Exploring Data Geometry for Continual Learning

Zhi Gao[1], Chen Xu[2*], Feng Li, Yunde Jia[2,1], Mehrtash Harandi[3], Yuwei Wu[1,2*]
[1]Beijing Key Laboratory of Intelligent Information Technology,
School of Computer Science & Technology, Beijing Institute of Technology, China
[2]Guangdong Laboratory of Machine Perception and Intelligent Computing,
Shenzhen MSU-BIT University, China
[3]Department of Electrical and Computer Systems Eng., Monash University, and Data61, Australia
{gaozhi_2017,jiayunde,wuyuwei}@bit.edu.cn
xuchen@smbu.edu.cn, lifeng.passion@gmail.com, mehrtash.harandi@monash.edu

Abstract

Continual learning aims to efficiently learn from a non-stationary stream of data while avoiding forgetting the knowledge of old data. In many practical applications, data complies with non-Euclidean geometry. As such, the commonly used Euclidean space cannot gracefully capture non-Euclidean geometric structures of data, leading to inferior results. In this paper, we study continual learning from a novel perspective by exploring data geometry for the non-stationary stream of data. Our method dynamically expands the geometry of the underlying space to match growing geometric structures induced by new data, and prevents forgetting by keeping geometric structures of old data into account. In doing so, making use of the mixed curvature space, we propose an incremental search scheme, through which the growing geometric structures are encoded. Then, we introduce an angular-regularization loss and a neighbor-robustness loss to train the model, capable of penalizing the change of global geometric structures and local geometric structures. Experiments show that our method achieves better performance than baseline methods designed in Euclidean space.

1. Introduction

Unlike humans, artificial neural networks perform poorly to learn new knowledge in a continual manner. The tendency to lose the knowledge previously learned, known as *catastrophic forgetting*, is due to the fact that important parameters of a neural network for old data are changed to meet the objectives of new data. There have been many continual learning methods [12, 14, 23, 32, 49, 53], and their goal is to remember the knowledge from old data while effectively learning from new data. They have achieved impressive performance in alleviating catastrophic forgetting.

However, a long-lasting issue with existing methods is that data geometry is rarely studied in continual learning. Existing methods usually assume that data is Euclidean and they use Euclidean geometry to process the data stream. In fact, data in countless applications intrinsically has non-Euclidean geometric structures [2, 6]. Several studies show that non-Euclidean geometric structures can be better captured by particular forms of Riemannian geometry [15, 35]. For example, the hyperbolic geometry has a natural expressive ability for the hierarchical structure and is hence used successfully for fine-grained images [22, 29]. The spherical geometry is shown as a suitable choice for face images that have the cyclical structure [27, 46]. In addition to the geometric structures discussed above, natural data may be diverse and irregular in structure, *e.g.*, data exhibits hierarchical forms in some regions and cyclical forms in others [31, 40]. Overall, distortions produced when using Euclidean geometry for non-Euclidean geometric structures are overwhelming, causing the loss of semantic information, and hence resulting in inferior performance [3]. In this paper, we study how to attain suitable non-Euclidean geometry to capture the intrinsic geometric structures of data during continual learning.

To achieve our goal, we have to face two challenges (see Fig. 1). **(1)** Non-stationary stream of data will inevitably increase the complexity of intrinsic geometric structures. In other words, fixing the geometry of the underlying space cannot always match new and unseen data in continual learning. For example, more and more complex hierarchies in a data stream bring more leaf nodes, requiring a faster growing space volume with the radius, which conflicts with a fixed geometry [17]. **(2)** Old data is not accessible in con-

* Corresponding authors: Chen Xu and Yuwei Wu.

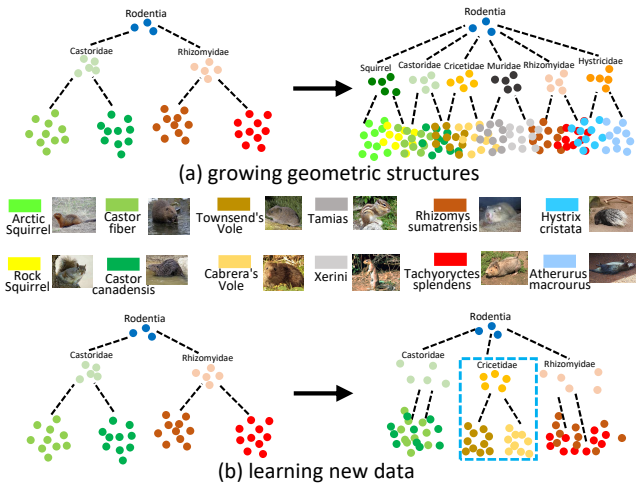

Figure 1. Illustrations of the two challenges of exploring data geometry in continual learning. Different color denotes different classes. (a) A fixed geometry cannot handle more and more complex hierarchy in a data stream. The geometric structure of leaf nodes is destroyed. (b) Learning from new data (in the blue dash box) will destroy the captured hierarchical structures of old data.

