Topology-Preserving Class-Incremental Learning

Xiaoyu Tao[1,#], Xinyuan Chang[2,#], Xiaopeng Hong[1,3], Xing Wei[2], and Yihong Gong[2⋆]

[1] Faculty of Electronic and Information Engineering, Xi'an Jiaotong University
[2] School of Software Engineering, Xi'an Jiaotong University
[3] Research Center for Artificial Intelligence, Peng Cheng Laboratory
{txy666793,cxy19960919}@stu.xjtu.edu.cn,hongxiaopeng@mail.xjtu.edu.cn
xingxjtu@gmail.com,ygong@mail.xjtu.edu.cn

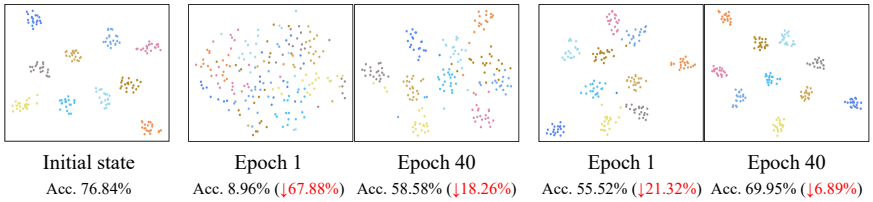
Abstract. A well-known issue for class-incremental learning is the *catastrophic forgetting* phenomenon, where the network's recognition performance on old classes degrades severely when incrementally learning new classes. To alleviate forgetting, we put forward to preserve the old class knowledge by maintaining the topology of the network's feature space. On this basis, we propose a novel *topology-preserving class-incremental learning* (TPCIL) framework. TPCIL uses an *elastic Hebbian graph* (EHG) to model the feature space topology, which is constructed with the *competitive Hebbian learning* rule. To maintain the topology, we develop the *topology-preserving loss* (TPL) that penalizes the changes of EHG's neighboring relationships during incremental learning phases. Comprehensive experiments on CIFAR100, ImageNet, and subImageNet datasets demonstrate the power of the TPCIL for continuously learning new classes with less forgetting. The code will be released.

Keywords: Topology-Preserving Class-Incremental Learning (TPCIL), Class-Incremental Learning (CIL), Elastic Hebbian Graph (EHG), Topology-Preserving Loss (TPL)

1 Introduction

To date, deep neural networks have been successfully applied to a large number of computer vision and pattern recognition tasks [16, 11, 34, 33, 22, 5, 20, 8, 41, 40, 24], etc. When applying a network to a classification problem, we generally first assume that the data classes are pre-defined and fixed, and then construct a network with the number of neural units in the output layer equal to the number of classes. In real applications, however, there often emerge new classes of data that have not been encountered before, and can not be recognized by the learnt model. Therefore, it is crucial to allow the model to incrementally expand, and to learn from data of new classes. This ability is referred to as *class-incremental learning* (CIL) in the literature [32, 3].

⋆ Yihong Gong is the corresponding author. # Xiaoyu Tao and Xinyuan Chang are co-first authors.



(a) Base model (b) Distillation (LUCIR) (c) Ours (TPCIL)

Fig. 1. *t*-SNE visualization of the comparison between TPCIL and the distillation approach in classifying the base class exemplars. We report the test accuracy on the base class test set during incremental learning. (a) Initially, the base class exemplars are well separated in feature space. (b) The distillation approach fails to maintain the feature space topology of the base class exemplars at the beginning of incremental learning, where *catastrophic forgetting* is clearly identified at *epoch 1* with severe degradation of the base class test accuracy. As a result, it has to take a much longer time (*e.g.*, *40* epochs here) to re-learn discriminative features for the exemplars. (c) TPCIL uses the *topology-preserving loss* (TPL) that maintains the topology of these old class exemplars, which can avoid forgetting during the entire incremental learning phase

CIL aims to incrementally learn a unified classifier to recognize *new* classes without forgetting the *old* ones at the same time. This problem is usually studied under a practical condition that the training set of old classes is unavailable when learning new classes [32]. As a consequence, it is prohibitive to retrain the model on the joint training set of both old and new classes. A straight-forward approach is to directly finetune the model on new class data. However, it is prone to *catastrophic forgetting* (CF) [10], where classification accuracies on old classes deteriorate drastically during finetuning.

To tackle catastrophic forgetting, a number of CIL methods [32, 3, 42, 13] adopt the *knowledge distillation* [12] technique to preserve the old class knowledge contained in the network's output logits. Knowledge distillation was originally proposed for transferring 'dark knowledge' from the teacher model to a student model [12, 43]. LwF [19] introduces this idea to incremental learning to alleviate the forgetting of the old tasks' knowledge when learning a new task. When applying to CIL, one typically stores a smaller set of exemplar images representative of old classes, and incorporates the distillation loss with the classification loss (i.e., cross-entropy) for learning from new class training samples.

Although the distillation approaches can mitigate forgetting to some extent, they face the bias problem [13, 42] caused by imbalanced number of old/new class training samples, which hurts the recognition performance [9]. Moreover, it is observed in our experiments that the distillation-based methods seem to *forget* the old knowledge at first and then re-learn the knowledge from the old class exemplars during incremental learning, which is termed the *start-all-over* phenomenon, as shown in Fig. 1 (b). As a result, it takes more additional epochs to re-acquire the old class knowledge. Besides, excessive re-learning also increases

the risk of overfitting to the old class exemplars. These issues restrict the ability of incremental learning from a (potentially) infinite sequence of new classes.

