designed to retain all the information of the classes for the past tasks. Unlike these, our contributions of this paper are to propose a new problem setting of lifelong learning that requires forgetting only specified classes and a solution to this problem.

2.2 Machine Unlearning

The concept of Machine Unlearning (MU) was first introduced by Cao et al. [Cao and Yang, 2015]. Its typical definition is to remove the effect of specified training samples without retraining the whole model so that the resulting model is indistinguishable from a model trained on a dataset without those samples. General approaches are to train multiple small models on separated subsets of the training data to prevent retraining the whole model [Bourtoule *et al.*, 2019] or to utilize vestiges of the learning process, i.e., the stored learned model parameters and their gradients [Wu *et al.*, 2020]. Specialized methods for some basic learning algorithms such as linear discriminant analysis [Guo *et al.*, 2020] and k-means [Ginart *et al.*, 2019] have also been presented. Inspired by differential privacy [Abadi *et al.*, 2016], Eternal Sunshine of the Spotless Net [Golatkar *et al.*, 2020] introduced a scrubbing procedure that removes information from the trained weights of deep neural networks using the Fisher information matrix. Mixed-Linear Forgetting [Golatkar *et al.*, 2021] derived a tractable optimization problem by linearly approximating the amount of change in weights due to the addition of training data. Variational Bayesian inference also provides a compelling approach for MU [Nguyen *et al.*, 2020].

Our work differs from these previous studies in the following two points. First, we focus on lifelong learning. To the best of our knowledge, this is the first work that considers the forgetting problem in the context of lifelong learning. Second, we address the problem of class-level forgetting, i.e., making a specified set of classes unrecognizable, rather than sample-level forgetting. This is a practical forgetting problem that has not yet been thoroughly studied in past MU literature.

3 Learning with Selective Forgetting

Let us begin with an introduction to a standard lifelong learning setting. Denote by $\{\mathcal{D}_1, \cdots \mathcal{D}_k, \cdots \mathcal{D}_K\}$ a sequence of datasets, where $\mathcal{D}_k = \{(\mathbf{x}_k^i, y_k^i)_{i=1}^{n_k}\}$ is the dataset of the k-th task. $\mathbf{x}_k^i \in \mathcal{X}$ is an input and $y_k^i \in \mathcal{Y}$ is its class label. While observing the datasets in a streaming manner, the purpose of standard lifelong learning is to learn a model $f_\theta : \mathcal{X} \to \mathcal{Y}$ parameterized by θ so that it can map a test input $\mathbf{x}$ of any learned tasks to its correct class label y.

Now we define our new problem illustrated in Figure 1, called Learning with Selective Forgetting (LSF). In this problem, each of the learned classes is assigned to either preservation set or deletion set. Formally,

- **Preservation Set** C_k^P: A set of classes learned in the past and should be preserved at k-th task.
- **Deletion Set** $\overline{C_k^P}$: A set of classes still memorized and should be forgotten at k-th task (the complement of C_k^P).

At the k-th task, we are given the dataset $\mathcal{D}_k$ and the preservation set C_k^P. We use index k for the new task and p for the previous tasks.

Definition 1 (LSF Problem)**.** The Learning with Selective Forgetting (LSF) problem is defined as follows:

- **Objective**: Learn a model $f_\theta : \mathcal{X} \to \mathcal{Y}$. This model f_θ should map a test input $\mathbf{x}$ to its correct class label y if $\mathbf{x}$ is in the preservation set C^P. Otherwise, f_θ should map $\mathbf{x}$ to a wrong class label $y' \neq y$.
- **Constraint**: No original samples or generative models for the past tasks are available after the new task begins.

4 Method

We propose a method to solve the LSF problem. An overview of our method is shown in Fig. 2. Our method uses a multi-headed network architecture that has one head per task, which is a common architecture in lifelong learning [Li and Hoiem, 2017; Chaudhry *et al.*, 2018a]. We first introduce *mnemonic code*, which is the key to solving the LSF problem, and then we present loss functions for learning our model. Finally, we empirically analyze the key properties of our method.

4.1 Mnemonic Code

The challenge of the LSF problem is to retain memorizing the classes listed in the preservation set while forgetting those in the deletion set without accessing the original dataset. Our idea aims to associate information of each class with a fairly simple code, called mnemonic code, and to use only that code to control whether the class will be retained or forgotten.

