DYSON: Dynamic Feature Space Self-Organization for Online Task-Free Class Incremental Learning

Yuhang He[1*], Yingjie Chen[1*], Yuhan Jin[2], Songlin Dong[1], Xing Wei[2], Yihong Gong[1,2†]
[1] College of Artificial Intelligence, Xi'an Jiaotong University
[2] School of Software Engineering, Xi'an Jiaotong University
{heyuhang@xjtu.edu.cn},{cyj1450670826,3122358042,dsl972731417}@stu.xjtu.edu.cn
{xingwei,yhgong}@mail.xjtu.edu.cn

Abstract

In this paper, we focus on a challenging Online Task-Free Class Incremental Learning (OTFCIL) problem. Different from the existing methods that continuously learn the feature space from data streams, we propose a novel compute-and-align paradigm for the OTFCIL. It first computes an optimal geometry, i.e.*, the class prototype distribution, for classifying existing classes and updates it when new classes emerge, and then trains a DNN model by aligning its feature space to the optimal geometry. To this end, we develop a novel Dynamic Neural Collapse (DNC) algorithm to compute and update the optimal geometry. The DNC expands the geometry when new classes emerge without loss of the geometry optimality and guarantees the drift distance of old class prototypes with an explicit upper bound. On this basis, we propose a novel DYnamic feature space Self-OrganizatioN (DYSON) method containing three major components, including 1) a feature extractor, 2) a Dynamic Feature-Geometry Alignment (DFGA) module aligning the feature space to the optimal geometry computed by DNC and 3) a training-free class-incremental classifier derived from the DNC geometry. Experimental comparison results on four benchmark datasets, including CIFAR10, CIFAR100, CUB200, and CoRe50, demonstrate the efficiency and superiority of the DYSON method. The source code is released at https://github.com/isCDX2/DYSON.*

1. Introduction

Taking a continuous data stream as input, Online Task-Free Class Incremental Learning (OTFCIL) [19, 34] aims to incrementally learn newly emerged classes by online learning without task boundaries and identifiers. Compared to the classical class-incremental learning [12, 27, 32, 45, 49] that splits a data stream into a series of subsets (namely sessions or tasks), where each subset contains the samples of certain classes and different subset classes are mutually excluded, the OTFCIL is a practical but more challenging problem wherein the samples are one-shot provided and old and new class knowledge are blended.

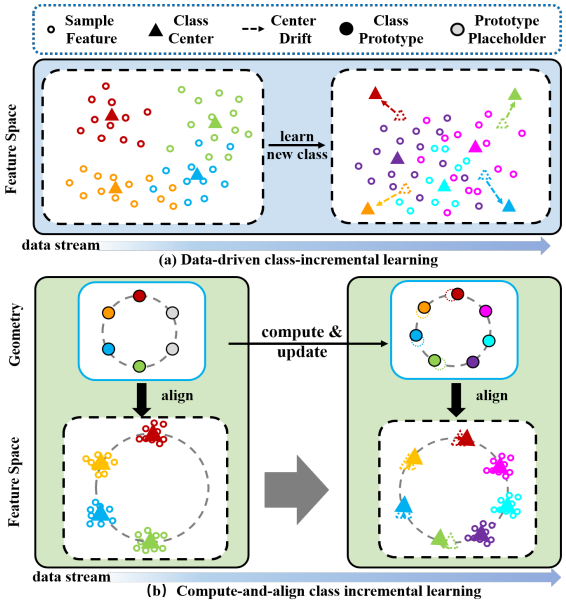

Figure 1. The illustration of (a) data-driven and (b) compute-and-align OTFCIL paradigm. In (a), samples of a new class are difficult to converge into a class center and old class centers drift significantly when learning new classes. In (b), the proposed paradigm first computes an optimal geometry for classifying existing classes and updates it when new classes emerge, and then learns the new classes by aligning the feature space to the geometry.

According to the knowledge learning and maintaining strategies, existing OTFCIL methods can be mainly divided into three sub-categories, *i.e.*, data replay [19, 34], network expansion [39, 41] and knowledge distillation [2, 33] methods. The data replay methods [7, 34, 37, 43, 51] store a

*Equal contribution
†Corresponding author

small number of training samples [7, 34, 37] or generated samples [43, 51] (namely exemplars) of the old classes in a memory buffer, and replay them when learning new class data to alleviate the catastrophic forgetting of old classes. The network expansion methods [32,34,39,41,47] initialize a new branch [32, 39, 47], a new prompt [34] or a new classifier [41] to learn the new class knowledge and remain the other parts frozen during incremental learning to maintain the old class knowledge. The knowledge distillation methods [17, 27, 30, 33, 37, 46] regard the previously obtained model as a teacher model, and transfer the old knowledge to the new model by logic- [27, 37], feature- [17, 33] and relation-distillation [30, 46].

All these methods, during the incremental learning process, have to continuously and dynamically tune their feature spaces to fit the seamless data stream. This paradigm is *ponderous* in learning new and *unstable* in maintaining old. As shown in Figure 1 (a), when learning new classes, the features of a new class are difficult to converge into a certain class center (*i.e.*, within-in class mean) as the center varies with newly obtained samples, and the old class centers inevitably drift along with the feature space tuning. This is especially true when samples are one-shot provided (*i.e.*, online learning) and the old and new class data are blended (*i.e.*, without task boundaries and identifiers).

Based on the above observation, we wonder, **instead of *learning* the feature space geometry from data, whether it is possible to *compute* an optimal classification geometry and dynamically *update* it when new classes emerged**. On this basis, the OTFCIL can be simply solved by aligning the feature space to the optimal geometry, *i.e.*, the compute-and-align paradigm shown in Figure 1 (b). To this end, we propose a novel DYnamic feature space Self-OrganizatoiN (DYSON) framework inspired by the Neural Collapse (NC) theory [53]. The NC theory reveals that, for a K-class classification problem, the last-layer features of the same class will collapse to K prototypes of a simplex equiangular tight frame (ETF) at the final training stage (training error equals 0). The prototypes construct an optimal geometry for classification, where all the prototypes are unit vectors having the same l_2 norm and share the same pair-wise angles that maximally separate the feature space. This opens an opportunity to compute the optimal classification geometry. However, as the vanilla NC requires the number of classes to be pre-defined and fixed during training, it is infeasible for class incremental learning. To meet this challenge, we propose a novel Dynamic Neural Collapse (DNC) algorithm to adaptively compute and update the optimal geometry along with the continuously emerging classes. The DNC expands the geometry when new classes emerge without loss of the geometry optimality and guarantees the drift distance old class prototypes are with an explicit upper bound.