To solve the above problems, in this paper, we propose a cognitive-inspired Topology-Preseving CIL (TPCIL) method. Recent advances in cognitive science reveal that forgetting is caused by the disruption of the topology of human visual working memory [39, 6]. Analogously, for deep CNNs, we have also observed that catastrophic forgetting occurred together with the disruption of the feature space topology once learning new classes. Based on these discoveries, we endeavor to preserve the old class knowledge and mitigate forgetting by maintaining the topology of CNN's feature space. We model the topology using an *elastic Hebbian graph* (EHG) constructed with *competitive Hebbian learning* (CHL) [28]. During CIL, we impose new constraints, namely the *topology-preserving loss* (TPL) on EHG, to penalize the changes of its topological connections.

We conduct comprehensive experiments on popular image classification benchmarks CIFAR100, ImageNet, and subImageNet, and compare TPCIL with the state-of-the-art CIL methods. Experimental results demonstrate the effectiveness of TPCIL for improving recognition performance in a long sequence of incremental learning. To summarize, our main contributions include:

- We propose a neuroscience inspired, topology-preserving framework for effective class-incremental learning with less forgetting.
- We construct an *elastic Hebbian graph* (EHG) by *competitive Hebbian learning* to model the topology of CNN's feature space.
- We design the *topology-preserving loss* (TPL) to maintain the feature space topology and mitigate forgetting.

2 Related Work

There are two branches in recent incremental learning studies. The *multi-task* incremental learning [30, 19] aims at learning a sequence of independent tasks, each of which is assigned a specific classifier, while the *single-task* incremental learning [27, 38] employs a unified classifier to treat the entire incremental learning process as one task. The CIL focused by this paper belongs to the single-task incremental learning, where only one classification head is incrementally learnt to recognize all encountered data batches of different classes.

2.1 Multi-task Incremental Learning

The multi-task incremental learning [30] assumes the task identity is always known during training and testing. The model is required to learn new tasks without degrading the old tasks' performance. Research works usually adopt the following strategies to mitigate forgetting: (1) regularization strategy [14, 45, 17, 19], (2) architectural strategy [26, 25, 44, 35, 1], and (3) rehearsal strategy [23, 4, 36, 46]. Regularization strategy imposes regularization on the network weights or outputs when learning the new tasks. For example, EWC [14] and

SI [45] impose constraints on the network weights, penalizing changing of the weights important to old tasks. Architectural strategy dynamically modifies the network's structures by expanding, pruning [21, 47] or masking the neural connections. For example, PackNet [26] creates free parameters for new tasks by network pruning. HAT [35] learns the attention masks to constrain the weights for old tasks when learning new tasks. Rehearsal strategy periodically replay the memory for the past experiences of the old tasks to the network when learning new tasks. For example, GEM [23] uses an external memory to store a small set of old tasks' exemplar images and use them to constrain the old tasks' losses during incremental learning. DGR [36] and LifelongGAN [46] use a generative model to memorize the old tasks' data distribution, with a generative adversarial network learnt to produce pseudo training samples of old tasks.

In short, the *multi-task* methods perform incremental learning in task-level with task-specific classifiers. As a consequence, these methods can not be directly used by CIL, which only has a single, incrementally expanded classifier.

2.2 Class-Incremental Learning

Most *class-incremental learning* (CIL) works [32, 3, 42, 13] alleviates forgetting using the knowledge distillation [12, 43, 31, 18] technique, which is initially introduced by LWF [19] for multi-task incremental learning. An earlier work iCaRL [32] decouples the learning of the classifier and the feature representation, where the classifier is implemented by the nearest matching of the pre-stored exemplars in an *episodic memory*. When learning the representation for the new classes, the a distillation loss term is added to the cross-entropy loss function to maintain the representations of the old class exemplars. A later work EEIL [3] learns the network in an end-to-end fashion with the *cross-distillated loss*. It overcomes the limitation of iCaRL by learning the representation and the classifier jointly. More recent CIL studies [42, 13, 9, 37] reveal the critical *bias* issue caused by the imbalanced number of training samples of old and new classes, where the classification layer's weights and logits are biased towards new classes after incremental learning. To eliminate the bias, LUCIR [13] normalizes the feature vectors and the weights of the classification layer, adopts the cosine similarity metric, and applies distillation to the feature space rather than the output logits. BIC [42] develops a bias correction technique that learns a linear model to unify the distribution of the output logits. IL2M [9] proposes a dual-memory approach that finetunes the model without the distillation loss. It stores the exemplars and the statistics of historical classes to rectify the prediction scores.

In short, the distillation-based CIL methods maintain the distribution of the output/feature logits [32, 3, 42, 13] for the old class exemplars. However, it is observed that such kind of objective is not well achieved during incremental learning, where the old class knowledge is likely forgotten at the beginning of incremental learning (see Fig. 1). Besides, as the exemplar set is typically randomly sampled from the old class training set, it is only a rough approximation of the data distribution. Recent studies in *few-shot class-incremental learning* [37] show that the knowledge can be well preserved by learning the

topology of the feature space manifold, even when the manifold is non-uniform and heterogeneous. Different from the above approaches, TPCIL maintains the feature space topology by constraining the relations of the representative points, while allowing the shift of the representatives to adapt to new classes.

3 Topology-Preserving Class-Incremental Learning

3.1 Problem Definition

The *class-incremental learning* (CIL) problem is defined as follows. Let X, Y, and Z denote the training set, the label set, and the test set, respectively. A CNN model θ is required to incrementally learn a unified classifier from a sequence of training sessions $X^1, X^2, \cdots, X^t, X^{t+1}, \cdots$, where $X^t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$ is the labeled training set of the t-th session with N_t samples, and x_i^t and $y_i^t \in Y^t$ are the i-th image and its label, respectively. Y^t is the disjoint label set at session t, s.t. $\forall p \neq q,\ Y^p \cap Y^q = \varnothing$. At session $(t+1)$, a model θ^{t+1} is learnt from X^{t+1}, without the presence of the old class training sets $X^1, X^2, \cdots, X^t$. Then θ^{t+1} is evaluated on the union of all the encountered test sets $\bigcup_{j=1}^{t+1} Z^j$.

