Long-Tail Class Incremental Learning via Independent Sub-prototype Construction

Xi Wang, Xu Yang, Jie Yin, Kun Wei, Cheng Deng*
School of Electronic Engineering, Xidian University, Xi'an 710071, China
{wangxi6317, xuyang.xd, jieyin.xd, weikunsk, chdeng.xd}@gmail.com

Abstract

*Long-tail class incremental learning (LT-CIL) is designed to perpetually acquire novel knowledge from an imbalanced and perpetually evolving data stream while ensuring the retention of previously acquired knowledge. The existing method only re-balances data distribution and ignores exploring the potential relationship between different samples, causing non-robust representations and even severe forgetting in classes with few samples. In this paper, we constructed two parallel spaces simultaneously: **1) Sub-prototype space** and **2) Reminiscence space** to learn robust representations while alleviating forgetfulness. Concretely, we advance the concept of the sub-prototype space, which amalgamates insights from diverse classes. This integration facilitates the mutual complementarity of varied knowledge, thereby augmenting the attainment of more robust representations. Furthermore, we introduce the reminiscence space, which encapsulates each class distribution, aiming to constraint model optimization and mitigate the phenomenon of forgetting. The tandem utilization of the two parallel spaces effectively alleviates the adverse consequences associated with imbalanced data distribution, preventing forgetting without needing replay examples. Extensive experiments demonstrate that our method achieves state-of-the-art performance on various benchmarks.*

1. Introduction

Most deep learning literature focuses on learning a model on a fixed data stream [41, 42]. However, data in the real world is not static and even changes its distribution over time. Consider a scenario where a model trained on old data needs to be fine-tuned on new data, but the old data are unavailable due to privacy concerns; such fine-tuning will significantly degrade the model's performance in older data, known as **catastrophic forgetting**.

Continual learning endeavors to alleviate catastrophic

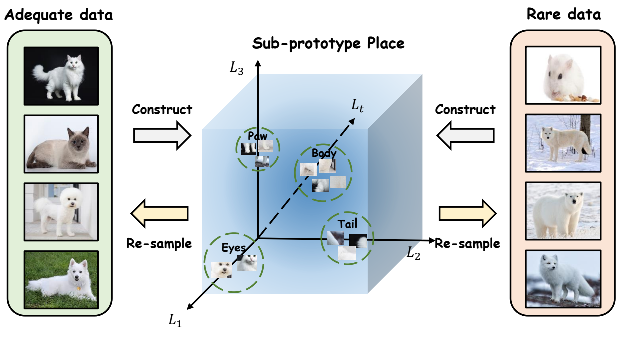

Figure 1. The sub-prototype space integration knowledge from different classes, and when the space is constructed, the features re-sampled from the space less affected by imbalanced data.

forgetting by maintaining a balance between the plasticity and stability of the model, which ensures that old knowledge remains preserved (stability to changes), while also accommodating the acquisition of new incoming data (plasticity to adapt) [24]. In the real world, most deep learning models need to tackle the forgetfulness caused by a continuous data stream. For this purpose, several works have been proposed to address catastrophic forgetting in many scenarios, including image classification [7], object detection [36], instance segmentation [10], and even domain adaptation [40]. Unfortunately, those continual learning methods assume the data distribution is balanced in different tasks. However, real-world data is often imbalanced, usually in a long-tailed distribution.

The deep learning on long-tailed data is often dominated by the majority classes (classes with amount samples), resulting in poor performance of the minority classes (classes with few samples) [46]. To tackle this problem, existing work attempts to expand data or change the network structure [35] and so on, both achieved good results. Surprisingly, continual learning on imbalanced data has yet to receive widespread attention. Due to the prevalence of imbalanced data distributions in the real-world, deep learning models can continually learn without catastrophically for-

*Corresponding author.

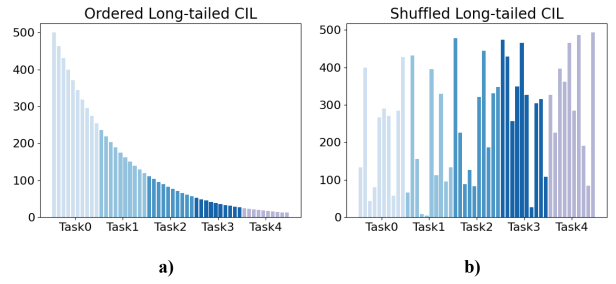

Figure 2. Illustration of long-tail class incremental learning (LT-CIL) scenarios. a) is Ordered LT-CIL and b) is Shuffled one.

getting that imbalanced data is more relevant to real-world needs. The paper [22] proposes long-tail class incremental learning and adds the balance training loss to the existed CL methods to tackle the LT-CIL. However, it ignores exploring the potential relationship between different classes, and there is still severe forgetting in classes with few samples. To tackle this problem, we first point out that imbalanced data distribution increases the difficulty of continual learning. Toward this end, our work aims to overcome two inevitable obstacles of LT-CIL: **1)** catastrophic forgetting: forgetting the knowledge of old classes while learning the new, due to the data in different tasks always being different. In this work, we nickname it **inter-task imbalance**, and **2)** long-tailed data distribution: in the same task, different classes have different sample sizes, and the distribution shows the imbalance, calling **intra-task imbalance**.