tinual learning, and learning from new data may destroy the captured geometric structures of old data, resulting in the catastrophic forgetting problem. Since geometric structures are characterized by distances or angles between instances, destroying the captured geometric structures may cause undesirable data distribution (*e.g.*, classes are not separable).

In this work, we use the mixed-curvature space to embed data, which is a product of multiple constant curvature spaces acting as submanifolds [19, 42]. The mixed-curvature space has shown to be superior to Euclidean space in some machine learning tasks, owing to its ability to capture non-Euclidean geometric structures [40, 43]. Examples include image classification [16], graph analysis [46], and information retrieval [48]. The geometry of a mixed-curvature space is determined by the number, dimension, and curvature of constant curvature spaces. By changing the geometry, we are able to adjust the mixed-curvature space to match specific geometric structures of data [52]. For example, positive curvatures are suitable for local cyclical structures, and negative curvatures are suitable for hierarchical structures in a region. Based on the mixed-curvature space, we restate the two challenges: **(1)** how to identify the suitable geometry of the mixed-curvature space for growing geometric structures, and **(2)** how to preserve the geometric structures of old data when learning from new data in the mixed-curvature space.

We introduce a geometry incremental search scheme to solve the first challenge. We build a submanifold pool by sampling subsets of coordinates from features, where the length of coordinates is the dimension of constant curvature spaces and features are projected to them using initial curvatures. Given new data, we select constant curvature spaces that contribute significantly to the current task to expand geometry of the mixed-curvature space. In this case, the growing geometric structures are well encoded. We introduce two loss functions, *i.e.*, an angular-regularization loss and a neighbor-robustness loss, to solve the second challenge. The angular-regularization loss penalizes the change of angles between any pair of instances to preserve global structures. The neighbor-robustness loss realizes within-class compactness and between-class separability in a neighbor to preserve the discriminative power of local structures. As a result, our method is capable of efficiently learning from new data and preventing forgetting of old data. Our method is evaluated on multiple continual learning settings, and experimental results show the effectiveness of our method.

In summary, our contributions are three-fold. (1) To the best of our knowledge, we are the first to explore data geometry for continual learning. Our method is efficient for learning from a non-stationary stream of data. (2) We introduce an incremental search scheme that identifies the suitable geometry for growing geometric structures of data. (3) We introduce an angle-regularization loss and a neighbor-robustness loss, capable of preserving geometric structures of old data in the mixed-curvature space.

2. Related Work

2.1. Continual Learning

Continual learning is mainly studied under two scenarios: task incremental learning [23, 51] and class incremental learning [36, 50]. In task incremental learning, a model is trained on incremental tasks with clear boundaries. At the test stage, the task ID is provided. In contrast, the class incremental learning targets a more challenging and practical setup, where the task ID is not provided at the test stage. In this paper, we focus on class incremental learning.

Existing techniques for continual learning can be broadly divided into three categories: replay-based methods, regularization-based methods, and architecture-based methods. Replay-based methods store a small amount of old data [8, 12, 36], gradients of old data [13, 30], features of old data [54], or train a generative model for old data [41, 53], and review them to tune the model when new data comes. Regularization-based methods penalize the update of features or parameters to avoid catastrophic forgetting. Representative schemes include measuring the importance of parameters [23, 51], projecting gradients onto a null space [24, 45], and distilling knowledge from the old model [28, 44]. Architecture-based methods allocate specific parameters to different data. When new data comes, some methods select parameters from a fixed supermodel [32, 39] and some methods add new parameters to an

expandable model [38, 49].