We implement this idea as a special type of data augmentation. An overview of the process is illustrated in the left-hand

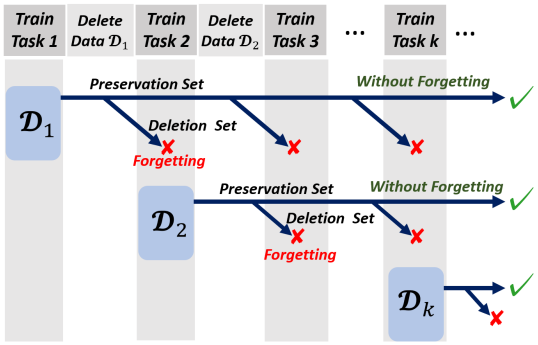


Figure 1: **Problem Setting of Learning with Selective Forgetting.** The goal is to carry out both selective forgetting and lifelong learning without using the original data of previous tasks.

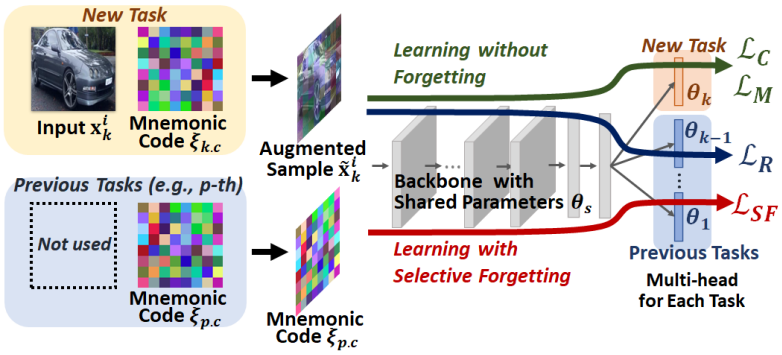


Figure 2: **Overview of Our Method.** We introduce *mnemonic code* – a class-specific random signal embedded in each sample of the same class and is trained to be an anchor of the class. Remembering/forgetting of the class information can be performed by only using the corresponding code without using the original data of the past tasks.

side of Fig. 2. When a new task is received, one synthetic image is generated per class with random pixel values as the class-specific mnemonic code and embedded in all the samples of the corresponding class. Formally, let $\{\xi_{k,c}\}$ be a set of mnemonic codes, where $\xi_{k,c}$ is the code for k-th task of the c-th class. During training for the k-th task, we generate an augmented sample $\tilde{\mathbf{x}}_k^i$ by embedding the mnemonic code $\xi_{k,c}$ into the original sample $\mathbf{x}_k^i$ of the c-th class such like mixup [Zhang *et al.*, 2018]:

$$\tilde{\mathbf{x}}_k^i = \lambda \mathbf{x}_k^i + (1-\lambda)\xi_{k,c}, \tag{1}$$

where λ is a uniform random variable in $[0, 1]$. Besides the set of the originals $\{(\mathbf{x}_k^i, y_k^i)_{i=1}^{n_k}\}$, we also use the augmented samples $\{(\tilde{x}_k^i, y_k^i)_{i=1}^{n_k}\}$ to update our model at the k-th task. Once the updating is done, we retain only the codes $\{\xi_{p,c}\}$ for later tasks to control remembering and forgetting the classes learned in the past tasks.

The intuition behind this procedure is as follows. By training with such augmented data, the samples of the same class are aggregated around the corresponding mnemonic code in the feature space. We therefore can control whether or not to maintain the feature distribution around the code locally depending on whether or not to use the code later at updating the model for a new incoming task, leading it possible to remember or forget arbitrary classes by only using the corresponding codes but without using the original samples (we will show later analytic results in Sec. 4.3). This idea is inspired by a human learning technique called "mnemonics" that aids in memory retention by associating different types of information (e.g., images and words), hence the name.

Implementation of Mnemonic Code: We use random color patterns to generate our mnemonic code as shown in Fig. 2. Specifically, we assign a random color to each grid of an image of the same size as the original sample. Other types of codes are possible; however, we argue several strengths of using such a random code: i) the random pattern can be generated easily, ii) the patterns are i.i.d. for each class and each task, and iii) unlike existing memory-based-approaches that uses (a part of) the original samples, the pattern itself does not directly represent any information of the raw data, which is suitable for privacy protection and data leakage prevention purposes. Interestingly, as we will show later in Sec. 5.3, the performance of our random code is comparable to that of a content-based code, i.e., an average image of the original samples within the same class, which emphasizes the advantages of our version. One limitation so far is that our code has been customized for image data, but the idea itself can be readily extended to other data types, which will be a compelling future research direction.

4.2 Loss Function

We train the model with our mnemonic codes. As shown in Fig. 2, the total loss function $\mathcal{L}$ for training consists of four terms: classification loss $\mathcal{L}_C$, mnemonic loss $\mathcal{L}_M$, selective forgetting loss $\mathcal{L}_{SF}$, and regularization term $\mathcal{L}_R$. The first two are for learning a new task and the last two are for maintaining the previous tasks.

$$\mathcal{L} = \overbrace{\mathcal{L}_C + \mathcal{L}_M}^{\text{new task}} + \overbrace{\mathcal{L}_{SF} + \mathcal{L}_R}^{\text{previous tasks}}. \tag{2}$$

Below we detail each of them one-by-one.