3.2 Overall Framework

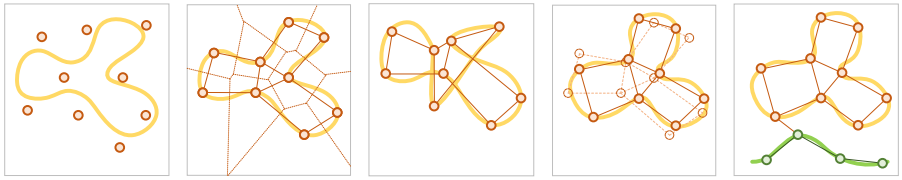
A CNN can be regarded as the composition of a feature extractor $f(\cdot;\theta)$ with parameters θ and a classification layer with a weight matrix W. Given an input x, CNN outputs $o(x;\theta) = W^\top f(x;\theta)$, which is followed by a softmax layer to produce multi-class probabilities. Let $\mathcal{F} \subseteq \mathbb{R}^n$ denotes the feature space defined by $f(\cdot;\theta)$. Initially, we train θ^1 on the base class training set X^1 with the cross-entropy loss. Then, we incrementally finetune the model on $X^2, \cdots, X^t, X^{t+1}, \cdots$, to get $\theta^2, \cdots, \theta^t, \theta^{t+1}, \cdots$. At session $(t+1)$, the output layer is expanded for new classes by adding $|Y^{t+1}|$ new neurons. Directly finetuning θ^{t+1} on X^{t+1} will overwrite old weights in θ^t important for recognizing old classes, which disrupts the feature space topology and causes catastrophic forgetting, with a degradation of the recognition performance on $\bigcup_{j=1}^{t} Y^j$.

In this paper, we alleviate forgetting by maintaining the feature space topology for the old classes. To achieve this purpose, we first model the feature space topology using the *elastic Hebbian graph* (EHG), and then propose the *topology-preserving loss* (TPL) term to penalize changing of the feature space topology represented by EHG. Let G^t denote the EHG constructed at session t. The overall loss function at the next session $(t+1)$ is defined as:

$$\ell(X^{t+1}, G^t; \theta^{t+1}) = \ell_{CE}(X^{t+1}, G^t; \theta^{t+1}) + \lambda \ell_{TPL}(G^t; \theta^{t+1}). \tag{1}$$

In the above equation, ℓ_{CE} is the standard cross-entropy loss:

$$\ell_{CE}(X^{t+1}, G^t; \theta^{t+1}) = \sum_{(x,y)} -\log \hat{p}_y(x), \tag{2}$$



(a) (b) (c) (d) (e)

Fig. 2. Conceptual visualization of the topology-preserving mechanism. The golden curve stands for the feature space manifold; The circles and solid lines indicate the vertices and edges of EHG, respectively. (a) N points are randomly picked to initialize EHG's vertices. (b) By *competitive Hebbian learning* (CHL), the feature space is partitioned into N disjoint *Voronoi cells*, each of which is encoded by a vertex. The neighborhood relations is described by the connections between the vertices. (c) Finetuning CNN for new classes may greatly change the neighborhood relationship of vertices and disrupt the feature space topology. (d) The TPL term compels EHG to maintain the relations of the vertices. (e) After learning new class, EHG grows by inserting new vertices. Then all vertices are updated by CHL and the similarities are re-computed

where (x, y) denotes a training image and its label, and $\hat{p}_y(x)$ is the CNN's predicted probability of label y given input x. We use X^{t+1} as well as the old class images assigned to EHG's vertices (see Section 3.3 for details) for training. ℓ_{TPL} is the proposed TPL loss term applied to G^t. The hyper-parameter λ is used for controlling the strength of TPL. We elaborate our approach in the following subsections.

3.3 Topology Modelling via Elastic Hebbian Graph

An effective way to model the topology of a feature space is to perform *Competitive Hebbian learning* (CHL) [28] on the feature space manifold. CHL can learn a set of points representative of any manifold (e.g., non-uniform), and is proved to well preserve the topological structure [29]. To enable topology modelling for CIL and cooperate with CNN, we design the *elastic Hebbian graph* (EHG) which is constructed using CHL. The detailed algorithm is described as follows.

For computational stability, we normalize the feature space and adopt the *cosine similarity* metric. Let $\bar{\cdot}$ denotes the normalization operation, where $\bar{\mathbf{f}} = \mathbf{f}/\|\mathbf{f}\|$. Given the normalized feature space $\bar{\mathcal{F}}$, the EHG is defined as $G = \langle V, E \rangle$, where $V = \{\bar{\mathbf{v}}_1, \cdots, \bar{\mathbf{v}}_N | \bar{\mathbf{v}}_i \in \bar{\mathcal{F}}\}$ is the set of N vertices representative of $\bar{\mathcal{F}}$, and E is the edge set describing the neighborhood relations of the vertices in V. Each vertex $\bar{\mathbf{v}}_i$ is the centroid vector representing the feature vectors within a neighborhood region $\mathcal{V}_i$, which is refered to as the *Voronoi cell* [29]:

$$\mathcal{V}_i = \{\bar{\mathbf{f}} \in \bar{\mathcal{F}} | \bar{\mathbf{f}}^\top \bar{\mathbf{v}}_i \geq \bar{\mathbf{f}}^\top \bar{\mathbf{v}}_j, \ \forall j \neq i\}, \ \forall i. \tag{3}$$