Different from existing methods that do not consider data geometry and use Euclidean space for all data, we explore data geometry by producing suitable manifolds for new data and ensuring that the geometric structures are kept unchanged to avoid forgetting. Compared with preserving features or parameters, preserving geometric structures provides flexibility to the model and makes a better balance between learning new knowledge and remembering old knowledge. Meanwhile, compared with methods that expand the network architecture, expanding geometry of the underlying space in our method avoids huge resource consumption.

2.2. Mixed-curvature Space

The mixed-curvature space has shown superior performance to the Euclidean space in various problems such as natural language processing [33], computer vision [16], and graph analysis [3]. A mixed-curvature space uses multiple constant-curvature spaces as submanifolds to model complex geometric structures. Despite its success, one challenge that remains is how to match its geometry to the intrinsic structures of data. Unsuitable geometry results in overwhelming distortions and inferior performance. Some methods identify the geometry by trial-and-error experiments [19, 42], which yet is a resource-intensive process. Recently, several methods learn to automatically identify the suitable geometry. Sun *et al.* [43] utilize the attention mechanism to assign weights for submanifolds. Wang *et al.* [46] update curvatures of the mixed-curvature space via gradient descent. Shevkunov and Prokhorenkova [40], and Zhang *et al.* [52] employ overlapping spaces and a sparse gating mechanism to select submanifolds, respectively.

Different from these methods that identify geometry for a fixed geometric structure of data, our incremental search scheme is designed for a non-stationary stream of data with growing structures. In addition, compared with most methods that use a fixed number of submanifolds and learn curvatures and weights for them, our method dynamically increases the number of constant curvature spaces, capable of matching more complex structures.

3. Preliminary

This section describes the constant curvature space and mixed-curvature space involved in our method.

Constant curvature space. A constant curvature space (CCS) is a smooth Riemannian manifold, denoted by $\mathbb{C}_K^d$ with the dimension being d and the curvature being K. The sign of K defines three types of spaces. A negative curvature defines a hyperbolic space, and we use the Poincaré ball model [10] to work for hyperbolic space. A zero curvature defines the Euclidean space. A positive curvature defines a hypersphere space. We opt for the projected sphere model [7] to work for hypersphere. For $\boldsymbol{u} \in \mathbb{C}_K^d$, its tangent space, denoted by $T_{\boldsymbol{u}}\mathbb{C}_K^d$, is a Euclidean space. We use the exponential map $\exp_{\boldsymbol{u}}^K(\cdot) : T_{\boldsymbol{u}}\mathbb{C}_K^d \to \mathbb{C}_K^d$ and the logarithmic map $\log_{\boldsymbol{u}}^K(\cdot) : \mathbb{C}_K^d \to T_{\boldsymbol{u}}\mathbb{C}_K^d$ to achieve transformations between CCS and its tangent space,

$$\begin{cases} \exp_{\boldsymbol{u}}^K(\boldsymbol{q}) = \boldsymbol{u} \oplus_K \Big(\tan_K(\sqrt{|K|}\frac{\lambda_{\boldsymbol{u}}^K \|\boldsymbol{q}\|}{2}) \frac{\boldsymbol{q}}{\sqrt{|K|} \cdot \|\boldsymbol{q}\|} \Big) \\ \log_{\boldsymbol{u}}^K(\boldsymbol{x}) = \frac{2}{\sqrt{|K|}\lambda_{\boldsymbol{u}}^K} \tan_K^{-1}(\sqrt{|K|} \cdot \| -\boldsymbol{u} \oplus_K \boldsymbol{x}\|) \frac{-\boldsymbol{u} \oplus_K \boldsymbol{x}}{\| -\boldsymbol{u} \oplus_K \boldsymbol{x}\|} \end{cases}, \tag{1}$$

where, $\oplus_K$ is the Möbius addition operation, defined as

$$\boldsymbol{x} \oplus_K \boldsymbol{y} = \frac{(1 - 2K\langle \boldsymbol{x}, \boldsymbol{y} \rangle_2 - K\|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 + K\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 - 2K\langle \boldsymbol{x}, \boldsymbol{y} \rangle_2 + K^2\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2},$$

and $\tan_K(\cdot)$ is $\tan(\cdot)$ if $K \geq 0$, otherwise $\tan_K(\cdot)$ is the hyperbolic tangent function $\tanh(\cdot)$. Similarly, $\tan_K^{-1}(\cdot) = \tan^{-1}(\cdot)$ if $K \geq 0$, otherwise $\tan_K^{-1}(\cdot) = \tanh^{-1}(\cdot)$.