The existing method captures the hidden properties between different classes in the mini-batch to alleviate intra-task imbalance. Thus, although there is still a large gap between different classes, there are still shared sub-prototypes (For example, most mammals have shared characteristics such as claws, eyes, and noses). Our work considers learning different sub-prototypes as basis vectors to construct a sub-prototype space shared by different classes, as shown in Figure 1. With the constructed sub-prototype space, we can amalgamate insights from diverse classes. This integration facilitates the mutual complementarity of varied knowledge, thereby augmenting the attainment of more robust representations and mitigating the intra-task imbalance. At the same time, we propose a reminiscence space to store the data distribution when the number of tasks is increasing, expand the sub-prototype space to accommodate new knowledge while trying to keep the original space unchanged, and the newly constructed sub-prototype space has aggregated knowledge from different tasks at the same time, which alleviates the forgetfulness of previously learned thus alleviates the inter-task imbalance. In this way, the proposed method overcomes intra- and inter-task imbalance simultaneously. Our main contributions can be summarized as follows:

- We propose a novel and effective learnable sub-prototype space that simultaneously mitigates intra-task and inter-task imbalances in long-tail class incremental learning.
- We propose a reminiscence space to store data distribution, which prevents the model from collapsing under the influence of new knowledge and forgetting the learned old knowledge during training.
- We perform extensive experiments to demonstrate the effectiveness of our method, all achieving state-of-the-art in diverse settings.

2. Related Work

2.1. Long-tailed representation learning

The long-tailed data distribution is an enduring and pervasive problem in machine learning [12]. Long-tailed distribution is where a few categories (also called majority classes) contain many samples, while most categories (also called minority classes) have only a tiny number of samples. Such datasets make the deep learning network perform well in the majority classes and inefficiently in the minority classes, with a significant drop in overall recognition accuracy [26]. The current long-tailed representation methods mainly consist of Class Re-balancing [16, 37], Information Augmentation [5, 38] and Module Improvement [15]. Class Re-balancing methods seek to balance the samples of different classes during model training, but even if we balance them along one dimension, they can become unbalanced in another dimension [19]. Information-Augmentation-based methods seek to introduce additional information into model training so that the model performance can be improved in minority classes [8]. Besides Class Re-Balancing and Information Augmentation, existing methods also try to expand the network, such as changing the feature extractor, enhancing the model classifier, and proposing a new structure, but this will increase the number of parameters of the model [35].

2.2. Continual learning

Continual learning aims to continuously learn new knowledge from a never-ending data stream. The main challenge of continual learning is to learn without catastrophic forgetting: with the incoming new data, the model performance should not significantly degrade on the past learned tasks [27]. Current solutions for continual learning can be divided into three main categories: regularization-based methods [2, 20, 30, 45], replay-based methods [11, 33], and architecture-based methods [9, 25].

Regularization-based methods focus on weight regularization by estimating and preventing the important network weights from changing. Some methods add well-designed regularization terms into the loss function to constrain the update of the model parameters [30]. Some methods constrain model changes by the gradient [13, 23]. The most

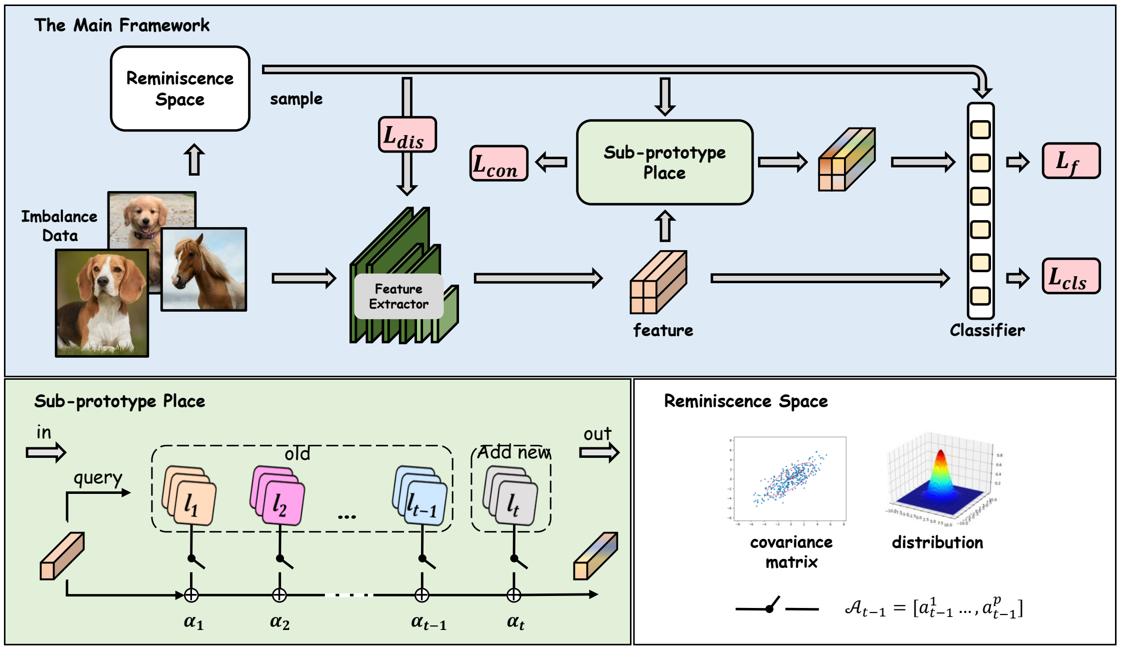

Figure 3. The overall framework of our method is composed of the sub-prototype place and reminiscence space. Specifically, sub-prototype space consists of independent sub-prototype basis vectors, which integrate of different classes and mitigate the data imbalanced distribution. The reminiscence space regularizes the whole model with the feature distribution of each class.

popular method is knowledge distillation which constrains the new model as similar as possible to the old [39]. The difference between Regularization-based methods is the way to compute the importance of the parameters and constraints on the model can be applied directly to the weights, predicted probabilities, or gradients.