On this basis, we propose the DYSON framework containing three major components, including 1) a feature extractor, 2) a Dynamic Feature-Geometry Alignment (DFGA) module aligning the feature space to the optimal geometry computed by the DNC algorithm, and 3) a training-free class incremental classifier derived from the DNC geometry. We conduct comprehensive experiment results on four benchmark datasets, including CIRAF-10, CIFAR100, CUB200 and Core50, and compare the DYSON with state-of-the-art methods. The comparison results demonstrate the superiority and efficiency of the DYSON. The main contributions can be summarized as:

- We propose a novel learn-and-align paradigm for OT-FCIL, which first computes an optimal classification geometry and updates it when new classes emerge, and then aligns the feature space to the geometry.
- We design a novel Dynamic Neural Collapse (DNC) algorithm, which extends the geometry without losing the geometry optimality and guarantees the old class prototype shifts have an explicit upper bound.
- We propose a novel Dynamic feature space Self-Organization (DYSON) containing a feature extractor, a Dynamic Feature-Geometry Alignment (DFGA) module and a training-free class-incremental classifier.
- The DYSON significantly outperforms state-of-the-art methods by a large margin (at most 8.9%, 16.6%, 26.8% and 4.4% on CIFAR-10, CIFAR-100, CoRe50 and CUB-200, respectively).

2. Related Work

2.1. Class Incremental Learning

The existing continual learning methods can be mainly divided into three categories: data replay, network expansion and knowledge distillation methods.

The data replay methods prevent models from catastrophic forgetting by storing or synthesizing old task sample data. The End-to-end [7], iCaRL [37] and its improved method [4, 11, 18] select the stored old samples based on herding algorithm. In addition, stored old task data can also be used as regularization constraints. A-GEM [8] uses the old and new data to calculate the scalar product of the loss gradient vector, and only updates the parameters when the scalar product is positive. Generating pseudo samples for replay can also effectively alleviate model forgetting. ILU-GAN [51] generates pseudo samples through a generative adversarial network [13] to solve the problem of data imbalance, as do the methods [36] and [44].

The network expansion methods mitigate catastrophic forgetting by freezing or isolating parts of the model's structure. [40] freeze the most important weights for a particu-

lar task so that they are not updated in back-propagation. PackNet [32] and its variants [1] allocate a portion of the parameter space for each task using different strategies to isolate new and old task parameters. On the other hand, some architecture-based methods dynamically add new layers to the network to increase the model's ability to learn new knowledge, such as [38, 48]. PNN [39], Dytox [12], Foster [47] and Der [52] methods replicate a new organization by task to achieve the transfer of old and new knowledge.

The knowledge distillation is introduced in the loss function so that the updated model retains the memory of the past. Weight distillation is a method based on regularization. By reducing the degree of change in the weight associated with the previous task, the weight retains the knowledge learned in the old task. EWC [21] proposes to use the fisher information matrix to calculate the importance of weights, and its variant methods [2, 29, 56] improve the calculation of importance. Another method of regularization is data regularization. Methods such as LWF [27] and [10, 25], based on the knowledge distillation of the new task data, make the prediction of the new task by the new model similar to the prediction of the new task by the old model.

In addition to the above three types, there are some new trends in CL field methods. L2P [49] uses a pre-trained model as a backbone network and recovers knowledge by learning prompts. FearNet [20] uses a dual memory system to prevent catastrophic forgetting. Mixed-system models are also becoming more common. A-GEM [8] combines playback and regularization methods. DSDM [34] uses dynamic memory while changing the model architecture, so it is a mixture of architecture-based and rehearsal-based.

2.2. Online Class-Incremental Learning

Online Class-Incremental Learning. Online class-incremental learning aims to train models on how to effectively learn knowledge from a single transmitted online data stream. Compared to offline learning, online learning will limit the learning efficiency of the model and exacerbate catastrophic forgetting. There are many algorithms proposed from different perspectives in this field, such as [3, 5, 14, 15, 28, 35, 42]. MIR [3] proposes to select samples according to gradient for training. DVC [14] proposes to use the mutual information of images to fully explore the semantic information in unidirectional data flow. GDUMB [35] proposes to use buffer data to retrain the model to address catastrophic forgetting. [28] couples agent-based losses with contraction-based losses.

Online Task-Free Class-Incremental Learning. Unlike online continuous learning, online task-free continuous learning has no task boundaries. To solve the OTFCL problem, some methods randomly initialize the model and train it from scratch. CoPE [9] proposes an architecture for balancing model stability and plasticity in OTFCL experiments. Considering gradients, GMED [19] proposes a strategy to select the sample data that best represents the old task. CN-DPM [26] designs a scalable model architecture that breaks through the limitations of OTFCL. However, the above models are trained by random initialization, which makes their classification accuracy relatively low. Therefore, DSDM [34] and Ensemble [41] proposed to use the pre-trained model as the backbone network to solve the OTFCL problem, and achieved excellent results. DSDM [34] updates the unit pool composed of position vectors and label vectors according to the distribution of features. Ensemble [41] is primarily trained in single-layer linearity so that it can correspond to the class label chosen by the NCM classifier in the inference phase.

The most related method to our DYSON is FCA [54], which employs the neural collapse technique to solve the few-shot class-incremental learning problem. Nevertheless, the differences and contributions of our DYSON *w.r.t* FAC are distinct and significant. First, the DYSON has the technical ability (using DNC) to incrementally update the geometry for newly emerged classes, while FCA has to pre-define the total class number and fix it during training. Second, the FCA relies on task boundaries and identifiers to distinguish old and new class knowledge and employs memory buffers storing old class samples to alleviate catastrophic forgetting, while DYSON is an online task-free method and does not require any memory buffers. In Section 5, we compare our DYSON to the FCA to demonstrate the efficiency and superiority of the proposed method.

3. Preliminary

The Neural Collapse (NC) [53] reveals a phenomenon that at the terminal phase of training (after 0 training error) of a classification task, the optimal geometric structure of the last-layer feature space can be defined by a simplex equiangular tight frame (ETF).