To get $\bar{\mathbf{v}}_i$, we first randomly initialize its value by picking a random position in feature space, as shown in Fig. 2 (a). Then we update $\bar{\mathbf{v}}_i$ iteratively using the

following normalized *Hebbian* rule:

$$\mathbf{v}_i^* = \bar{\mathbf{v}}_i + \epsilon \cdot e^{-k_i/\alpha}(\bar{\mathbf{f}} - \bar{\mathbf{v}}_i),\ \bar{\mathbf{v}}_i^* = \mathbf{v}_i^*/\|\mathbf{v}_i^*\|,\ i = 1, \cdots, N, \tag{4}$$

where $\bar{\mathbf{v}}_i^*$ denotes the updated vertex, and $e^{-k_i/\alpha}$ is the decay function to scale the updating step. The decay factor is measured by the proximity rank k_i, where $\bar{\mathbf{v}}_i$ is the k_i-th nearest neighbor of $\bar{\mathbf{f}}$ among all vertices in V. The hyper-parameter ϵ is the learning rate, and α controls the strength of the decay. Eq. (4) ensures the vertex nearest to $\bar{\mathbf{f}}$ has the largest adapting step towards $\bar{\mathbf{f}}$, while other vertices are less affected. We execute Eq. (4) until $\bar{\mathbf{v}}_i^*$ is converged.

With the updated vertex set V, we may construct the corresponding *Delaunay graph* as in [29] to model the neighborhood relations of the vertices, as shown in Fig. 2 (b). However, it is difficult to directly constrain the Delaunay graph under the gradient descent framework, as the adjacency of vertices are changed by the Hebbian rule, which is hard to cooperate with CNN's back-propagation. Alternatively, we convert G as a similarity graph for ease of optimization. Each edge e_{ij} is assigned with a weight s_{ij}, which is the similarity between $\bar{\mathbf{v}}_i$ and $\bar{\mathbf{v}}_j$:

$$s_{ij} = \bar{\mathbf{v}}_i^\top \bar{\mathbf{v}}_j. \tag{5}$$

In this way, the changing of G can be back-propagated to CNN and optimized with the gradient decent algorithm. For computing the observed values of each vertex at the next incremental learning session, we assign $\bar{\mathbf{v}}_i$ with an image u_i drawn from the old training samples whose feature vector is the closet to $\bar{\mathbf{v}}_i$.

When applying EHG to incremental learning, we first construct the graph using the base class training data. When the training of θ^1 is completed, we extract the set of normalized feature vectors on X^1, by which we have $\bar{F}^1 = \{\bar{f}(x;\theta^t)|\forall(x,y) \in X^1\}$. $\bar{F}^1$ forms the feature space manifold of the base classes. We compute EHG G^1 on $\bar{F}^1$ using Eq. (4) and Eq. (5). G^1 is stored to alleviate forgetting at the next session. Iteratively, at session $(t+1)$, after learning θ^{t+1}, we grow and update the pre-stored EHG G^t to make it consistent with the adapted feature space. We insert K new vertices $\{\bar{\mathbf{v}}_{N+1}, \cdots, \bar{\mathbf{v}}_{N+K}\}$ to get V^{t+1}, and then update all vertices on $\bar{F}^{t+1}$ using Eq. (4). After that, the similarities are recomputed to get E^{t+1}. Fig. 2 (e) illustrates the growth of EHG. The topology of the newly formed manifold for new classes is modelled by new vertices and integrated into the EHG.

3.4 Topology-Preserving Constraint

At session $(t+1)$, given EHG $G^t = \langle V^t, E^t \rangle$, when catastrophic forgetting occurs, G^t is distorted with the disruption of the old edges, as shown in Fig. 2 (c). To alleviate forgetting, the original connections in G^t should be maintained when finetuning CNN on X^{t+1}. This is achieved by constraining the neighboring relations of vertices described by the edges' weights (i.e., similarities) in E^t. For this purpose, one approach is to maintain the rank of the edges' weights during learning. However, it is difficult and inefficient to optimize the nonsmooth global ranking [2], while the local ranking can not well preserve the global relations.

Alternatively, we can measure the changing of the neighboring relations by computing the correlation between the initial and observation values of the edges' weights. A lower correlation indicates a higher probability that the rank of the edges has changed during learning, which should be penalized. On this basis, we define the *topology-preserving loss* (TPL) term as:

$$\ell_{TPL}(G^t;\theta^{t+1}) = -\frac{\sum_{i,j}^{N}(s_{ij}-\frac{1}{N^2}\sum_{i,j}^{N}s_{ij})(\tilde{s}_{ij}-\frac{1}{N^2}\sum_{i,j}^{N}\tilde{s}_{ij})}{\sqrt{\sum_{i,j}^{N}(s_{ij}-\frac{1}{N^2}\sum_{i,j}^{N}s_{ij})^2}\sqrt{\sum_{i,j}^{N}(\tilde{s}_{ij}-\frac{1}{N^2}\sum_{i,j}^{N}\tilde{s}_{ij})^2}}, \tag{6}$$

where $S = \{s_{ij}|1 \le i,j \le N\}$ and $\tilde{S} = \{\tilde{s_{ij}}|1 \le i,j \le N\}$ are the sets of the initial and observation values of edges' weights in E^t, respectively. The active value $\tilde{s}_{ij}$ is estimated by:

$$\tilde{s}_{ij} = \tilde{\mathbf{f}}_i^\top \tilde{\mathbf{f}}_j = \bar{f}(u_i;\theta^{t+1})^\top \bar{f}(u_j;\theta^{t+1}), \tag{7}$$

where u_i and u_j are the pre-stored images assigned to $\bar{\mathbf{v}}_i$ and $\bar{\mathbf{v}}_j$, respectively. As $\bar{\mathbf{v}}_i$ encodes the i-th region in feature space, the TPL term implicitly maintains the adjacency of these regions. Another choice for the loss term is to penalize the l_1 or l_2 norms of the similarities. In our experiments, we found such restrictions are not as flexible as the correlation form and behave worse, since they do not allow a linear changing of the similarities' scale.