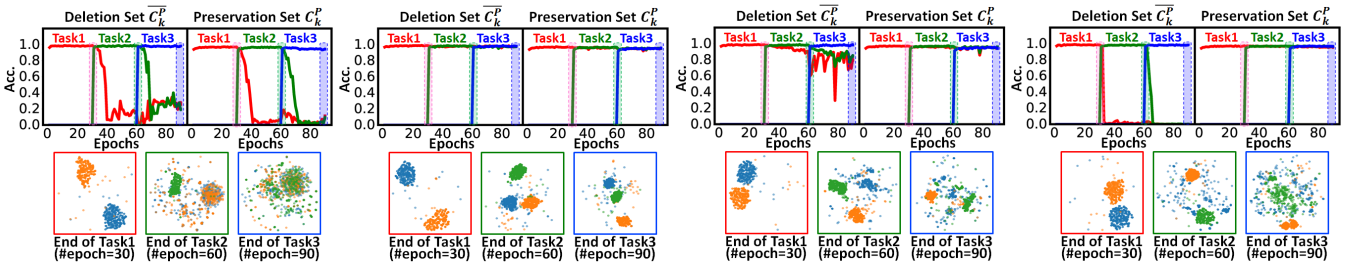
Classification Loss $\mathcal{L}_C$: The classification loss for the k-th new task is given as

$$\mathcal{L}_C = \frac{1}{N_k} \sum_i l(\mathbf{x}_k^i, y_k^i), \tag{3}$$

where N_k is the number of the training samples in the k-th task, $l(\mathbf{x}, y)$ is a loss function for the input $\mathbf{x}$ and its class label y. A typical choice would be softmax cross entropy (CE) or additive margin softmax (AMS) loss [Wang *et al.*, 2018a; Wang *et al.*, 2018b]. We use AMS for $l(\mathbf{x}, y)$, as we found it is better than CE.

Mnemonic Loss $\mathcal{L}_M$: In addition to the classification loss that uses the original samples $\mathcal{D}_k = \{(\mathbf{x}_k^i, y_k^i)_{i=1}^{n_k}\}$, we also use another loss using the augmented samples with our mnemonic codes $\tilde{\mathcal{D}}_k = \{(\tilde{\mathbf{x}}_k^i, y_k^i)_{i=1}^{n_k}\}$ for tying each code to the corresponding class. The loss function is given by

$$\mathcal{L}_M = \frac{1}{N_k} \sum_i l(\tilde{\mathbf{x}}_k^i, y_k^i). \tag{4}$$



(a) Vanilla (b) EWC (c) EWC* (d) **Ours**$_E$

Figure 3: **Analysis**. The accuracy of each task for each epoch (top) and t-SNE plots of the features obtained from the last layer of the backbone network at the end of each task (bottom) are shown. Each color in the t-SNE plot represents the following categories (best viewed in color). *Orange*: Belongs to the preservation set throughout all three tasks. *Blue*: Changes from the preservation set to the deletion set after completing task 1. *Green*: Learns as a new task, i.e., the preservation set, in task 2, then change to the deletion set in task 3.

We use AMS for $l(\cdot,\cdot)$ as the classification loss.

Selective Forgetting Loss $\mathcal{L}_{SF}$: The aim of this loss function is to keep remember only the classes in the preservation set and to forget the others in the deletion set. This can be achieved by training with only the mnemonic codes corresponding to the classes in the preservation set and discarding the other codes. For convenience, let us denote by ξ_p^i the mnemonic code used to generate $\tilde{\mathbf{x}}_p^i$ (i.e., the code for the class of $\mathbf{x}_p^i$). The loss function is

$$\mathcal{L}_{SF} = \gamma_{SF} \sum_{p=1}^{k-1} \frac{1}{N_p} \sum_i l(\xi_p^i, y_p^i), \tag{5}$$

where N_p is the number of training samples at the p-th task, and γ_{SF} is a balancing weight. $l(\cdot,\cdot)$ is the AMS function. Note that this loss function does not use any of the original samples. By ignoring the codes of the classes in the deletion set, these classes will experience catastrophic forgetting. This allows us to achieve the selective forgetting for the previous tasks without using any of the original samples.

Regularization Term $\mathcal{L}_R$: The regularization term is often introduced to prevent catastrophic forgetting. In this work, we consider using three existing regularization terms, namely, Learning without Forgetting (LwF) [Li and Hoiem, 2017], Elastic Weight Consolidation (EWC) [Kirkpatrick *et al.*, 2017], and Memory Aware Synapses (MAS) [Aljundi *et al.*, 2018]. LwF and EWC are originally designed to retain all classes, while we only need to memorize the classes included in the preservation set for our LWS problem. Thus, we make the following minor modifications to adapt LwF and EWC to our problem. The modified versions are distinguished from their original versions by "*", for example LwF*.