Definition of (Simplex Equiangular Tight Frame). For a classification problem of K classes, its within-class means of K classes correspond to a K-prototype simplex ETF, where the prototypes, *i.e.*, $\mathbf{m}^i \in \mathbb{R}^d,\ i = 1, ..., K$, can be obtained by:

$$\mathbf{M} = \sqrt{\frac{K}{K-1}}\mathbf{U}(\mathbf{I}_K - \frac{1}{K}\mathbf{1}_{K_t}\mathbf{1}_K^T), \tag{1}$$

where $\mathbf{M} = [\mathbf{m}^1, \ldots, \mathbf{m}^K] \in \mathbb{R}^{d\times K}$, $\mathbf{U} \in \mathbb{R}^{d\times K}$ is a randomly initialized orthogonal matrix satisfying $\mathbf{U}^T\mathbf{U} = \mathbf{I}_K$, $\mathbf{I}_{K_t}$ is an identity matrix of shape K and $\mathbf{1}_K$ is an all-one vector.

The prototypes in a simplex ETF construct an optimal geometric structure for classification, that all the prototypes are unit vectors having the same l_2 norm and maximally

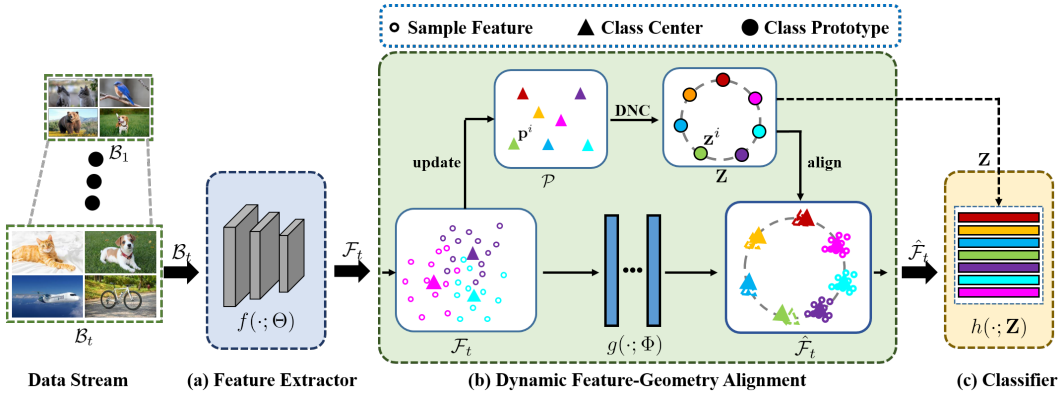

Figure 2. The illustration of the proposed DYSON framework, which contains (a) a feature extractor $f(\cdot;\Theta)$, (b) a Dynamic Feature-Geometry Alignment module containing a projection layer $g(\cdot;\Phi)$ and (c) a training-free class-incremental classifier.

separates the feature space with the same pair-wise angles, *i.e.*,

$$\mathbf{m}_i^T \mathbf{m}_j = \frac{K}{K-1}\delta_{i,j} - \frac{1}{K-1}, \forall i,j \in [1,...,K], \quad (2)$$

where $\delta_{i,j} = 1$ when $i = j$, and 0 for otherwise. The pair-wise angle $-\frac{1}{K-1}$ is the maximal equiangular separation of K vectors in the d-dimension feature space.

Based on the above definition, the NC phenomenon can be summarized as:

(NC1) Within-class last-layer feature collapse: the last-layer features of the same class will collapse to its within-class mean, *i.e.*, $\Sigma_W \to 0$ where $\Sigma_W = \text{Avg}_{i,k}\{(\mathbf{h}_{i,k} - \mathbf{h}_k)(\mathbf{h}_{i,k} - \mathbf{h}_k)^T\}$. The $\mathbf{h}_{i,k}$ is last-layer feature of the i-th sample of the k-th class and $\mathbf{h}_k = \text{Avg}_i(\mathbf{h}_{i,k})$ is the within-class mean of the k-th class.

(NC2) Convergence to a simplex ETF: the within-class means centered by the global mean, *i.e.*, $\tilde{\mathbf{h}}_k = (\mathbf{h}_k - \mathbf{h}_G)/||\mathbf{h}_k - \mathbf{h}_G||$ will converge to the prototypes of a simplex ETF defined in Eq (1), where $\mathbf{h}_G = \text{Avg}_{i,k}(\mathbf{h}_{i,k})$ is the last-layer global mean.

(NC3) Self-duality: the classifier weight, *i.e.*, $\mathbf{w}_k$, of the k-th class is parallel (aligned) to its global centered within-class mean, *i.e.*, $\tilde{\mathbf{h}}_k = \mathbf{w}_k/||\mathbf{w}_k||$.

(NC4) Simplification to the nearest class center prediction: $\text{argmax}_k\langle \mathbf{h}, \mathbf{w}_k\rangle = \text{argmin}_k||\mathbf{h} - \mathbf{w}_k||$, where $\mathbf{h}$ is a last-layer feature of an input sample.

4. Methodology

4.1. Problem Formulation

Let $\mathcal{D} = \{\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_T\}$ be an input data stream of length T ($T \to \infty$ for infinite data stream), where each element $\mathcal{B}_t = \{(\mathbf{X}_t^i, y_t^i)\}_{i=1}^N$ of $\mathcal{D}$ denotes the t-th sample batch and N is the batch size. The $\mathbf{X}_t^i \in \mathbb{R}^{W\times H}$ denotes the i-th image in $\mathcal{B}_t$, and $y_t^i \in \mathcal{C}_t$ denotes its corresponding class label, where $\mathcal{C}_t$ is the class set of $\mathcal{B}_t$. Following the OTFCL setting [34], each sample is one-shot provided, *i.e.*, $\forall i \neq j, \mathcal{B}_i \bigcap \mathcal{B}_j = \emptyset$, and there are no task boundaries between different batches. The objective of OTFCL is to train a unified model that can incrementally learn and recognize the continuously emerging new classes in $\mathcal{D}$ by online training (*i.e.*, training epoch equals 1). At each training step t, we only have access to the current data batch $\mathcal{B}_t$, and the previous batches $\mathcal{B}_1, \cdots, \mathcal{B}_{t-1}$ are not available. For evaluation, the model is expected to recognize all the encountered classes $\mathcal{C}_{1\sim t} = \mathcal{C}_1 \bigcup \cdots \bigcup \mathcal{C}_t$.