The replay-based method attempts to store old data for replay. Most of these methods save the raw data directly and adopt it alongside the current data in the learning of new tasks [33]. The difference between saving-raw-data methods is how to choose the samples, and there is random probability method [17] and meta-learning methods [14, 28]. In addition to storing real data, some works try to generate samples by generating models by saving the distribution of old data, which takes data privacy into account [6].

Architecture-based methods tend to continually extend the network structure for different tasks of incremental learning [25, 32, 43]. Existing methods have tried to make the model have multiple classifiers. However, as the incremental learning task continues to increase and the demands on the model become higher, it is clear that continually extending the model structure is highly impractical.

However, these methods barely consider imbalanced data distribution, which can cause a drop in overall accuracy by not paying attention to the performance degradation of the minority classes during continuous learning. Our proposed method explores relationships between different classes, concerns the more severe forgetting in the minority classes, and takes advantage of the rich knowledge of the majority classes to assist in the minority classes learning, solving inter-class forgetting and intra-class data imbalance, enabling continual learning on imbalanced data.

3. Method

Our goal is to enable the network to learn multiple tasks sequentially with imbalanced data streams. In this section, we present the problem definition of long-tail class incremental learning. After that, we detail the proposed method.

3.1. Preliminary

Typically, we consider a supervised class incremental learning setting where a model needs to learn $\mathcal{T}$ different tasks in turn. Each task contains different classes, and the classes between tasks are disjoint: $\mathcal{C}^0 \cap \mathcal{C}^1 \cap \dots \cap \mathcal{C}^{\mathcal{T}} = \emptyset$ and $\mathcal{C}^t$ is the class set of task t. At each task $t \in \{1, \dots, \mathcal{T}\}$, $(x, y) \in \mathcal{D}^t$ denotes the training sample, where x is a sample in the input space $\mathcal{X}$, y is its corresponding label and $\mathcal{D}$ is sample space. Different from the previous class incremental learning where the samples per class are equal, long-tail class incremental learning has imbalanced data distribution in each task, which means samples per class are unequal. The imbalanced distribution is parameterized by ρ, which is the ratio between the most and least sample size. For example, when $\rho = 0.01$, the most class sample size is

100 times to the least. Meanwhile, when $\rho = 1$, the most class sample size equals the least, which means the balanced data distribution. Given a random imbalance ratio ρ, after subjecting the training data to an imbalanced distribution, we follow the existing work [22] to propose two different LT-CIL scenarios:

- **Ordered LT-CIL**. The dataset is sequentially divided into different tasks, as shown in Figure 2 a). There is an imbalanced distribution over the complete dataset, and the total number of samples within the tasks is decreasing.
- **Shuffled LT-CIL**. The majority and minority classes randomly belong to any task. While there is still an imbalanced distribution in each task, the total number of samples from different tasks shows a random trend as shown in Figure 2 b).

To facilitate analysis, we divide the network into two parts: a feature extractor and a unified classifier. Specifically, the feature extractor $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$, parameterized by θ, maps the input x into a feature vector $z = f_\theta(x) \in \mathbb{R}^d$ in the deep feature space $\mathcal{Z}$; the unified classifier $g_\varphi : \mathcal{Z} \rightarrow \mathbb{R}^{\mathcal{C}^{1:t}}$ parameterized by φ, produces a probability distribution $g_\varphi(z)$ as the prediction for x. The model has to classify all seen classes at any point in training.

Our proposed method comprises two parallel spaces: the sub-prototype space and the reminiscence space. In the next, we present the specific definitions of these two spaces.

3.2. Sub-prototype space

The proposed sub-prototype space comprises two primary processes: space construction and feature re-sampling.

3.2.1 Space construction

We propose a sub-prototype space (SS) with independent sub-prototype basis vectors, which are obtained by learning from the training data, thus constructing the different semantic information of the subspace to alleviate both intra- and inter-task imbalance. The main framework is shown in Figure 3. Assume the feature extractor output is $z^t = f_\theta(x^t) \in \mathbb{R}^{B\times D}$, where x is the input images, B is the batch size and D is the feature dimension and t is the task index. The SS takes the intermediate features z^t as input and then re-samples the features in sub-prototype space using sub-prototype basis vectors:

$$\widetilde{z^t} = \mathrm{SS}\left(z^t\right) \in \mathbb{R}^{B\times D}. \tag{1}$$

Specifically, sub-prototype space consists of M^t different basis vectors in task t. The number of basis vectors increases by n when a new class is to be learned. Thus $M^0 = n \times N^0$ and $M^t = M^{t-1} + n \times N^t (t \geq 1)$, where N^t denote the number of new classes in task t. When it comes to learning the new task t, the basis vectors can be represented as:

$$L_t = \left[L_{t-1}; l_1, \ldots, l_n\right], \tag{2}$$

where $l_n \in \mathbb{R}^D$. When learning a newly arriving task t, the dimension of the sub-prototype space is $L_t \in \mathbb{R}^{M^t\times D}$. To construct the sub-prototype space, independent basis vectors must be learned from the training data. When a new set of learned features z^t is input, sub-prototype space first projects the features into the existing basis vectors, queries the correlation between the input features and the basis vectors using the query function Q and then obtains correlation matrix A_t between the sub-prototype basis vectors and the input features.