- LwF* [Li and Hoiem, 2017]: The regularization term of LwF* is defined as $\mathcal{L}_{\text{LwF}^*} = -\gamma \sum_{i \in C_k^P} y_o'^{(i)} \log \hat{y}_o'^{(i)}$, where γ is the weight for the term, and i is the index of the class label. We change the summation to only be taken over the preservation set, i.e., $j \in C_k^P$. y_o' and $\hat{y}_o'$ are the modified versions of recorded and current probabilities as in [Li and Hoiem, 2017][1].

- EWC* [Kirkpatrick *et al.*, 2017]: The regularization term of EWC* is $\mathcal{L}_{\text{EWC}^*} = \frac{\gamma}{2} \sum_{q,p} \mathcal{F}_{q,p} (\theta_q - \hat{\theta}_{q,p})^2$, where γ is the weight for the regularization term, $\mathcal{F}_{q,p}$ is the diagonal component of the Fisher matrix for the p-th previous task corresponding to the q-th parameter $\hat{\theta}_{q,p}$. We change the Fisher matrix to be evaluated only for the classes corresponding to the preservation set[2].

- MAS [Aljundi *et al.*, 2018]: The regularization $\mathcal{L}_R$ for MAS is given by $\mathcal{L}_{\text{MAS}} = \frac{\gamma}{2} \sum_{q,p} \Omega_{q,p} (\theta_q - \hat{\theta}_{q,p})^2$, where γ is the regularization strength, $\Omega_{q,p}$ is the constraint strength, i.e., the importance parameter, for the p-th previous task for the q-th parameter, which is estimated by the sensitivity of the squared l_2 norm of the function output to their changes.

Beyond the cases of using each of these individually as our regularization term $\mathcal{L}_R$, we can also consider combinations of them. Specifically, we test the following two versions of combinations for our method in the experiments.

$$\mathcal{L}_R = \mathcal{L}_{\text{LwF}^*} + \mathcal{L}_{\text{EWC}^*}, \tag{6}$$
$$\mathcal{L}_R = \mathcal{L}_{\text{LwF}^*} + \mathcal{L}_{\text{MAS}}, \tag{7}$$

which we denote Ours$_E$ and Ours$_M$, respectively.

4.3 Analysis

In our preliminary analysis, we demonstrate that our mnemonic code can forget only the specified classes in the deletion set while maintaining the rest.

Setting: We use Permuted MNIST [Kirkpatrick *et al.*, 2017], which is an artificial dataset often used for lifelong learning benchmarks. We prepare three tasks with different permutations; ten digit classes for each task (30 classes total). In this analysis, we always set three classes {"0", "1", "2"} at each task as the deletion set and the other seven classes as the

[1]The modified versions of recorded and current probabilities, i.e., y_o' and $\hat{y}_o'$, are given by $y_o'^{\,i} = (y_o'^{(i)})^{1/T} / \sum_{j \in C_k^P} (y_o'^{(j)})^{1/T}$ and $\hat{y}_o'^{\,i} = (\hat{y}_o'^{(i)})^{1/T} / \sum_{j \in C_k^P} (\hat{y}_o'^{(j)})^{1/T}$, where T is a hyperparameter for the distillation knowledge. We set $T = 2$ as in the original paper [Li and Hoiem, 2017]. The summation is only taken over the preservation set, i.e., $j \in C_k^P$.

[2]In the case of multiple tasks, EWC requires storing the Fisher matrix for each task independently and performing regularization on all of them together [Chaudhry *et al.*, 2018a].

preservation set, so whenever a new task comes, only {"0", "1", '2"} from the past tasks need to be forgotten, and the rest are required to be retained.

We compare four methods: 1) **Vanilla**, which only uses the classification loss $\mathcal{L}_C$, 2) **EWC**, 3) **EWC***, and 4) **Ours**$_E$. The standard two-conv-two-FC CNN is used for all methods[3].

Results: Figure 3 shows the accuracy vs. epoch plots (top) and t-SNE visualization results (bottom), where the features on the final layer of the shared backbone at the end of each task are visualized. The two accuracy plots show the performance for the deletion set (left) and the preservation set (right). In contrast to the others, we can see that Ours correctly reduces the accuracy of the past three classes to be forgotten and also maintains the accuracy of the seven classes to be retained. Vanilla forgets everything of the past and EWC remembers all. EWC*, which has been modified to apply regularization only to classes that should be preserved, tends to manage the task correctly, but is still inadequate. This proves that straightforward modifications of existing methods are not satisfactory and supports the unique effectiveness of our mnemonic code. Another important observation is that the mnemonic code is only embedded in the training images, which leads to a gap between the training and test images, however, this has no significant negative impact on the final classification accuracy.