Rather than penalizing the shift of EHG's vertices in feature space, TPL penalizes the changing of the topological relations between the vertices, while allowing the reasonable shift of vertices. Such constraint is 'soft', easier to optimize, which makes the EHG structure 'elastic' and does not interfere the learning of new classes. Fig. 2 (d) illustrates the effect of TPL.

3.5 Optimization

TPCIL integrates a CNN model and an EHG G^t, where G^t is used to preserve the topology of CNN's feature space manifold. It is noteworthy that the CNN model is trained with the *minibatch stochastic gradient descent* (minibatch SGD) algorithm, while G^t is constructed and updated with the *competitive Hebbian learning* (CHL). It is less efficient to update the vertices of G^t using Eq. (4) at each minibatch iteration, as the features obtained at intermediate training sessions have not been fully optimized. Therefore, we learn G^t after the training of CNN is completed. G^t is then used for the next incremental session $(t+1)$.

3.6 Comparison with the Distillation-based Approaches

In contrast to our approach that maintains the feature space topology, other CIL works [3, 13, 42] are mostly based on knowledge distillation, where a distillation loss term is appended to the cross-entropy loss:

$$\ell(\tilde{X}^{t+1};\theta^{t+1},\theta^t) = \ell_{CE}(\tilde{X}^{t+1};\theta^{t+1}) + \gamma\ell_{DL}(\tilde{X}^{t+1};\theta^{t+1},\theta^t), \tag{8}$$

where $\tilde{X}^{t+1} = X^{t+1} \cup M^t$ denotes the joint set of new class training samples X^{t+1} and the old class exemplars M^t, and θ^t and θ^{t+1} are the parameter sets achieved at session t and $(t+1)$, correspondingly. The distillation loss term ℓ_{DL} is applied to the network's output logits corresponding to the old classes [32, 3]:

$$\ell_{DL}(\tilde{X}^{t+1}; \theta^{t+1}, \theta^t) = - \sum_{(x,y)\in\tilde{X}^{t+1}} \sum_{c=1}^{C^t} \frac{e^{-o_c(x;\theta^t)/T}}{\sum_{j=1}^{C^t} e^{-o_j(x;\theta^t)/T}} \log \frac{e^{-o_c(x;\theta^{t+1})/T}}{\sum_{j=1}^{C^t} e^{-o_j(x;\theta^{t+1})/T}}, \tag{9}$$

where $C^t = |Y^t|$ is the number of the old classes and T (e.g., $T = 2$) is the temperature for distillation. Another distillation approach is to apply the distillation loss to the feature space, which is called *feature distillation loss* [13]:

$$\ell_{FDL}(\tilde{X}^{t+1}; \theta^{t+1}, \theta^t) = \sum_{(x,y)\in\tilde{X}^{t+1}} (1 - \bar{f}(x;\theta^t)^\top \bar{f}(x;\theta^{t+1})), \tag{10}$$

where $\bar{f}(x;\theta^t)$ denotes the normalized feature vector.

The distillation losses ℓ_{DL} and ℓ_{FDL} penalize the changing of the output logits or feature vectors computed by the old model. Such a restriction is too strict and difficult to satisfy, as the cross-entropy loss ℓ_{CE} dominantly brings adaptation to new classes in the feature or output space. We have observed the *start-all-over* phenomenon, where the features of the base class exemplars in M^t are 'forgotten' at the beginning of incremental learning, as illustrated in Fig. 1 (b). In comparison, the TPL term in Eq. (6) constrains the neighboring relations between the EHG vertices, allowing the feature space to adapt to new classes more freely without losing discriminative power. In this way, the plummeting of the old classes' recognition performance at the initial training iterations can be avoid. Detailed experimental comparisons are described in Section 4.3.

4 Experiments

We conduct comprehensive experiments under the CIL setting in [13] on three popular image classification datasets CIFAR100 [15], ImageNet [7] and subImageNet [32, 13]. Following [13], for each dataset, we choose half of the classes as the base classes for the base session and divide the rest of the classes into 5 or 10 incremental sessions. Detailed setups are described as follows.

4.1 Datasets and Experimental Setups

CIFAR100. It contains 60,000 natural RGB images of the size 32×32 over 100 classes, including 50,000 training and 10,000 test images. We follow the protocols in [13] to process the dataset, where 50 classes are selected as the base classes, and the rest 50 classes are equally divided for incremental learning phases. We randomly flip each image for data augmentation during training.

ImageNet. The large-scale ImageNet (1k) dataset has 1.28 million training and 50,000 validation images over 1000 classes. We select 500 classes as the base

classes and split the rest 500 classes for incremental learning. We randomly flip the image and crop a 224×224 patch for data augmentation during training, and use the single-crop center image patch for testing.

SubImageNet. This dataset is the 100-class subset of ImageNet, which contains about 130,000 images for training and 5,000 images for testing. We select 50 classes as the base classes and equally divide the rest 50 classes for incremental learning. For data augmentation, we use the same technique as ImageNet.

Experimental Setups. All the experiments are performed using PyTorch. As in [13], we choose the popular 32-layer ResNet as the baseline CNN for CIFAR100 and the 18-layer ResNet for ImageNet and subImageNet, respectively.