$$A_t = Q(z^t, L_t) = \left[a_{ij}\right]_{i\in B, j\in M^t}, \tag{3}$$

where $A_t \in \mathbb{R}^{B\times M^t}$, a_{ij} indicates the correlation factor between the ith feature and the jth sub-prototype basis vector, we regard the correlation factor as the component of the feature projection to this basis vector. According to the correlation matrix A_t, the input features can be projected into the space. Note that since the number of basis vectors in the space is larger than the number of classes, to prevent the projection of features from spreading out too much to get the better sub-prototype, we only select the most relevant k basis vectors with correlation factor in the actual projection process. At the same time, due to the imbalanced data distribution within the task, the selection based on the correlation factor may lead to a part of the basis vectors being selected frequently while others are selected too infrequently or never selected, resulting in the space being skewed towards a certain dimension, so we add the controlling factor $\mathcal{H} = [\eta_i]_{i\in M^t}$. During the training process of each task, the number of selected times of vector i is counted as n_i, and the controlling factor $\eta_i = e^{-n_i}$. Thus, the actual correlation matrix A_t is:

$$\begin{gathered} A_t = Q(z^t, L_t) = \left[a_{ij}\right]_{i\in B, j\in M^t} \\ a_{ij} = \begin{cases} a_{ij} & \text{if} \quad a_{ij} \in \left[\mathcal{H}\cdot A[i:]\right]_{top_k} \\ 0 & \text{else} \end{cases} \end{gathered} \tag{4}$$

After obtaining the correlation matrix, the projection of the input features in the sub-prototype space can be expressed as:

$$\widetilde{\mathcal{Z}} = A_t\, {L_t}^\top = \left[\ \widetilde{z_1}, \quad \cdots \quad, \widetilde{z_B}\ \right]^\top, \tag{5}$$

Note that the sub-prototype basis vectors in the space are trainable, and in order to keep the semantic information before and after the feature projection consistent, we use the L2 Norm to constrain the updating of the basis vectors:

$$\mathcal{L}_{con} = \|\widetilde{\mathcal{Z}} - \mathcal{Z}\|_2, \tag{6}$$

Therefore, the loss in the construction phase consists of two parts: the classification loss based on supervised information $\mathcal{L}_{cls}(\phi; x, y) = \mathcal{L}_{ce}(\phi(x), y)$, where $\mathcal{L}_{ce}$ is the cross-entropy loss and the construction loss L_{con}. Thus the entire training loss is:

$$\mathcal{L}_1 = \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_{con}. \tag{7}$$

In the construction phase, we use cosine similarity as the query function. When the training is over, we simultaneously construct a sub-prototype space of different classes and a learned model corresponding to the task. In the re-sampling phase, we use the constructed sub-prototype space to fine-tune the model and mitigate the impact of imbalanced data on the model.

3.2.2 Feature re-sampling

In the feature re-sampling phase, the sub-prototype space is fixed, and only the feature extractor f_θ and the classifier g_φ are trained. When the training data stream re-pass through the model and get the corresponding features z^t, firstly, we re-calculate the correlation factor between the features and the basis vectors in the sub-prototype space. For a random feature, based on the magnitude of the correlation factor, we select the top two most correlated directions: l_{top_1}, l_{top_2} and regard it as the re-sampling base direction. Then, combined with the remaining h basis vectors randomly selected to generate the augmentation features in the sub-prototype space. Since the imbalanced distribution within the task, the number of re-sampled features is considered to be dynamically adjusted according to the frequency of sample occurrences, the number of i^{th} class in task t correspondingly re-sampled features $N_{re,i}^t = N_{\max}^t - N_i^t + 10$, where $N_{\max}^t$ indicates the max sample size in task t and N_i^t is the sample size of i^{th} class in task t.

$$\hat{z} = l_{top_1} + l_{top_2} + \mathbb{I}\left[\mathrm{C}_{M^t-2}^h \sum_{i=1}^h l_i\right], \tag{8}$$

where $\left[\mathrm{C}_{M-2}^h \sum_{i=1}^h l_i\right]$ denotes all the combinations of any h permutations of the remaining $M^t - 2$ basis vectors, excluding the dimensions of the top two correlation factors, C is combination and $\mathbb{I}$ is indicator matrix consists of 0 and 1 with the same dimension of combinations. Thus, the features through the classifier become $\mathcal{Z}_{new} = \left[\mathcal{Z}, \hat{\mathcal{Z}}\right]$ and the re-sampling features have the same label as the original feature. Thus, the loss in the re-sampling phase is the classification loss based on supervised information with a cross-entropy loss: $\mathcal{L}_f(\phi; x, y) = \mathcal{L}_{ce}(\mathcal{Z}_{new}, Y_{new})$.

3.3. Not to forget: Reminiscence space

The number of sub-prototype basis vectors in the sub-prototype space increases with the number of tasks, but in the construction phase, whether existing or new, all the basis vectors need to be trained at the same time, if there are no constraints, the existing basis vectors will undoubtedly forget what has been learned before. To prevent the previously constructed space from being corrupted by the new knowledge without storing any data, we propose the reminiscence space to alleviate forgetfulness.

Specifically, when each task is trained over, we calculated the class feature centroid $\mu_k = \frac{1}{n_k}\sum_{i=1}^{n_k} f_\theta(x_i)$ and the corresponding covariance matrix Σ_k, where each element is the covariance between the two features in the same class. We use the class-specific statistics to form a multivariate normal distribution $\mathcal{N}_k = \mathcal{N}(\mu_k, \Sigma_k)$ for each class. Meanwhile, we calculated the mean of the correlation matrix in sub-prototype space for each class:

$$\mathcal{A}^k = \frac{1}{n} \sum_{\{\triangle_k | y \in \text{ class } k\}} \triangle_k, \tag{9}$$

where $\triangle_k = A_t\left[a_{k:}\right]$ indicates the correlation matrix between class k and all basis vectors in space. Thus, we form the reminiscence space for class k as $\mathcal{S}_k = \left(\mathcal{N}_k, \mathcal{A}^k\right)$. To prevent the previously constructed sub-prototype space from being corrupted by the new knowledge while training a new task, we perform the distillation loss on the sub-prototype basis vectors using $\mathcal{A}^k$. Since the number of basis vectors in the sub-prototype space corresponding to different tasks is different, the dimensions of the correlation matrix stored in the reminiscence space are different, so we need to mask the dimension of redundancy when distillation:

$$\mathcal{L}_{dis1} = \sum_{i=1}^{N_t} \left\|\mathcal{A}^i - Q\left(\mu_i, L_t\right)\left[: I_i\right]\right\|_2, \tag{10}$$

where I_i indicates the dimension of $\mathcal{A}^i$.