4.2. Framework Overview

Figure 2 depicts the framework of our proposed DYSON method, which is an online and parameter-efficient learning model containing three major components: (a) a pre-trained feature extractor backbone $f(\cdot;\Theta)$, (b) a Dynamic Feature-Geometry Alignment (DFGA) module containing a feature projection layer $g(\cdot;\Phi)$ and (c) a training-free class-incremental classifier $h(\cdot;\mathbf{Z})$, where Θ, Φ and $\mathbf{Z}$ are the parameter sets of the three components, respectively. During incremental learning, the parameters of the feature extractor $f(\cdot;\Theta)$ and the classifier $h(\cdot;\mathbf{Z})$ are frozen and training-free, and only the projection layer $g(\cdot;\Phi)$ is optimized to align the feature space to the optimal geometry computed by the DNC algorithm.

Specifically, at the beginning, we initialize an empty set $\mathcal{P} = \{\}$ to store the class centers. **Feature extractor**. At each step t, taking the sample batch $\mathcal{B}_t$ as input, the Module (a) outputs a feature collection $\mathcal{F}_t$ of $\mathcal{B}_t$:

$$\mathbf{f}_t^i = f(\mathbf{X}_t^i;\Theta), \quad (3)$$

$$\mathcal{F}_t = \{\mathbf{f}_t^i | i = 1,\ldots,|\mathcal{B}_t|\}, \quad (4)$$

where $\mathbf{f}_t^i \in \mathbb{R}^{d\times 1}$ is the feature of $\mathbf{X}_t^i$ and d is the feature dimension.

Dynamic Feature-Geometry Alignment. Taking the feature set $\mathcal{F}_t$ as input, we fist compute a within-class mean $\mathbf{p}_t^j \in \mathbb{R}^d$ for each class j in $\mathcal{C}_t$, where $\mathbf{p}_t^j = avg(\{\mathbf{f}_t^i | \forall i, y_t^i = j\})$ computes the average feature of the

j-th class features in $\mathcal{F}_t$. For each class $j \in \mathcal{C}_t$, if j is a new class, *i.e.*, $j \notin \mathcal{C}_{1\sim t-1}$, we initialize $\mathbf{p}_t^j$ as the class center $\mathbf{p}^j$ of the class j; If j is an old class, *i.e.*, $j \in \mathcal{C}_{1\sim t-1}$, we update the class center $\mathbf{p}^j$ of class j by moving average using $\mathbf{p}_t^j$:

$$\forall j \in \mathcal{C}_t,\ \mathbf{p}^j = \begin{cases} \mathbf{p}_t^j, & \text{if } j \notin \mathcal{C}_{1\sim t-1}, \\ \beta \cdot \mathbf{p}^j + (1-\beta)\cdot \mathbf{p}_t^j, & \text{if } j \in \mathcal{C}_{1\sim t-1}, \end{cases} \tag{5}$$

where β is an updating rate ratio. The output set $\mathcal{P}$ collects the class centers of all the encountered classes, *i.e.*, $\mathcal{P} = \{\mathbf{p}^1, ..., \mathbf{p}^{K_t}\}$, where $K_t = |\mathcal{C}_{1\sim t}|$ denotes the total number of currently encountered classes.

On this basis, we compute the class prototype $\mathbf{Z}$ for the current K_t classes using the Dynamic Neural Collapse (DNC) algorithm:

$$\mathbf{Z} = [\mathbf{z}^1, ..., \mathbf{z}^{K_t}] = \varphi_{dnc}(\mathcal{P}), \tag{6}$$

where $\mathbf{Z} \in \mathbb{R}^{d\times K_t}$ outputs the concatenated class prototypes of K_t classes and each column vector $\mathbf{z}^j \in \mathbb{R}^{d\times 1}$ in $\mathbf{Z}$ corresponds the j-th class prototype. It is worth mentioning that, the class prototypes in $\mathbf{Z}$ are updated using Eq. (6) only when new classes emerge. In Section 4.4, we prove that the prototypes in $\mathbf{Z}$ construct an optimal classification geometry satisfying Eq. (2) and old class prototype drift distance after the geometry updating are with an upper bound.

During training, at each learning step t, we generate a set of pseudo features $\mathcal{V}_t = \{\mathbf{v}_t^1, ..., \mathbf{v}_t^N\}$ for the old classes, *i.e.*, $\forall j \in \mathcal{C}_{1\sim t-1}$, by augmenting their class centers $\mathbf{p}^j$ using Gaussian noise [57], where $\mathbf{v}_t^i$ denotes the i-th pseudo feature and $|\mathcal{V}_t| = |\mathcal{B}_t| = N$. By projecting the sample features $\mathbf{f}_t^i$ and pseudo features $\mathbf{v}_t^i$ to their corresponding class prototypes using the projection layer $g(\cdot;\Phi)$, we align the feature space to the optimal geometry constructed by $\mathbf{Z}$. The feature-geometry alignment objective can be written as:

$$\begin{aligned} \mathcal{L}_{AL} = & \frac{1}{|\mathcal{B}_t|}\sum_{i=1}^{|\mathcal{B}_t|} ||g(\mathbf{f}_t^i;\Phi) - \sigma(\mathbf{Z}, \mathbf{f}_t^i)||_2 + \\ & \frac{1}{|\mathcal{V}_t|}\sum_{i=1}^{|\mathcal{V}_t|} ||g(\mathbf{v}_t^i;\Phi) - \sigma(\mathbf{Z}, \mathbf{v}_t^i)||_2, \end{aligned} \tag{7}$$

where $\sigma(\mathbf{Z}, \mathbf{f}_t^i)$ and $\sigma(\mathbf{Z}, \mathbf{v}_t^i)$ output the corresponding class prototypes $\mathbf{z}^j \in \mathbf{Z}$ of the input features $\mathbf{f}_t^i$ and $\mathbf{v}_t^i$, respectively, according to their class labels. We use $\hat{\mathbf{f}}_t^i = g(\mathbf{f}_t^i;\Phi)$ to denote the projected feature of $\mathbf{f}_t^i$. The Module (b) outputs a collection of projected features $\hat{\mathcal{F}}_t = \{\hat{\mathbf{f}}_t^1, ..., \hat{\mathbf{f}}_t^N\}$.