Initially, we train the base model for 120 epochs using minibatch SGD with the minibatch size of 128. The learning rate is initialized to 0.1 and decreased to 0.01 and 0.001 at epoch 60 and 100, respectively. At each incremental learning session, we finetune the model for 90 epochs, where the learning rate is initially set to 0.01 for CIFAR100 and 5e-4 for ImageNet and subImageNet, respectively, and decreased by 10 times at epoch 30 and 60. We set the hyper-parameter $\lambda = 15$ in Eq. (1) for CIFAR100 and $\lambda = 10$ for subImageNet and ImageNet, respectively. For EHG, we insert 20 vertices for each new class, which leads to 2,000 vertices for CIFAR100 and subImageNet, and 20,000 vertices for ImageNet. We set $\epsilon = 0.1$ and $\alpha = 10$ in Eq. (4). At the end of each session, we evaluate the model on the union of all the encountered test sets.

We compare TPCIL with the representative CIL methods, including the classical iCARL [32], EEIL [3] and recent state of the arts LUCIR [13] and BiC [42]. To show the effectiveness of alleviate forgetting, we also directly finetune the CNN model using both the new class training samples and the old class exemplars without forgetting-reduction techniques. We denote this baseline method as "Ft-CNN". For the upper-bound, we follow [13] and retrain the model at each session on a joint set of all training images of encountered classes, which is denoted as "Joint-CNN". The distillation temperature in Eq. (9) is set to $T = 2$. For fair comparisons, we use the equal number of old class exemplars for all comparative methods. All results are averaged over 5 runs. We report the top-1 test accuracy of each session, as well as the accuracy averaged over all sessions.

4.2 Comparison Results

Fig. 3 shows the comparison results between TPCIL and other CIL methods. Each curve reports the changing of the test accuracy at each session. The green curve stands for the baseline "Ft-CNN", while the yellow curve indicates the *upper-bound* "Joint-CNN". The orange curve reports the accuracy achieved by TPCIL, while the cyan, blue and purple curves report the accuracies of LUCIR, iCARL and EEIL, respectively. We summarize the results as follows:

– For training with both the 5 and 10-session settings on all datasets, TPCIL greatly outperforms all other CIL methods on each incremental session by a large margin, and is the closest to the upper-bound joint training method. By comparing each pair of the orange and cyan curves in Fig. 3, we observe that

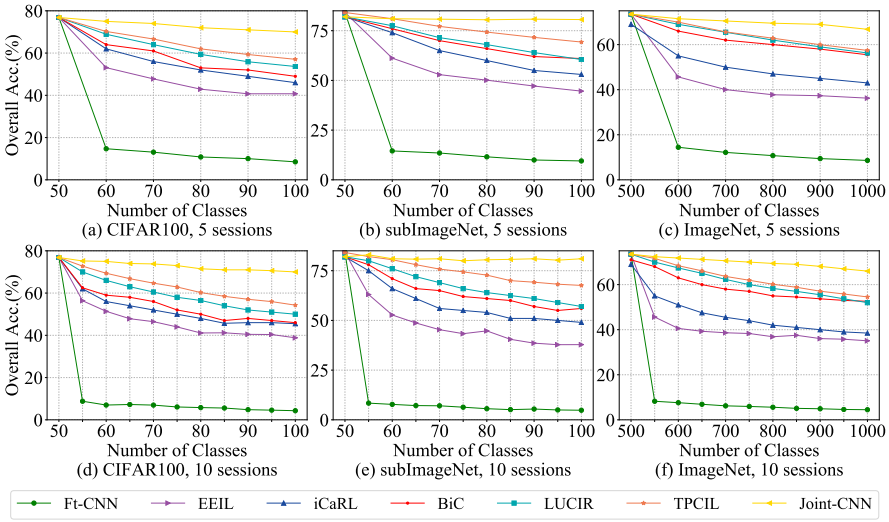


Fig. 3. Comparison results on CIFAR100, subImageNet, and ImageNet under the 5-session (a)-(c) and 10-session (d)-(f) settings. Noting that the original EEIL in [3] uses more data augmentation techniques to boost the performance, which has higher accuracy than iCaRL. In our experiments, we apply the same data augmentation operation to all methods for fair comparisons, which causes the accuracy of EEIL lower

TPCIL achieves higher accuracy than the state-of-the-art LUCIR. Moreover, the superiority of TPCIL is more obvious after learning all the sessions. It shows the effectiveness of TPCIL for long-term incremental learning.

- On CIFAR100, TPCIL achieves the average accuracy of **65.34%** and **63.58%** with the 5 and 10-session settings, respectively. In comparison, the second-best LUCIR achieves the average accuracy of 63.42% and 60.18%, correspondingly. TPCIL outperforms LUCIR by up to **3.40%**. After learning all the sessions, TPCIL greatly outperforms LUCIR by up to **4.28%**.
- On subImageNet, TPCIL has the average accuracy of **76.27%** and **74.81%** with the 5 and 10-session settings, respectively, while the second-best LUCIR has the average accuracy of 70.47% and 68.09%, correspondingly. TPCIL greatly outperforms LUCIR by up to **6.72%**. Furthermore, at the last session, TPCIL significantly outperforms LUCIR by up to **10.60%**.
- On ImageNet, with the 5-session CIL setting, the average accuracy of TPCIL is **64.89%**, exceeding the second-best LUCIR (64.34%) by up to **0.55%**. With the 10-session setting, TPCIL achieves the average accuracy of **62.88%**, surpassing LUCIR (61.28%) by up to **1.60%**. After learning all the sessions, TPCIL outperforms LUCIR by up to **1.43%** and **2.53%**, correspondingly.