To maximize the effectiveness of our proposed method, we sample features from the reminiscence space $\widetilde{v}^k \sim \mathcal{S}_k$ and compute the cross-entropy loss of the model: $\mathcal{L}_{dis2} = \mathcal{L}_{ce}(\widetilde{v}^k, k)$. Thus, the two phases can be briefly described:

Space construction: The sub-prototype space is constructed while training the model. The training loss is:

$$\mathcal{L}_1 = \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_{con} + \lambda_2 \mathcal{L}_{dis1}. \tag{11}$$

Feature re-sampling: The feature extractor and classifier are fine-tuned, and the total training loss is:

$$\mathcal{L}_2 = \mathcal{L}_f + \lambda_3 \mathcal{L}_{dis2}. \tag{12}$$

4. Experiments

4.1. Experiments Setttings

Datasets. Following previous work [22], we perform our experiments on CIFAR100 and ImageNet-Subset datasets

Table 1. **Results on Shuffled LT-CIL**. We compare our method with Baselines and previous methods.

Methods	Memory Size	Shuffled LT-CIL											
		CIFAR100						ImageNet-Subset					
		$\rho = 0.01$		$\rho = 0.05$		$\rho = 0.1$		$\rho = 0.01$		$\rho = 0.05$		$\rho = 0.1$	
		5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks
Baseline	0	11.3	7.3	13.6	8.1	13.9	8.2	12.9	11.2	15.3	13.1	16.7	12.7
LDAM[4]	0	11.5	11.4	15.2	14.3	19.8	18.2	15.4	15.2	19.4	19.0	22.6	22.1
BalPoE[1]		18.9	17.6	20.5	20.0	26.1	25.4	17.8	17.3	22.6	21.4	25.3	24.3
MDCS[47]		18.2	16.3	19.5	19.1	24.3	23.7	16.9	16.2	22.4	21.4	25.6	24.1
EWC[18]	0	28.7	25.3	33.1	31.9	40.6	39.7	30.8	30.4	35.6	34.7	43.8	43.6
LwF[21]		29.3	25.1	34.3	33.5	41.2	41.0	31.6	31.0	36.1	35.4	44.4	43.9
SDC[44]		32.7	29.6	35.2	34.1	42.9	42.3	33.9	33.4	39.4	38.1	45.9	45.1
PASS[49]		33.6	31.8	37.9	35.5	43.2	42.1	34.2	33.8	39.9	38.5	46.2	45.7
IL2A[48]		35.1	36.2	43.9	39.4	50.2	49.3	40.5	39.2	44.2	43.7	53.4	52.7
SAVC[34]		34.4	32.3	38.3	35.9	43.1	42.0	35.3	34.9	40.1	39.6	48.3	47.6
iCaRL[29]	1000	31.5	30.5	40.2	39.1	46.5	45.9	35.4	34.6	42	41.3	48.2	47.4
TwF[3]		34.2	33.8	42.3	42.1	49.3	48.7	38.6	38.1	43.6	43.3	52.2	51.7
SCoMMER[31]		35	35.2	43.4	42.3	49.9	49.1	39.3	38.9	44.9	44.1	52.9	52.6
LUCIR+LWS[22]		37.2	36.9	45.2	45.0	51.9	51.2	43.1	42.3	47.3	47.1	54.7	54.1
Ours	0	**40.2**	**39.4**	**47.3**	**47.0**	**53.6**	**53.1**	**45.3**	**44.8**	**49.2**	**48.9**	**56.2**	**55.4**

Table 2. **Results on Orderd LT-CIL**. We compare our method with Baselines and previous methods.

Methods	Memory Size	Ordered IL-CIL											
		CIFAR100						ImageNet-Subset					
		$\rho = 0.01$		$\rho = 0.05$		$\rho = 0.1$		$\rho = 0.01$		$\rho = 0.05$		$\rho = 0.1$	
		5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks	5 tasks	10 tasks
Baseline	0	16.5	15.4	18.9	15.3	20.1	15.1	20.3	18.3	21.7	18.9	23.5	21.1
LDAM[4]	0	20.1	19.4	24.8	23.1	33.8	33.0	19.7	18.6	27.3	26.4	35.7	34.6
BalPoE[1]		24.9	24.0	28.3	27.5	37.3	36.1	23.5	23.0	30.4	28.5	36.2	35.6
MDCS[47]		24.3	23.7	27.9	27.1	37.0	35.4	23.1	22.8	30.1	28.3	35.9	35.5
EWC[18]	0	32.1	31.6	36.7	36.1	43	42.1	35.7	31.2	36.4	36.1	46.7	46.1
LwF[21]		32.8	31.9	36.4	35.8	43.5	42.7	36.1	33.4	36.7	35.7	47.1	46.5
SDC[44]		34.9	34.5	39.2	38.8	45.7	45.0	43.2	42.0	44.5	44.1	48.1	47.2
PASS[49]		35.8	35.2	39.8	39.3	46.1	45.5	43.9	42.6	45.0	44.7	48.9	47.3
IL2A[48]		40.6	40.9	43.2	42.9	52.3	52.0	47.2	46.9	50.6	50.4	53.7	53.4
SAVC[34]		36.1	35.8	40.0	39.6	46.7	46.2	45.5	45.0	46.8	46.3	50.6	49.9
iCaRL[29]	1000	36.4	36.2	40.2	39.4	48.7	48.5	43.6	42.7	47.9	47.5	51.6	51.3
TwF[3]		40.1	39.8	42.7	42.1	51.1	51.0	46.1	45.4	49.4	48.9	52.8	52.3
SCoMMER[31]		41.2	41.0	43.8	42.5	52.9	52.4	47.0	46.5	51.1	50.6	54.3	53.6
LUCIR+LWS[22]		42.3	42.1	45.7	45.3	54.2	53.4	50.3	49.6	53.1	52.7	56.2	56.0
Ours	0	**44.8**	**44.3**	**47.8**	**47.4**	**54.6**	**54.3**	**52.7**	**52.6**	**55.7**	**55.2**	**57.4**	**57.2**