The t-SNE plots also show that only Ours could keep the samples of the classes to be remembered agglomerated in the feature space and quickly scattered those to be forgotten, which is the desirable behavior for the problem. This implies that our random mnemonic code, despite its simplicity, is able to tightly link the samples of each class to the corresponding code in the feature space, and as a result, can control remembering/forgetting of the classes individually.

5 Experiments

5.1 Setting

Datasets: We use three widely used benchmark datasets for lifelong learning, i.e., CIFAR-100, CUB200-2011 [Wah *et al.*, 2011], and Stanford Cars [Krause *et al.*, 2013]. CUB-200-2011 has 200 classes with 5,994 training images and 5,794 test images. CIFAR-100 contains 50,000 training images and 1,0000 test images overall. Stanford Cars comprises 196 cars of 8,144 images for training and 8,041 for testing. Unless otherwise noted, as the analysis in Sec. 4.3, the first 30% of classes for each task belongs to the deletion set, while the other classes belong to the preservation set.

Implementation Details: We used ResNet-18 [He *et al.*, 2016] for the classification model. The final layer was changed to the multi-head architecture as shown in Fig. 2. We trained the network for 200 epochs for each task. Mini-batch sizes were set to 128 for new tasks and 32 for past tasks in CIFAR-100, and 32 for new tasks and 8 for previous tasks in CUB-200-2011 and Stanford Cars. The weight decay was 5.0×10^{-4}. We used SGD for optimization. We employed a standard data augmentation strategy: random crop, horizontal flip, and rotation. In this experiment, we used Xavier's initialization.

Baselines: We compared our proposed method with LwF [Li and Hoiem, 2017], EWC [Kirkpatrick *et al.*, 2017], and MAS [Aljundi *et al.*, 2018], which are the popular regularization-based lifelong learning methods. In the following experiments, γ for LwF/LwF*, EWC/EWC*, and MAS are set to 5, 100, and 5, respectively. The weight parameter for the selective forgetting loss γ_{SF} is set to 10. We compare the above methods, including their combinations. To sum up, the specific methods compared are as follows:

- **Vanilla:** Trained using only the classification loss L_C,
- **LwF**: [Li and Hoiem, 2017],
- **LwF***: Modified version of LwF,
- **EWC**: [Kirkpatrick *et al.*, 2017],
- **EWC***: Modified version of EWC,
- **EWC*****+LwF***: Combination of EWC* and LwF*
- **MAS**: [Aljundi *et al.*, 2018]
- **MAS+LwF***: Combination of MAS and LwF*
- **Ours**$_E$: Our method with EWC* and LwF*
- **Ours**$_M$: Our method with MAS and LwF*

Evaluation Metric: In our LSF problem setting, the goal is to forget the deletion set and preserve the preservation set. There is no suitable evaluation metric for evaluating the performance on this setting, because it contains a new criterion, selective forgetting. We introduce a new evaluation metric S, called *Learning with Selective Forgetting Measure (LSFM)*.

LSFM is calculated as the harmonic mean of the two standard evaluation measures for lifelong learning [Chaudhry *et al.*, 2018b]: the average accuracy A_k for the preservation set and the forgetting measure F_k for the deletion set, i.e.,

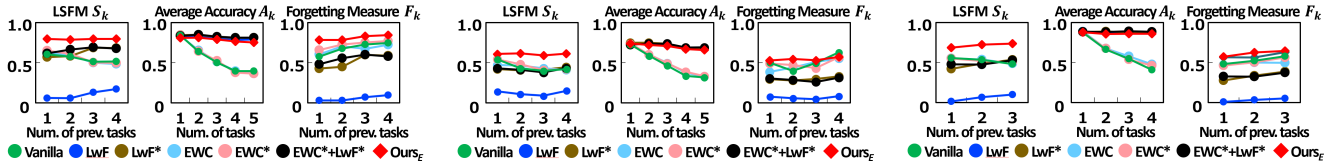
$$S_k = \frac{2 \cdot A_k \cdot F_k}{A_k + F_k}. \tag{8}$$

The average accuracy A_k is evaluated for the preservation set after the model has been trained up untill the k-th task. The specific definition is given by $A_k = \frac{1}{k}\sum_{p=1}^{k} a_{k,p}$, where $a_{k,p}$ is the accuracy for the p-th task after the training for the k-th task is completed. A_k is evaluated only for the preservation set. Similarly, the forgetting measure F_k is computed for the deletion set after completing the k-th task. This is given by $F_k = \frac{1}{k}\sum_{p=1}^{k} f_k^p$, where $f_k^p = \max_{l \in 1 \cdots k} a_{l,p} - a_{k,p}$, which represents the largest gap (decrease) from the past to the current accuracy for the p-th task. This is evaluated only for the deletion set[4]. The ranges of A_k and F_k are both $[0, 1]$.