Training-free class-incremental classifier. As the projection features $\mathbf{v}_t^i$ of the same class collapse to a class prototype $\mathbf{z}^j$ and the class prototypes are unit vectors sharing the same pair-wise angles, we regard each prototype $\mathbf{z}^j$ as the classification weight for the j-th class and conduct a linear classifier with weight $\mathbf{Z} = [\mathbf{z}^1, ..., \mathbf{z}^{K_t}]$. Given the projection feature $\hat{\mathbf{f}}_t^i$ of a sample $\mathbf{X}_t^i$, the classification can be solved by:

$$h(\hat{\mathbf{f}}_t^i; \mathbf{Z}) = \arg\max_{\mathbf{z}^j}\ \hat{\mathbf{f}}_t^{i^T}[\mathbf{z}^1, ..., \mathbf{z}^{K_t}]. \tag{8}$$

When new classes emerge, we extend the classifier by updating $\mathbf{Z}$ computed by the DNC.

Loss function. During training, taking each sample batch $\mathcal{B}_t$ as input, we optimize the projection layer $h(\cdot;\Phi)$ using the feature-geometry alignment loss $\mathcal{L}_{AL}$ in Eq. (7) and the cross-entropy loss [24]:

$$\min_{\Phi} \mathcal{L}_{total}(\mathcal{B}_t; \Theta, \Phi, \mathbf{Z}) = \mathcal{L}_{AL} + \mathcal{L}_{CE}. \tag{9}$$

4.3. Dynamic Neural Collapse

In the following, we omit the subscript t for description simplicity. Taking the class center set $\mathcal{P}$ as input, we concatenate the class centers into a matrix $\mathbf{P} = [\mathbf{p}^1, ..., \mathbf{p}^K] \in \mathbb{R}^{d\times K}$, where $\mathbf{p}^j \in \mathbb{R}^{d\times 1}$ is the j-th class center and K is the class number. We first compute the QR decomposition of the matrix $\mathbf{P}$:

$$\mathbf{P} = \mathbf{Q}\mathbf{R}, \tag{10}$$

where $\mathbf{Q} = [\mathbf{q}^1, ..., \mathbf{q}^K] \in \mathbb{R}^{d\times K}$ is an orthogonal matrix with $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_K$, and $\mathbf{R} \in \mathbb{R}^{K\times K}$ is an upper triangular matrix. Then we compute the class prototype matrix $\mathbf{Z}$ using $\mathbf{Q}$:

$$\mathbf{Z} = \sqrt{\frac{K}{K-1}}\mathbf{Q}(\mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^T), \tag{11}$$

where $\mathbf{Z} = [\mathbf{z}^1, ..., \mathbf{z}^K] \in \mathbb{R}^{d\times K}$ is the concatenated matrix of K class prototypes and $\mathbf{z}^j \in \mathbb{R}^{d\times 1}$ is the prototype of class j. The $\mathbf{I}_K$ is an identity matrix of shape K and $\mathbf{1}_K$ is an all-one vector. The prototypes in $\mathbf{Z}$ construct an optimal geometry for K-class classification, that for any $\mathbf{z}^i$ and $\mathbf{z}^j$ in $\mathbf{Z}$, we have:

$$\forall i, j,\ \mathbf{z}^{i^T}\mathbf{z}^j = \frac{K}{K-1}\cdot \mathbf{q}^{i^T}\mathbf{q}^j - \frac{1}{K-1}, \tag{12}$$

where all the prototypes are unit vectors, *i.e.*, $\forall i,\ ||\mathbf{z}^i||^2 = \mathbf{z}^{i^T}\mathbf{z}^i = 1$, and any pair of the prototypes have the same angle, *i.e.*, $\forall i \neq j, \mathbf{z}^{i^T}\mathbf{z}^j = -\frac{1}{K-1}$.

Prototype update. When C new classes emerge, we have $\mathcal{P} = \{\mathbf{p}^1, ..., \mathbf{p}^K; \mathbf{p}^{K+1}, ..., \mathbf{p}^{K+C}\}$. We concatenate $\mathcal{P}$ into $\mathbf{P}' = [\mathbf{p}^1 \cdots \mathbf{p}^{K+C}]$ and compute the updated class prototypes $\mathbf{Z}' = [\mathbf{z}^{1'}, ..., \mathbf{z}^{K'}; \mathbf{z}^{K+1'}, ..., \mathbf{z}^{K+C'}]$ using Eqs. (10) and (11). The drift distances of the old class prototypes, *i.e.*, $\mathbf{z}^i,\ i = 1, ..., K$, can be computed by:

$$||\mathbf{z}^i - \mathbf{z}^{i'}||^2 = 2 - 2\sqrt{\frac{(K-1)(K+C)}{K(K+C-1)}}. \tag{13}$$

In Section 4.4, we prove that the DNC extends the geometry from K to $K+C$ prototypes without loss of the geometry optimality, and the old class drift distances, *i.e.*, $||\mathbf{z}^i - \mathbf{z}^{i'}||, \forall i \leq K$ are with an upper bound.

Improve geometry stability. At the early learning stage when class number K is small, the class prototypes in $\mathbf{Z}$ are sparse-distributed and the shift distances of old class prototypes are large. This results in the geometry being unstable in maintaining old knowledge. To improve the geometry stability during the incremental learning process, we introduce prototype placeholder $\mathbf{E} \in \mathbb{R}^{d\times M}$ to the DNC, where M is the number of placeholders.

Specifically, we initialize a random orthogonal matrix $\mathbf{E} = [\mathbf{e}^1, ..., \mathbf{e}^M] \in \mathbb{R}^{d\times M}$ with M prototype placeholders, where $\forall i \neq j, \mathbf{e}^{i^T}\mathbf{e}^j = 0$. Taking the class center matrix $\mathbf{P} \in \mathbb{R}^{d\times K}$ as input, when $K < M$, we first pad the $\mathbf{P}$ using the placeholders in $\mathbf{E}$:

$$\mathbf{P}_{pad} = [\mathbf{p}^1, ..., \mathbf{p}^K; \mathbf{e}^{K+1}, ..., \mathbf{e}^M], \tag{14}$$

and then compute the class prototype $\mathbf{Z} = [\mathbf{z}^1, ..., \mathbf{z}^K; \mathbf{z}^{K+1}, ..., \mathbf{z}^M]$ using the Eqs.(10) and (11) with the padded $\mathbf{P}_{pad}$. We then select the former K columns in $\mathbf{Z}$ as the prototypes of K classes. When $K \geq M$, we directly compute $\mathbf{Z}$ using $\mathbf{P}$. On this basis, the shift distance of the old class prototypes are with an upper bound:

$$||\mathbf{z}^i - \mathbf{z}^{i'}||^2 \leq 2 - 2\sqrt{\frac{(M-1)(M+C)}{M(M+C-1)}}, \tag{15}$$

where C is the number of newly emerged classes. With Eq. (15), the stability of the DNC geometry is guaranteed.