with 100 classes. For both datasets, we split them into different settings: $50 + 5 \times 10, 50 + 10 \times 5$. For example, $50 + 5 \times 10$ means the first task has 50 classes and has 5 incremental phases, each incremental phase has 10 classes. At the same time, we divide the dataset into Ordered LT-CIL and Shuffled LT-CIL according to different imbalanced rate.

Implementation Details. For both datasets, our baseline feature extractor is ResNet18. The training batch size is 64 with 121 epochs in the construction phase and 81 epochs in the re-sampling phase. The learning rate is 0.001 and will be reduced by a factor 10 at every 40 epoch. In our experiments, $n = 2$, $k = 10$ and $h = 4$. The values of the hyperparameters are as follows: $\lambda_1 = 6 \times 10^{-4}$, $\lambda_2 = 10$ and $\lambda_3 = 10$. For CIFAR100, the class sample size less than 100 is the minority class, while for ImageNet-Subset, class sample size less than 200 is the minority class. For page limitation, we have placed the sensitivity analysis of the parameters, and the analysis of the values that need to be considered in the supplementary material.

Metrics. We use the standard metrics in the continual learning methods to measure performance: Average Accuracy, which calculates all seen classes' accuracy. Let $a_{i,j}$ be the accuracy of the model on the testing set of task j after the model is trained from task 1 to task i, Average Accuracy can be calculated:

$$\text{Average Accuracy } (A_T) = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}. \tag{13}$$

Comparison Methods. We compare our method with the existing methods including both class incremental methods without replay data: EWC [18], LwF [21], IL2A [48],

Table 3. **Ablation study**. The experiment setting is CIFAR100 with 5 incremental tasks and $\rho = 0.01$.

	Components					Accuracy
	$\mathcal{L}_{cls}$	$\mathcal{L}_{con}$	$\mathcal{L}_{f}$	$\mathcal{L}_{dis1}$	$\mathcal{L}_{dis2}$	
a)	√					11.3
b)	√	√				10.4
c)	√	√	√			34.7
d)	√	√	√		√	36.3
e)	√	√	√	√		38.9
f)	√	√	√	√	√	40.2

class incremental methods with replay data: iCaRL [29], LUCIR+LWS [22], TwF [3], SCoMMER [31], and class incremental method based on prototype: SDC [44], PASS [49] as well as the long-tail methods: LDAM [4], BalPoE [1], MDCS [47] and method for few-shot incremental learning: SAVC [34]. All comparison experiments use the same ResNet18 as backbone and for the methods that needs to store data, the memory size is fixed to 1000.

4.2. Experimental Results

It is noted that the Baseline refers to distillation only, and the long-tail method is applied directly to the Baseline. We calculated the average accuracy of all tasks and comparative results are presented in Table 1 and Table 2.

Shuffled LT-CIL results. The experimental results are shown in Table 1. When the benchmark is CIFAR100, imbalance rate 0.01, and 5 incremental tasks, our method obtained an improvement of 3% over the LUCIR+LWS, even without any replay data. and when increasing the number of tasks to 10, our method has 2.5% improvement to the best before. After adjusting the learning rate to 0.05 to make the training data more, our method also achieves optimal performance, with 2.1% and 2.0% improvements on the 5 and 10 tasks, respectively. Meanwhile, on the ImageNet-Subset benchmark, we experimented with the same settings, and our method achieved the same stunning results, with a higher accuracy of 2.1% than the previous best method when the imbalance rate was 0.01 for a total of 5 incremental tasks. In all other settings, our method achieved the best performance.

Orderd LT-CIL results. The experiment results are shown in Table 2. When the benchmark is CIFAR100, the imbalance rate is set to 0.01 and there are 5 incremental tasks, our method achieves accuracy 44.8%, 2.5% improvement over the LUCIR+LWS, and when the tasks are extended to 10, 2.2% improvement over the previous method, and the improvement also exists under the other settings. When switching the benchmark to ImageNet-Subset, the same stunning performance was achieved.

4.3. Ablation Study

In this section, we discuss the effectiveness of each part in our method. The experiment setting is in Shuffled LT-CIL, CIFAR100, 5 incremental tasks, and the imbalance rate of 0.01. The results are shown in Table 3. As Table 3 shows, we can observe that only constructing the sub-prototype space makes the performance of the original model drop slightly to 10.4%, but when re-train the model with the re-sampling features, the performance improves to 34.7%, which proves what we emphasized in the introduction, that there exist shared sub-prototypes between different classes, and with the help of the knowledge integrating, the model can learn more robust representation. Meanwhile, in order to prevent the inter-task imbalance from leading to catastrophic forgetting, we propose a reminiscence space to store the feature distribution of each class, and after performing the constraints on the sub-prototype space and the classifier respectively, the performance of the model improves to 38.9% and 36.3% from 34.7%, which is a good proof of our effectiveness of the reminiscence space. Finally, by adding constraints on both the sub-prototype space and the classifier, the performance improves to 40.2% and obtains the best results in this setting, which proves that the two-parallel space is effective to the LT-CIL.