In addition to the 5 and 10-session setting, we have also evaluated 1 and 2-session incremental learning and permuted the order of the sessions. We find

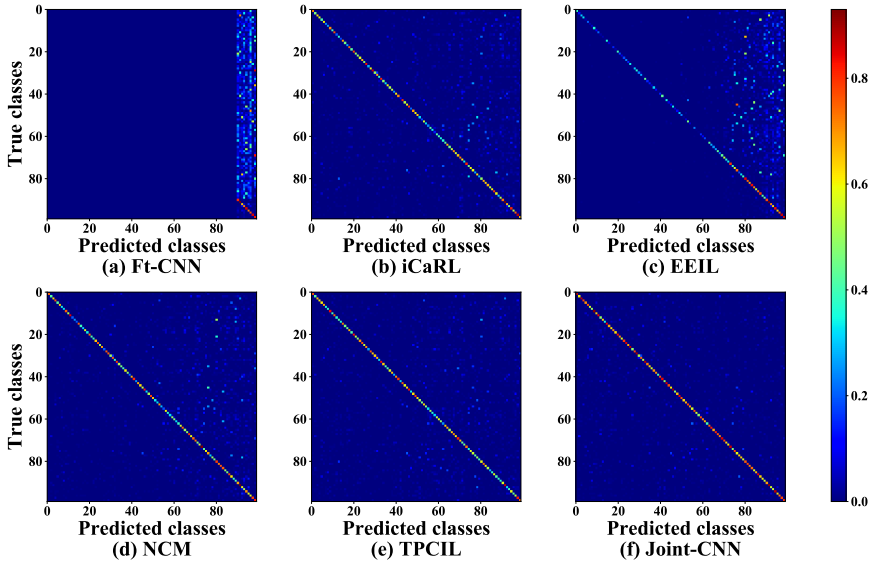


Fig. 4. Confusion matrices of different methods on CIFAR100 under the 5-session setting. The horizontal/vertical axes indicate the predicted/true classes, respectively. The color bar at the right side indicates the activation intensity

the rank of the methods' accuracies remains the same, by which we can draw the same conclusion for the comparison results.

Fig. 4 shows the confusion matrices of classification results produced by different CIL methods. In Fig. 4 (a), simply finetuning for new class will cause severe misclassifications, where the old class samples are prone to be classified as new classes. The iCARL (b), EEIL (c), and LUCIR (d) methods can correct some misclassified cases, but there are still many unsatisfactory activations outside the diagonal. In comparison, our TPCIL (e) produces a much better confusion matrix, where the activations are mostly distributed at the diagonal, which is the closest to the upper-bound Joint-CNN (f) method. It demonstrates the effectiveness of TPCIL for alleviating forgetting and improving the accuracy.

4.3 Analysis of the TPCIL Components

We perform ablation studies on CIFAR100 under the 5-session incremental learning setting to analyse the effect of TPCIL components, as described in follows.
The effect of different loss terms. We explore how different loss terms affect the recognition performance, including the *distillation loss* (DL) and *feature distillation loss* (FDL) in Section 3.6, and different choices (i.e., Eq. (6), l_1 or l_2) of the TPL form. The experiments are performed on CIFAR100 under the 5-session setting. For fair comparisons, all loss terms use the same set of representative

Table 1. Comparison of the test accuracy achieved by different loss terms

Method	encountered classes						avg. acc.
	50	60	70	80	90	100	
finetuning	76.84	51.90	49.66	43.23	40.21	39.40	50.21
DL(Eq. (9))	76.84	61.57	55.27	48.76	46.04	45.20	55.61
FDL(Eq. (6))	76.84	66.32	62.11	55.73	51.56	50.74	60.55
TPL	76.84	**70.23**	**66.64**	**61.99**	**59.32**	**57.04**	**65.34**
TPL(l_1)	76.84	68.33	65.21	61.21	57.63	55.80	64.17
TPL(l_2)	76.84	66.60	63.23	59.11	56.64	54.08	62.75
TPL+DL	76.84	63.72	57.44	48.75	45.31	45.07	56.19
TPL+FDL	76.84	68.60	61.04	52.33	47.41	45.76	58.66

Table 2. Comparison of different exemplar generation techniques

Method	the number of exemplars/class				
	1	2	5	10	20
Random	33.86	45.83	48.89	58.26	64.70
k-means	39.45	48.75	52.24	60.67	65.03
EHG	**42.26**	**51.27**	**52.89**	**61.47**	**65.34**

images given by EHG. Additionally, we also combine TPL with DL and FDL and evaluate their performances.

Table 1 reports the comparison results. The TPL term achieves the best accuracy after learning all sessions, exceeding FDL significantly by up to **6.3%** and DL by up to **11.84%**. While the combinations of TPL and distillation losses degrade the performance of using TPL alone. It demonstrates that maintaining the feature space topology is more effective to alleviate forgetting than maintaining the stability of the output logits or feature vectors using distillation.

Comparison of different exemplar generation techniques. In TPCIL, the EHG vertices learned by Hebbian rules can be seen as the exemplars of the feature space. Alternatively, we can randomly sample points in feature space, or run a clustering method (e.g., k-means) and treat the cluster centroids as the exemplars. Table 2 compares the average test accuracy achieved by the three exemplar generations approaches under different number of the exemplars. Apparently, using a large number of exemplars can achieve higher accuracy even for random sampling, while EHG behaves better especially when the number of exemplars is small, thanks to the topology-preservation mechanism [29].