We report the averages of S_k, A_k and F_k over k after the last task has been completed, which are denoted by S, A, and F, respectively.

[3]The detailed configuration of the CNN is: Conv(3,32) - Conv(3,64) - MaxPool(2) - Dropout(0.25) - Linear(9216,120) - Dropout(0.5) - Linear(120,10), where Conv(k,c) denotes a convolution-ReLU layer with the kernel size $k \times k$ and output channel c, Maxpool(2) denotes max pooling with stride 2, and Dropout(p) denotes dropout with probability p.

[4]In the first task (i.e., the number of previous tasks is zero), no class belongs to the deletion set, so F_k and S_k are not defined.



(a) Results on CIFAR-100 (b) Results on CUB-200-2011 (c) Results on Stanford Cars

Figure 4: **Per-Task Performance.** The left, center, and right figures show the LSFM S_k, average accuracy A_k for the preservation set, and forgetting measure F_k for the deletion set at the end of each task, respectively. Higher is better for each.

	CIFAR-100 # Task:5, # Class:20		CUB-200-2011 # Task:5, # Class:40		Stanford Cars # Task:4, # Class:49	
	$S\uparrow$	$(A\uparrow, F\uparrow)$	$S\uparrow$	$(A\uparrow, F\uparrow)$	$S\uparrow$	$(A\uparrow, F)\uparrow$
Vanilla	51.79	(39.66,74.62)	42.23	(31.77,62.93)	48.14	(41.08,58.12)
LwF	17.23	(79.05,9.67)	15.20	(69.04,8.54)	9.93	(88.10,5.26)
LwF*	68.24	(81.32,58.79)	44.54	(68.27,33.05)	53.70	(88.22,38.60)
EWC	48.57	(36.54,72.42)	41.36	(32.97,55.47)	48.74	(47.72,49.80)
EWC*	49.61	(36.58,77.08)	42.08	(33.38,56.92)	50.71	(46.17,56.24)
EWC*+LwF*	67.64	(81.20,57.96)	43.42	(69.29,31.62)	52.79	(88.65,37.58)
MAS	47.46	(34.89,74.17)	45.07	(34.87,63.71)	48.80	(44.66,53.80)
MAS+LwF*	66.35	(81.83,55.79)	47.49	(69.69,36.02)	50.57	(89.01,35.32)
Ours$_M$	<u>73.21</u>	(72.61,73.83)	<u>57.97</u>	(63.07,53.63)	<u>72.24</u>	(84.57,63.04)
Ours$_E$	**79.60**	(75.33,84.37)	**61.41**	(65.99,57.43)	**73.70**	(85.98,64.49)

Table 1: **Results on CIFAR-100, CUB-200-2011, and Stanford Cars.** Bold and underline indicate the best and second best methods, respectively.

	# Task:2, # Class:50		# Task:5, # Class:20		# Task:10, # Class:10	
	$S\uparrow$	$(A\uparrow, F\uparrow)$	$S\uparrow$	$(A\uparrow, F\uparrow)$	$S\uparrow$	$(A\uparrow, F\uparrow)$
Vanilla	55.87	(55.21,56.55)	51.79	(39.66,74.62)	37.88	(25.41,74.41)
LwF	9.02	(74.69,4.80)	17.23	(79.05,9.67)	22.50	(80.74,13.07)
LwF*	54.64	(76.44,42.52)	68.24	(81.32,58.79)	63.62	(82.29,51.85)
EWC	58.58	(56.73,60.55)	48.57	(36.54,72.42)	34.91	(23.07,71.70)
EWC*	57.17	(56.25,58.13)	49.61	(36.58,77.08)	36.90	(23.68,83.52)
EWC*+LwF*	53.51	(77.11,40.98)	67.64	(81.20,57.96)	69.17	(74.11, 64.85)
MAS	55.44	(54.42,56.49)	47.46	(34.89,74.17)	35.26	(23.25,72.96)
MAS+LwF*	56.54	(76.85,44.72)	66.35	(81.83,55.79)	70.83	(74.63,67.41)
Ours$_M$	<u>70.08</u>	(74.89,65.84)	<u>73.21</u>	(72.61,73.83)	<u>71.63</u>	(68.56,75.00)
Ours$_E$	**74.02**	(74.93,73.14)	**79.60**	(75.33,84.37)	**76.01**	(67.93,86.26)

Table 2: **Results on CIFAR-100 for Varying Number of Tasks/Classes.** Bold and underline indicate the best and second best methods, respectively.