4.4. Further analysis

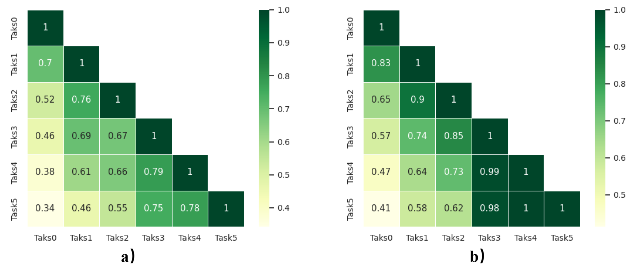

Figure 4. **Similarity matrix between models**. Experiments were performed in CIFAR100, Shuffled LT-CIL, $\mathcal{T} = 5$, $\rho = 0.01$. Compare the similarity between the models obtained at the end of each task. a) is the Baseline and b) is our method.

Similarity analysis. In order to better visualize the results of our method, we calculated the similarity of the models obtained after different tasks and presented them in the form of heat maps in Figure 4, a) is the Baseline result, and b) is our proposed method and experiments were performed in CIFAR100, Shuffled LT-CIL, $\mathcal{T} = 5$, $\rho = 0.01$. Since task 0 is the initial learning task, the model obtained after task 0 will theoretically have the maximum forgetting rate, so in the case of distillation only, the similarity with the model obtained after task 5 is only 0.34, meanwhile, after adding our proposed two-parallel space, the similarity improves to 0.41, and the improvement also exists in the other models which are evident that our method makes the model retain more knowledge learned before. This accumulation of similarity eventually leads to a qualitative leap in the performance of our method compared to the baseline.

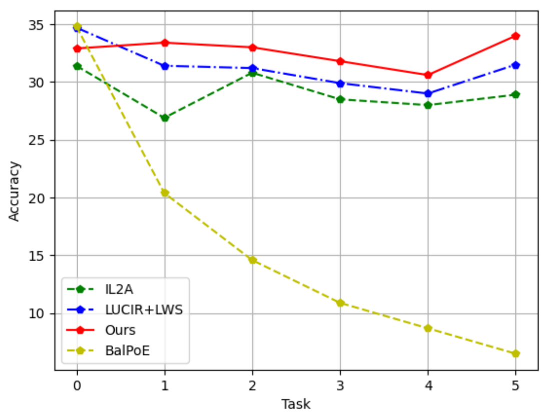

Figure 5. **Accuracy on minority classes**. The setting is the same as the similarity matrix, and in order to make the results more convincing, we chose methods that require replay data (LUCIR+LWS), do not require replay data (IL2A), and method for long-tail distribution (BalPoE).

Minority classes accuracy. The main issue to overcome in LT-CIL is the forgetting of the minority classes, it is necessary to focus on the classification performance of the minority classes separately. We present the minority classes' experimental results in graphs in Figure 5. Specifically, we compare the method with replay data (LUCIR+LWS), without replay data (IL2A), and the long-tail method (BalPoE), and the experiment setting are same as similarity analysis. Our method is lower than BalPoE and LUCIR+LWS in Task 0 but significantly higher than IL2A. When incremental tasks are to be learned, the performance degrades because BalPoE cannot alleviate the catastrophic forgetting. Meanwhile, LUCIR+LWS and IL2A can be stabilized within a certain accuracy range, but all of them are declining. At the same time, our method consistently maintains high classification performance on the minority classes due to the construction of the sub-prototype space to eliminate the effects of imbalanced data distribution.

Basis vectors selection frequency ablation study. The features are reconstructed using basis vectors in sub-prototype space, and since all majority and minority classes share these vectors, the rich knowledge of the majority classes can assist the minority classes in learning a more robust representation. During the reconstruction, to prevent some basis vectors from being chosen too frequent, which would result in tilting of the sub-prototype space in certain directions, we use a controlling factor to regulate dynamically. We add all basis vectors selection frequency in task 0 to prove that. Figure 6 a) shows the frequency of all class occurrences in task 0 and b) is the selection frequency of all basis vectors in task 0 (both are normalized); It is obvious that the basis vector selection has been much more balanced. Due to the approximate balance within the sub-prototype space, it can be used to solve the problem caused by imbalanced data distribution.

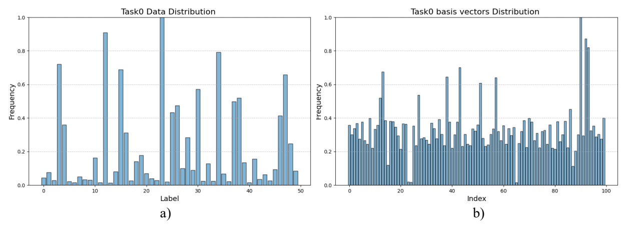

Figure 6. **Basis vectors selection frequency ablation study**. The setting is the same as the similarity matrix, and a) is the frequency of all class occurrences in task 0, b) is the selection frequency of all basis vectors in task 0.

5. Conclusion

In this work, we address two fundamental challenges in long-tail class incremental learning: intra-task imbalance due to data imbalanced distribution and inter-task imbalance due to forgetting what was learned before during incremental learning. We propose two parallel spaces: sub-prototype space, where the sub-prototypes of distinct classes serve as its basis vectors, and reminiscence space, comprised of the feature distribution of each class. The sub-prototype space amalgamates insights from diverse classes. It facilitates the mutual complementarity of varied knowledge, thereby augmenting the attainment of more robust representations and mitigating the impact caused by the imbalanced data distribution. Meanwhile, reminiscence space adds constraints to the learning process to prevent the model from catastrophic forgetting. Through the collaborative operation of these two spaces, we significantly alleviate the adverse effects associated with data imbalance and forgetting. Extensive experiments on different benchmark datasets demonstrate the effectiveness of our method.