4.4. Theoretical Analysis

Theorem 1. *During incremental learning, the DNC algorithm extends a simplex ETF with K prototypes $\mathbf{Z}$ to one with $K+C$ prototypes $\mathbf{Z}'$ without loss of the geometry optimality,* i.e.*, the prototypes in $\mathbf{Z}'$ satisfy:*

$$\forall i, j,\ \mathbf{z}^{i'^T}\mathbf{z}^{j'} = \frac{K+C}{K+C-1}\delta_{i,j} - \frac{1}{K+C-1}, \tag{16}$$

where $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ for the opposite.

The prototypes in $\mathbf{Z}'$ construct an optimal geometry for $(K+C)$-class classification, where all the vectors are unit vectors, *i.e.*, $\forall i, ||\mathbf{z}^{i'}||^2 = \mathbf{z}^{i'^T}\mathbf{z}^{i'} = 1$, and have the same pair-wise angle maximally separating the feature space with $K+C$ vectors, *i.e.*, $\forall i \neq j, \mathbf{z}^{i'^T}\mathbf{z}^{j'} = -\frac{1}{K+C-1}$. This theorem allows us to adaptively update the optimal classification geometry using the DNC algorithm when new classes emerge and incrementally train a unified by aligning the feature space to the optimal geometry. The proof of Theorem 1 is provided in the Appendix.A.

Theorem 2. *With M prototype placeholders, extending existing K-class prototypes to $K+C$ ones using the DNC algorithm when C new class emerged, the upper bound of the old class prototypes is guaranteed, i.e., $\forall i = 1, ..., K, ||\mathbf{z}^i - \mathbf{z}^{i'}||^2 \leq 2 - 2\sqrt{\frac{(M-1)(M+C)}{M(M+C-1)}}$, where $\mathbf{z}^i$ and $\mathbf{z}^{i'}$ are the old and new prototypes of class i, respectively.*

This theorem guarantees the old class geometry stability when learning new classes. The proof of Theorem 2 is provided in Appendix.B.

5. Experiments

In this section, we conduct comprehensive experiments on four benchmark datasets to demonstrate the superiority and generality of the proposed method.

5.1. Dataset and Evaluation Details

We conduct experimental results on four benchmark datasets, including CIFAR-10 [23], CIFAR-100 [23], CUB-200 [50] and CoRe50 [31], to validate the efficiency and effectiveness of the proposed DYSON method.

Datasets. CIFAR-10 [23] consists of 60,000 images in 10 classes, where 50,000 are used for training and the rest are used for testing. The image resolution is 32×32. CIFAR-100 [23] contains 60,000 images for 100 classes, where each class contains 600 images with 500 training and 100 testing images, respectively. CUB-200 [50] consists of 11,788 images of 200 bird categories, where each class contains about 30 and 29 images for training and testing, respectively. CoRe50 [31] contains 50 classes with a total number of 16,4866 images. Each class contains around 2400 and 900 training and testing images, respectively.

Evaluation Protocol. For fair comparisons, following the evaluation protocol in [34, 41], we conduct the *n-step* comparison results on the four benchmark datasets. The n-step protocol denotes that, with random class order, each incremental learning stage contains of n classes. For example, for a 2-step incremental learning on 10 classes, the OTFCIL contains 5 incremental learning stages with each stage containing 2 classes. We conduct 1-step, 2-step, 5-step and the Gaussian-step to evaluate our method.

Evaluation metrics. We adopt the widely used average accuracy (Avg) and last step accuracy (Last) to evaluate the proposed method. The Avg is the average of accuracy after each learning stage and the Last is the final accuracy on all classes. We conduct the experiments on a PC with an NVIDIA 3090Ti GPU, and run 5 times for each experiment and report the average performance.

Implementation details. Similar to the state-of-the-art OTFCIL methods [34], CoPE [9] and Ensemble [41], we employ a ResNet50 [16] and a ViT-S/8 [6] pre-trained on ImageNet as our feature extractor backbone, where the classifier is removed. We set the batch size $N = 50$ and the

Method	Backbone	Buffer Size	CIFAR-10			Buffer Size	CIFAR-100		
			1-step	2-step	Gaussian		1-step	5-step	Gaussian
ODDL [55]	Resnet18	2k	-	52.7	-	2k	-	27.2	-
SDP [22]		500	-	66.2	76.3	2k	-	-	-
Ours		0	**73.4**	**74.4**	**76.5**	0	**49.6**	**45.3**	**47.0**
CoPE [9]	Resnet50	1k	-	48.9	-	5k	-	21.6	-
CN-DPM [26]		1k	-	45.2	-	1k	-	20.1	-
GMED [19]		5k	-	47.7	-	5k	-	19.6	-
FCA [54]		0	77.5	78.7	76.1	0	53.3	48.7	53.4
Ensemble [41]		2k	78.3	79.0	50.1	6k	54.1	55.3	39.0
DSDM [34]		2k	79.4	79.6	78.7	6k	54.9	55.3	55.5
Ours		0	**79.5**	**80.7**	**79.1**	0	**58.9**	**59.2**	**58.6**
L2P [49]	ViT-S/8	1k	46.8	61.4	57.5	3k	8.4	27.3	47.8
DSDM [34]		1k	85.5	85.6	84.9	3k	61.1	60.8	61.4
Ours		0	**92.6**	**93.5**	**93.8**	0	**77.7**	**75.6**	**76.4**

Table 1. The OTFCIL performance in terms of Last accuracy (%) on the CIFAR-10 and CIFAR-100 datasets. The best and second results are noted by **bold** and underline, respectively.