5.2 Comparative Results

Overall Results: Table 1 shows the comparative results of all the methods. We can clearly see that Ours$_E$ is the best and Ours$_M$ is the second best among all the methods in S_F. No other method that is better in terms of both A and F. This is mainly due to the advantage of our mnemonic codes; as we verified in our preliminary analysis, the codes enable accurate control over whether each class should be retained or forgotten on a class-by-class basis.

Per-Task Results: To visualize the performance changes over the tasks, we show S_k, A_k and F_k at each task in Fig. 4[5]. We can see that Ours$_E$ consistently achieves the best S on all the datasets. We can draw several observations from the results. First, any single existing method (like LwF, EWC, and MAS) cannot work well. While most of the methods maintain satisfactory performance in terms of A_k, Vanilla, EWC, and EWC* suffer from catastrophic forgetting, showing a decrease in accuracy with each new task added. These three methods look better in F_k than the other baselines, however, they forget all the classes whether they are in the preservation set or the deletion set, which is not desirable behavior. Conversely, LwF remembers all the classes, and thus has high A_k but sacrifices F_k. Second, even a combination of the existing methods, namely EWC*+LwF*, cannot yield satisfactory performance. This indicates that a straightforward idea to apply the strong regularization terms only to the preservation set is not sufficient to maintain adequate performance. These observations emphasize the difficulty of maintaining both A_k and F_k together. Unlike these methods, Ours$_E$ shows consistently high accuracy in both A_k and F_k. This proves the effectiveness of our mnemonic code and learning strategy in overcoming the new problem.

Results for Varying Number of Tasks/Classes: Table 2 shows the results on CIFAR-100 under the various numbers of tasks/classes. Ours is the best in S in all the cases. We also evaluated the performance of the methods when the ratio of the number of classes in the deletion set to that of all the classes, r_{del}, is varied in the range from 0.1 to 0.9. From the results shown in Table 3, S of our two methods are higher than those of the other methods for all the ratios. These results show the strong robustness of the proposed method to the various settings.

5.3 Sensitivity Analysis

We analyze the sensitivity of the performance to the hyperparameters, including the weight for the selective forgetting loss $\mathcal{L}_{SF}$ and the mnemonic loss $\mathcal{L}_M$. We also evaluate the performance with different types of mnemonic codes.

Effectiveness of $\mathcal{L}_{SF}$: Figure 5 shows the performance for various γ_{SF} in Eq. (5). First, as γ_{SF} is decreased, the performance decreases. This suggests that $\mathcal{L}_{SF}$ has a significant contribution to improving the performance. For larger γ_{SF}, the performance is high and stable, indicating that tuning of the the value is not severe.

Effectiveness of $\mathcal{L}_M$: We compared the performance of the proposed method with and without the mnemonic loss $\mathcal{L}_M$. The results are shown in the left side of Table 4. It clearly shows that the loss $\mathcal{L}_M$ significantly improves the performance, demonstrating the effectiveness of $\mathcal{L}_M$.

Mnemonic Code Choice: We evaluated the effectiveness of our choice for the mnemonic code, i.e., the random pattern

[5]Due to space limitations, we report only the results of Vanilla, EWC, LWF, EWC*, LWF*, and Ours$_E$ in this figure.

	$r_{del} = 0.1$		$r_{del} = 0.5$		$r_{del} = 0.9$	
	$S\uparrow$	$(A\uparrow, F\uparrow)$	$S\uparrow$	$(A\uparrow, F\uparrow)$	$S\uparrow$	$(A\uparrow, F\uparrow)$
Vanilla	46.20	(31.63,85.62)	52.58	(40.97,73.40)	69.23	(68.00,70.50)
LwF	16.13	(77.80,9.00)	14.96	(79.97,8.25)	13.51	(81.79,7.36)
LwF*	70.31	(79.14,63.25)	65.54	(83.30,54.02)	71.91	(88.50,60.56)
EWC	44.13	(30.43,80.25)	48.29	(36.37,71.85)	67.06	(66.79,67.33)
EWC*	43.19	(29.17,83.12)	52.98	(40.78,75.57)	67.88	(66.46,69.36)
EWC*+LwF*	67.04	(78.72,58.37)	70.07	(83.85,60.17)	75.47	(89.18,65.42)
MAS	45.52	(30.55,89.25)	51.85	(40.72,71.35)	68.16	(68.36,67.97)
MAS+LwF*	65.16	(77.43,56.25)	71.23	(82.02,62.95)	77.24	(88.75,68.37)
Ours$_M$	77.80	(76.99,78.62)	79.88	(79.45,80.32)	81.46	(87.46,76.24)
Ours$_E$	**83.46**	(75.87,92.75)	**83.16**	(81.80,84.57)	**83.68**	(86.57,80.97)

Table 3: **Results on CIFAR-100 for Various Ratio r_{del} of Deletion Set.**. Bold and underline indicate the best and second best methods, respectively.