6. Acknowledgement

This work was supported in part by the National Key Research and Development Program of China (No. 2023YFC3305600), Joint Fund of Ministry of Education of China (8091B022149, 8091B02072404), National Natural Science Foundation of China (62132016, 62171343, and 62201436), and Fundamental Research Funds for the Central Universities (ZDRC2102).

References

[1] Emanuel Sanchez Aimar, Arvi Jonnarth, Michael Felsberg, and Marco Kuhlmann. Balanced product of calibrated experts for long-tailed recognition. In *CVPR*, pages 19967–19977, 2023. 6, 7

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2

[3] Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *ECCV*, pages 692–709, 2022. 6, 7

[4] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019. 6, 7

[5] Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. In *ECCV*, pages 694–710, 2020. 2

[6] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. Gan memory with no forgetting. In *NeurIPS*, 2020. 3

[7] Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. In *NeurIPS*, 2019. 1

[8] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *ICCV*, pages 1851–1860, 2017. 2

[9] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pages 9285–9295, 2022. 2

[10] Yanan Gu, Cheng Deng, and Kun Wei. Class-incremental instance segmentation via multi-teacher networks. In *AAAI*, pages 1478–1486, 2021. 1

[11] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng. Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *CVPR*, pages 7442–7451, 2022. 2

[12] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21(9):1263–1284, 2009. 2

[13] Xu He and Herbert Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *ICLR*, 2018. 2

[14] Christian Henning, Maria Cervera, Francesco D'Angelo, Johannes Von Oswald, Regina Traber, Benjamin Ehret, Seijin Kobayashi, Benjamin F Grewe, and João Sacramento. Posterior meta-replay for continual learning. In *NeurIPS*, 2021. 3

[15] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *CVPR*, pages 5375–5384, 2016. 2

[16] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 2

[17] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *ECCV*, pages 411–428, 2020. 3

[18] 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. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 6

[19] Xuan Kou, Chenghao Xu, Xu Yang, and Cheng Deng. Attention-guided contrastive hashing for long-tailed image retrieval. In *IJCAI*, pages 1017–1023, 2022. 2

[20] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NeurIPS*, 2017. 2

[21] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2017. 6

[22] Xialei Liu, Yu-Song Hu, Xu-Sheng Cao, Andrew D Bagdanov, Ke Li, and Ming-Ming Cheng. Long-tailed class incremental learning. In *ECCV*, pages 495–512, 2022. 2, 4, 5, 6, 7

[23] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017. 2

[24] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. 1

[25] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, pages 7765–7773, 2018. 2, 3

[26] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *CVPR*, pages 864–873, 2016. 2

[27] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 2

[28] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *CVPR*, pages 13588–13597, 2020. 3

[29] Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. *IEEE Computer Society*, 2016. 6, 7

[30] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *NeurIPS*, 2018. 2

[31] Fahad Sarfraz, Elahe Arani, and Bahram Zonooz. Sparse coding in a dual memory system for lifelong learning. In *AAAI*, pages 9714–9722, 2023. 6, 7

[32] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4548–4557, 2018. 3

[33] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *AAAI*, pages 9630–9638, 2021. 2, 3

[34] Zeyin Song, Yifan Zhao, Yujun Shi, Peixi Peng, Li Yuan, and Yonghong Tian. Learning with fantasy: Semantic-aware virtual contrastive constraint for few-shot class-incremental learning. In *CVPR*, pages 24183–24192, 2023. 6, 7

[35] Jianfeng Wang, Thomas Lukasiewicz, Xiaolin Hu, Jianfei Cai, and Zhenghua Xu. Rsg: A simple but effective module for learning imbalanced datasets. In *CVPR*, pages 3784–3793, 2021. 1, 2

[36] Jianren Wang, Xin Wang, Yue Shang-Guan, and Abhinav Gupta. Wanderlust: Online continual object detection in the real world. In *ICCV*, pages 10829–10838, 2021. 1

[37] Tao Wang, Yu Li, Bingyi Kang, Junnan Li, Junhao Liew, Sheng Tang, Steven Hoi, and Jiashi Feng. The devil is in classification: A simple framework for long-tail instance segmentation. In *ECCV*, pages 728–744, 2020. 2

[38] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *NeurIPS*, 2017. 2

[39] Kun Wei, Cheng Deng, Xu Yang, and Maosen Li. Incremental embedding learning via zero-shot translation. In *AAAI*, pages 10254–10262, 2021. 3

[40] Kun Wei, Xu Yang, Zhe Xu, and Cheng Deng. Class-incremental unsupervised domain adaptation via pseudo-label distillation. *IEEE Trans. Image Process.*, 2024. 1

[41] Jiexi Yan, Lei Luo, Cheng Deng, and Heng Huang. Adaptive hierarchical similarity metric learning with noisy labels. *IEEE Trans. Image Process.*, 32:1245–1256, 2023. 1

[42] Jiexi Yan, Zhihui Yin, Erkun Yang, Yanhua Yang, and Heng Huang. Learning with diversity: Self-expanded equalization for better generalized deep metric learning. In *ICCV*, pages 19365–19374, 2023. 1

[43] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 3

[44] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, pages 6982–6991, 2020. 6, 7

[45] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017. 2

[46] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021. 1

[47] Qihao Zhao, Chen Jiang, Wei Hu, Fan Zhang, and Jun Liu. Mdcs: More diverse experts with consistency self-distillation for long-tailed recognition. In *ICCV*, pages 11597–11608, 2023. 6, 7

[48] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. In *NeurIPS*, 2021. 6

[49] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880, 2021. 6, 7