Datasets	CIFAR-10						CoRe50					
Buffer size	1k		2k		5k		1k		2k		5k	
Accuracy	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last
A-GEM [8]	43.0	17.5	59.1	38.3	74.0	59.0	20.7	8.4	21.9	10.3	22.9	11.5
MIR [3]	67.3	52.2	80.2	66.2	83.4	74.8	33.9	21.1	37.1	24.5	38.1	27.7
GSS [5]	70.3	56.7	73.6	56.3	79.3	64.4	27.8	17.8	31.0	18.9	31.8	21.1
ASER [42]	63.4	46.4	78.2	59.3	83.3	73.1	24.3	12.2	30.8	17.4	32.5	18.5
GDUMB [35]	73.8	57.7	83.8	72.4	85.3	75.9	41.2	23.6	48.4	32.7	54.3	41.6
CV [15]	76.0	62.9	**84.9**	**74.1**	**86.1**	**77.0**	**45.1**	26.5	50.7	34.5	56.3	43.1
DSDM [34]	**80.2**	**67.0**	83.8	72.5	85.6	76.0	43.9	**43.3**	**53.2**	**50.8**	**66.3**	**57.1**
Buffer size	0						0					
Accuracy	Avg			Last			Avg			Last		
Ours	**81.8**			**74.4**			**70.7**			**61.3**		

Table 2. Online CIL performance comparison on the CIFAR-10 and CoRe50 datasets.

number of prototype placeholders $M = 10$. We train our model using Adam with lr = 2e-5 and weight decay = 5e-6. We set $\beta = 0.3$ for center updating in Eq. (5).

5.2. Comparison with state-of-the-art methods

Online Task-Free Class-Incremental Learning. In Table 1, we compare our DYSON method with state-of-the-art OTFCIL methods, including CoPE [9], CN-DPM [26], GMED [19], FCA [54], Ensemble [41] and DSDM [34], and a represent class incremental learning L2P [49] method. Similar to the very recent work DSDM, we report the Last accuracy on the CIFAR-10 and CIFAR-100 datasets under the 1-, 2-, 5- and Gaussian-step evaluation protocols, where the compared using the identical ResNet50 and ViT-S/8 as the one of ours. From the table, we can see that, 1) by computing the DNC geometry and aligning the feature space to the geometry using new class features and old class pseudo features, the DYSON is a memory-buffer-free method that does not require storing any old class samples to alleviate the catastrophic forgetting. 2) Compared to the competitive DSDM method using a memory buffer storing 1k old class samples, the proposed DYSON method significantly and steadily outperforms the DSDM both on the CIFAR-10 and CIFAR-100 datasets by a large margin. On the CIFAR-10 dataset, it outperforms the DSDM by 7.1%, 7.9% and 8.9% in terms of the 1-, 2- and Gaussian-step protocols, respectively, using the same ViT-S/8 backbone. On the more challenging CIFAR-100 dataset, it dramatically outperforms the DSDM by 16.1%, 14.8% and 15.0% in terms of the 1-, 2- and Gaussian-step protocols, respectively, using the identical ViT-S/8 backbone. These results demonstrate the superiority of the DYSON method. 3) Using both the ResNet50 and ViT-S/8 as the backbone, the proposed DYSON steadily and dramatically outperforms the state-of-the-art methods, demonstrating the robustness of the proposed method to different backbones.

Online Class-Incremental Learning. Following the online CIL setting in [15,34], in Table 2, we conduct experimental comparisons on the CIFAR-10 and CoRe50 datasets. The CIFAR-10 is evaluated under the 2-step evaluation protocol, where the learning progress is split into 5 learning stages, where each stage contains the samples of two novel classes. The Core50 is split into 9 learning stages, where 10 classes are learned in the first stage and 5 classes for the following incremental stages. We compare our proposed method with SOTA online CIL methods, including A-GEM [8], MIR [3], GSS [5], ASER [42], GDUMB [35] and Candidates Voting [15]. For a fair comparison, following the settings in DSDM, all the compared methods (including DYSON) use an identical ImageNet pre-trained Resnet18 as the backbone. From Table 2, we can see that, 1) all the existing SOTA methods heavily rely on storing a

Method	CIFAR-100		CUB-200		Buffer Size
	2-step	5-step	2-step	5-step	
EWC [21]	15.4	17.9	15.9	18.2	
LwF [27]	32.8	37.9	29.8	36.1	
iCaRL [37]	52.8	57.2	44.3	49.6	
End-to-End [7]	50.3	49.7	44.5	49.9	3k
FearNet [20]	56.9	62.5	47.8	52.7	
ILUGAN [51]	58.0	63.1	49.7	54.9	
DSDM [34]	63.3	63.2	55.2	55.5	
Ours	**71.7**	**71.9**	**59.1**	**59.9**	0

Table 3. Offline TFCL performance comparison in terms of Avg accuracy (%) on the CIFAR-100 and CUB-200 datasets. The best and second results are noted by **bold** and underline.

M	0	10	20	50	100	200
Last acc	56.80	**59.18**	58.24	59.01	58.65	57.54
Avg acc	67.67	**70.54**	69.41	69.62	69.74	68.32

Table 4. The influence of hyper-parameter M on CIFAR-100.

number of old class samples (*i.e.*, the buffer size) to solve the challenging online incremental learning problem, where the more samples they stored, the more accurate they are. 2) The proposed DYSON does not require storing the old class samples and significantly outperforms the memory-buffer-based methods by a large margin. Specifically, on the CoRe50 dataset, the DYSON dramatically outperforms the competitive DSDM-1k by 26.8% and 18.0% on Avg and Last accuracies, respectively, and outperforms the DSDM-5k by 4.4% and 4.2% on Avg and Last accuracies.

Offline task-free class-incremental learning. Further, in Table 3, we also conduct experiments on the CIFAR-100 and CUB-200 datasets for offline TFCIL performance evaluation. Following the settings in [20, 51], all the compared methods use an ImageNet pre-trained ResNet50 as the backbone, and the existing methods use 3k buffer size storing old class samples. From the table, we can see that the proposed DYSON steadily and significantly outperforms the existing methods by a large margin. Compared to the competitive DSDM method, DYSON achieves 8.4% and 8.7% Avg accuracy improvements on 2- and 5-step protocols on CIFAR-100 dataset, and achieves 3.2% and 4.4% improvements on the CUB-200 dataset.