The effect of the number of the exemplars. In the experiments, the CIL methods use an external memory to store the old class exemplars. Though storing more representatives is helpful for the recognition performance, it also brings more memory overhead. Table 3 reports the average accuracy achieved by using different numbers of vertices/exemplars *per class*. It is observed that the test accuracy is prone to be saturated when the number of exemplars per class is greater than 30. For a better trade-off, we use 20 exemplars per class. Besides,

Table 3. Average accuracy of different methods with different number of exemplars

Method	the number of exemplars				
	10	20	30	40	50
iCaRL [32]	52.5	56.5	60.0	61.0	62.0
EEIL [3]	41.8	50.3	55.2	57.1	59.7
LUCIR [13]	61.0	64.0	64.5	65.5	66.0
TPCIL (ours)	**61.5**	**65.3**	**66.2**	**66.5**	**67.0**

Table 4. Average accuracy with different λ on CIFAR100 with the 5-session setting

λ	0	0.1	1	5	10	15	50	100
Average acc.	22.34	58.39	63.07	64.99	65.33	65.34	64.48	61.99

we can also observe that TPCIL achieves better performance than other methods when fixing the memory size, which demonstrates the efficiency of TPCIL.

4.4 Sensitivity Study of the Hyper-parameter λ

The hyper-parameter λ in Eq. (1) controls the strength of TPL term. We perform the sensitivity study to see how the recognition performance is influenced by changing λ. For other hyper-parameters ϵ and α in Eq. (4), we follow their settings in [29] and ensure the vertices of EHG well converged after *competitive Hebbian learning*. We run TPCIL on CIFAR100 with the 5-session setting and change λ in the range of $\{0.1, 1, 5, 10, 15, 50, 100\}$. Table 4 shows the average test accuracy achieved by different values of λ. We observe that with the increasing of λ within a reasonably wide range, the average test accuracy is improved, indicating the effectiveness of TPL. While too large λ (e.g., $\lambda = 100$) could weaken the contribution of the classification loss and hurt the accuracy.

5 Conclusion

This work focuses on the CIL task and addresses the catastrophic forgetting problem from a new, cognitive-inspired perspective. To alleviate forgetting, we put forward to preserve the old class knowledge by maintaining the topology of feature space. We propose a novel TPCIL framework, which uses an EHG graph to model the topology of the feature space manifold, and a TPL term to constrain EHG, penalizing the changing of the topology. Extensive experiments demonstrate that the proposed TPCIL greatly outperforms state-of-the-art CIL methods. In future works, we will generalize TPCIL to more applications.

Acknowledgements. This work is sponsored by National Key R&D Program of China under Grand No.2019YFB1312000, National Major Project under Grant No.2017YFC0803905 and SHAANXI Province Joint Key Laboratory of Machine Learning.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV (2018)
2. Burges, C.J., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: NeurIPS (2007)
3. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 233–248 (2018)
4. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018)
5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
6. Chen, L.: The topological approach to perceptual organization. Visual Cognition **12**(4), 553–637 (2005)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
8. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. arXiv preprint arXiv:1801.07698 (2018)
9. Eden, B., Adrian, P.: Il2m: Class incremental learning with dual memory. In: ICCV (2019)
10. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
12. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. Computer Science **14**(7), 38–39 (2015)
13. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR (2019)
14. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
15. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS (2012)
17. Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming catastrophic forgetting by incremental moment matching. In: NeurIPS (2017)
18. Lee, S., Song, B.C.: Graph-based knowledge distillation by multi-head attention network. In: BMVC (2019)
19. Li, Z., Hoiem, D.: Learning without forgetting. T-PAMI **40**(12), 2935–2947 (2018)
20. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: CVPR (2017)
21. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. In: ICLR (2019)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
23. Lopez-Paz, D., et al.: Gradient episodic memory for continual learning. In: NeurIPS (2017)

24. Ma, Z., Wei, X., Hong, X., Gong, Y.: Bayesian loss for crowd count estimation with point supervision. In: ICCV (2019)
25. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: ECCV (2018)
26. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: CVPR (2018)
27. Maltoni, D., Lomonaco, V.: Continuous learning in single-incremental-task scenarios. arXiv preprint arXiv:1806.08568 (2018)
28. Martinetz, T.M.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: International Conference on Artificial Neural Networks. pp. 427–434 (1993)
29. Martinetz, T., Schulten, K.: Topology representing networks. Neural Networks **7**(3), 507–522 (1994)
30. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks (2019)
31. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: CVPR (2019)
32. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR (2017)
33. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640 (2015)
34. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS (2015)
35. Serrà, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. arXiv preprint arXiv:1801.01423 (2018)
36. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NeurIPS (2017)
37. Tao, X., Hong, X., Chang, X., Dong, S., Xing, W., Yihong, G.: Few-shot class-incremental learning. In: CVPR (2020)
38. Tao, X., Hong, X., Chang, X., Gong, Y.: Bi-objective continual learning: Learning 'new' while consolidating 'known'. In: AAAI (February 2020)
39. Wei, N., Zhou, T., Zhang, Z., Zhuo, Y., Chen, L.: Visual working memory representation as a topological defined perceptual object. Journal of Vision **19**(7), 1–12 (2019)
40. Wei, X., Zhang, Y., Gong, Y., Zhang, J., Zheng, N.: Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In: ECCV (2018)
41. Wei, X., Zhang, Y., Gong, Y., Zheng, N.: Kernelized subspace pooling for deep local descriptors. In: CVPR (2018)
42. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR (2019)
43. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: CVPR (2017)
44. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)
45. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML (2017)
46. Zhai, M., Chen, L., Tung, F., He, J., Nawhal, M., Mori, G.: Lifelong gan: Continual learning for conditional image generation. In: ICCV (2019)
47. Zhuo, L., Zhang, B., Yang, L., Chen, H., Ye, Q., David, S.D., Ji, R., Guo, G.: Cogradient descent for bilinear optimization. In: CVPR (2020)