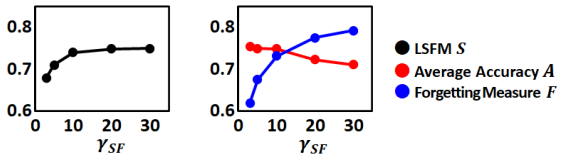


Figure 5: **Performance Sensitivity to γ_{SF}.** Left: LSFM (black line), Right: Average accuracy (red line) and Forgetting measure (blue line).

	$S\uparrow$	$(A\uparrow, F\uparrow)$
w/o $\mathcal{L}_M$	66.16	(56.71, 79.39)
w/ $\mathcal{L}_M$	74.02	(74.93, 73.14)

	$S\uparrow$	$(A\uparrow, F\uparrow)$
Mean	74.91	(73.20, 76.71)
Random	74.02	(74.93, 73.14)

Table 4: **Effectiveness of Mnemonic Loss $\mathcal{L}_M$ and Mnemonic Code.** Left: Effectiveness of the mnemonic loss, Right: Performance with different mnemonic code types.

code, by comparing it with the average code which is the average image of each class. The right side of Table 4 shows the results. We can see that the performance of these two methods is highly comparable. As we discuss in Sec. 4.1, the random code has several advantages compared with the average code. The results further emphasizes the merit of using the random code for lifelong learning with selective forgetting.

6 Conclusion

We opened up a new framework for lifelong learning called Learning with Selective Forgetting (LSF), which allows a model to continuously learn from new tasks while selectively forgetting undesirable class information. Our key contribution was the proposal of a simple and effective idea called mnemonic code. The code is a class-specific random signal embedded in each sample of the same class, which makes it possible to control the remembering and forgetting of the arbitrary class without using the original samples. Thorough experiments proved that our method could achieve significantly better performance than existing methods on this new problem. We believe that this paper will bring a new and practical direction of lifelong learning to the community and give the first baseline for the new problem.

References

[Abadi *et al.*, 2016] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proc. CCS*, pages 308–318, 2016.

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

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

[Bourtoule *et al.*, 2019] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019.

[Cao and Yang, 2015] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proc. S&P*, pages 463–480, 2015.

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

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

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

[Gilad-Bachrach *et al.*, 2016] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proc. ICML*, pages 201–210, 2016.

[Ginart *et al.*, 2019] A Ginart, M Guan, G Valiant, and J Zou. Making ai forget you: Data deletion in machine learning. In *Proc. NeurIPS*, 2019.

[Golatkar *et al.*, 2020] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proc. CVPR*, pages 9304–9312, 2020.

[Golatkar *et al.*, 2021] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proc. CVPR*, 2021.

[Guo *et al.*, 2020] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proc. ICML*, pages 3832–3842, 2020.

[He *et al.*, 2016] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

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

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

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

[Krause *et al.*, 2013] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proc. ICCVW*, pages 554–561, 2013.

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

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

[Liu *et al.*, 2020] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proc. CVPR*, page 12254, 2020.

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

[Mallya and Lazebnik, 2018] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proc. CVPR*, pages 7765–7773, 2018.

[Mallya *et al.*, 2018] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proc. ECCV*, pages 67–82, 2018.

[Nguyen *et al.*, 2020] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. In *Proc. NeurIPS*, pages 16025–16036, 2020.

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

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

[Shin *et al.*, 2017] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proc. NeurIPS*, pages 2990–2999, 2017.

[Speciale *et al.*, 2019a] Pablo Speciale, Johannes L Schonberger, Sing Bing Kang, Sudipta N Sinha, and Marc Pollefeys. Privacy preserving image-based localization. In *Proc. CVPR*, pages 5493–5503, 2019.

[Speciale *et al.*, 2019b] Pablo Speciale, Johannes L Schonberger, Sudipta N Sinha, and Marc Pollefeys. Privacy preserving image queries for camera localization. In *Proc. ICCV*, pages 1486–1496, 2019.

[Wah *et al.*, 2011] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[Wang *et al.*, 2018a] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.

[Wang *et al.*, 2018b] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proc. CVPR*, pages 5265–5274, 2018.

[Wang *et al.*, 2019] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proc. ICCV*, pages 5310–5319, 2019.

[Wu *et al.*, 2018] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. In *Proc. NeurIPS*, pages 5962–5972, 2018.

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

[Wu *et al.*, 2020] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *Proc. ICML*, pages 10355–10366, 2020.

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

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

[Zhang *et al.*, 2018] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. ICLR*, 2018.

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