Performance analyze. We can see the DYSON significantly outperforms the existing SOTA methods without using any memory buffer. The performance improvements come from two aspects. First, the learn-and-align paradigm, which first computes an optimal geometry for the emerged classes and then aligns the feature space to the geometry. Second, the proposed DNC algorithm computes and updates the geometry when new classes emerge. It extends the geometry without loss of structure optimality and guarantees the old class prototypes have an explicit upper bound, guarantees the stability of old knowledge.

Dataset	Backbone	1-step	2-step	Gaussian
CIFAR-10	ResNet50	79.5	80.7	79.1
	DINO-ResNet50	**84.5**	**85.6**	**83.8**
CIFAR-100	ResNet50	58.9	59.2	58.6
	DINO-ResNet50	**60.9**	**57.1**	**60.1**

Table 5. Ablation study using self-supervised pre-trained backbone on the CIFAR-10 and CIFAR-100 datasets.

5.3. Ablation Studies

Influence of hyper-parameter M. In Table 4, we conduct an ablation study on the CIFAR-100 dataset under the 5-step evaluation protocol to analyze the influence of parameter M on Last accuracy. The M is the number of placeholders influencing the stability of the DNC geometry. From the table, we can see that, when $M = 0$, the performance is inferior both in Avg and Last. This is due to the geometry being unstable without the placeholders at the early CIL stage. When $10 \leq M \leq 100$, the performance keeps steady with the growth of M. When $M = 200$, the performance decreases slightly as too many placeholders hamper the learning of new classes. We fix $M = 10$ for all the experiments according to the experiment results.

Pre-trained backbone without using label annotation. To get rid of the concerns that the new classes in the four benchmark datasets have already been learned by the ImageNet pre-trained model, in Table 5, we conduct an ablation study using a self-supervised DINO-ResNet50 backbone [6]. The DINO-ResNet50 is pre-trained by self-supervised information distillation, wherein no class information is used in training. From the table, we can see that, using the DINO-ResNet50 further improves the performance, which demonstrates the ability of the proposed method in learning new classes.

6. Conclusion

We propose a novel DYSON method containing a feature extractor backbone, a Dynamic Feature-Geometry Alignment (DFGA) module and a training-free class-incremental classifier, for the challenging OTFCIL problem. It follows a novel learn-and-align paradigm, which first computes an optimal geometry classifying existing classes and updates it when new classes emerge, and then aligns the feature space to the geometry. We derive a Dynamic Neural Collapse (DNC) algorithm to compute and update the geometry, where the geometry is updated without loss of the optimality and old class prototype drift have an upper bound.

7. Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No.U21B2048 and No.6230070870, and CAAI-MindSpore Open Fund, developed on OpenI Community.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020. 3

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018. 1, 3

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019. 3, 7

[4] Arjun Ashok, KJ Joseph, and Vineeth N Balasubramanian. Class-incremental learning with cross-space clustering and controlled transfer. In *European Conference on Computer Vision*, pages 105–122. Springer, 2022. 2

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 3, 7

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 6, 8

[7] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018. 1, 2, 8

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 2, 3, 7

[9] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 3, 6, 7

[10] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019. 3

[11] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 86–102. Springer, 2020. 2

[12] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022. 1, 3

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[14] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng. Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7442–7451, 2022. 3

[15] Jiangpeng He and Fengqing Zhu. Online continual learning via candidates voting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3154–3163, January 2022. 3, 7

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[17] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[18] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019. 2

[19] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 34:29193–29205, 2021. 1, 3, 7

[20] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017. 3, 8

[21] 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. 3, 8

[22] Hyunseo Koh, Minhyuk Seo, Jihwan Bang, Hwanjun Song, Deokki Hong, Seulki Park, Jung-Woo Ha, and Jonghyun Choi. Online boundary-free continual learning by scheduled data prior. In *The Eleventh International Conference on Learning Representations*, 2022. 7

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

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 5

[25] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 312–321, 2019. 3

[26] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*, 2020. 3, 7

[27] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 1, 2, 3, 8

[28] Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24246–24255, 2023. 3

[29] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE, 2018. 3

[30] Yu Liu, Xiaopeng Hong, Xiaoyu Tao, Songlin Dong, Jingang Shi, and Yihong Gong. Model behavior preserving for class-incremental learning. *IEEE Transactions on Neural Networks and Learning Systems*, PP. 2

[31] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pages 17–26. PMLR, 2017. 6

[32] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 1, 2, 3

[33] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *International Conference on Computer Vision*, 2021. 1, 2

[34] Julien Pourcel, Ngoc-Son Vu, and Robert M French. Online task-free continual learning with dynamic sparse distributed memory. In *European Conference on Computer Vision*, pages 739–756. Springer, 2022. 1, 2, 3, 4, 6, 7, 8

[35] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 524–540. Springer, 2020. 3, 7

[36] Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *Neurocomputing*, 404:381–400, 2020. 2

[37] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1, 2, 8

[38] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. Tree-cnn: A hierarchical deep convolutional neural network for incremental learning, 2019. 3

[39] 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. 1, 2, 3

[40] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018. 2

[41] Murray Shanahan, Christos Kaplanis, and Jovana Mitrović. Encoders and ensembles for task-free continual learning. *arXiv preprint arXiv:2105.13327*, 2021. 1, 2, 3, 6, 7

[42] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638, 2021. 3, 7

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

[44] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017. 2

[45] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 254–270. Springer, 2020. 1

[46] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, and Yihong Gong. Few-shot class-incremental learning. *arXiv*, 2020. 2

[47] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pages 398–414. Springer, 2022. 2, 3

[48] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2471–2480, 2017. 3

[49] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 1, 3, 7

[50] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010. 6

[51] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6619–6628, 2019. 1, 2, 8

[52] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021. 3

[53] Yibo Yang, Liang Xie, Shixiang Chen, Xiangtai Li, Zhouchen Lin, and Dacheng Tao. Do we really need a learnable classifier at the end of deep neural network? *arXiv e-prints*, pages arXiv–2203, 2022. 2, 3

[54] Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. Neural collapse inspired feature-classifier alignment for few-shot class incremental learning. *arXiv preprint arXiv:2302.03004*, 2023. 3, 7

[55] Fei Ye and Adrian G Bors. Task-free continual learning via online discrepancy distance learning. *Advances in Neural Information Processing Systems*, 35:23675–23688, 2022. 7

[56] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017. 3

[57] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021. 5