A CCS is equipped with a metric $g : T_{\boldsymbol{u}}\mathbb{C}_K^d \times T_{\boldsymbol{u}}\mathbb{C}_K^d \to \mathbb{R}$ that induces the distance and angle on the space. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}_K^d$, distance between them is

$$\psi_K(\boldsymbol{x}, \boldsymbol{y}) = \frac{2}{\sqrt{|K|}} \tan_K^{-1}(\sqrt{|K|} \cdot \| -\boldsymbol{x} \oplus_K \boldsymbol{y}\|). \tag{2}$$

The angle in the CCS at the origin $\mathbf{0}$ is conformal to that in Euclidean space. For vectors $\boldsymbol{q}, \boldsymbol{s} \in T_{\mathbf{0}}\mathbb{C}_K^d$, the angle between them is

$$\cos(\angle \boldsymbol{q}\mathbf{0}\boldsymbol{s}) = \frac{g(\boldsymbol{q}, \boldsymbol{s})}{\sqrt{g(\boldsymbol{q}, \boldsymbol{q})}\sqrt{g(\boldsymbol{s}, \boldsymbol{s})}} = \frac{\langle \boldsymbol{q}, \boldsymbol{s} \rangle}{\|\boldsymbol{q}\|\|\boldsymbol{s}\|}. \tag{3}$$

Mixed-curvature space. The mixed-curvature space is defined as the Cartesian product of multiple CCSs, $\mathcal{M} \coloneqq \times_{j=1}^m \mathbb{C}_{K_j}^{d_j}$ [19], where $\mathbb{C}_{K_j}^{d_j}$ is the j-th CCS, acting as the j-th submanifold. There are m CCSs in $\mathcal{M}$ totally. Curvatures of all CCSs are denoted by $\mathcal{K} = \{K_1, \cdots, K_m\}$. Any $\boldsymbol{x} \in \mathcal{M}$ is represented by a vector concatenation, $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m)$, where $\boldsymbol{x}_j \in \mathbb{C}_{K_j}^{d_j}$. Similarly, the tangent space $T_{\boldsymbol{u}}\mathcal{M}$ of $\mathcal{M}$ is defined by the Cartesian product of tangent spaces of CCSs, $T_{\boldsymbol{u}}\mathcal{M} \coloneqq \times_{j=1}^m T_{\boldsymbol{u}_j}\mathbb{C}_{K_j}^{d_j}$, with $\boldsymbol{u} = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m) \in \mathcal{M}$ being the tangent point. A tangent vector $\boldsymbol{q} \in T_{\boldsymbol{u}}\mathcal{M}$ is $\boldsymbol{q} = (\boldsymbol{q}_1, \cdots, \boldsymbol{q}_m)$, where $\boldsymbol{q}_j \in T_{\boldsymbol{u}_j}\mathbb{C}_{K_j}^{d_j}$. The exponential map $\mathrm{Exp}_{\boldsymbol{u}}^{\mathcal{K}}(\cdot) : T_{\boldsymbol{u}}\mathcal{M} \to \mathcal{M}$ and the logarithmic map $\mathrm{Log}_{\boldsymbol{u}}^{\mathcal{K}}(\cdot) : \mathcal{M} \to T_{\boldsymbol{u}}\mathcal{M}$ in the mixed-curvature space is

$$\begin{cases} \mathrm{Exp}_{\boldsymbol{u}}^{\mathcal{K}}(\boldsymbol{q}) = \Big(\exp_{\boldsymbol{u}_1}^{K_1}(\boldsymbol{q}_1), \cdots, \exp_{\boldsymbol{u}_m}^{K_m}(\boldsymbol{q}_m) \Big) \\ \mathrm{Log}_{\boldsymbol{u}}^{\mathcal{K}}(\boldsymbol{x}) = \Big(\log_{\boldsymbol{u}_1}^{K_1}(\boldsymbol{x}_1), \cdots, \log_{\boldsymbol{u}_m}^{K_m}(\boldsymbol{x}_m) \Big) \end{cases}. \tag{4}$$

If the j-th CCS has the metric g_j, the mixed-curvature space has the metric $G(\boldsymbol{q}, \boldsymbol{s}) = \sum_{j=1}^m g_j(\boldsymbol{q}_j, \boldsymbol{s}_j)$. For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$, the squared distance between them in

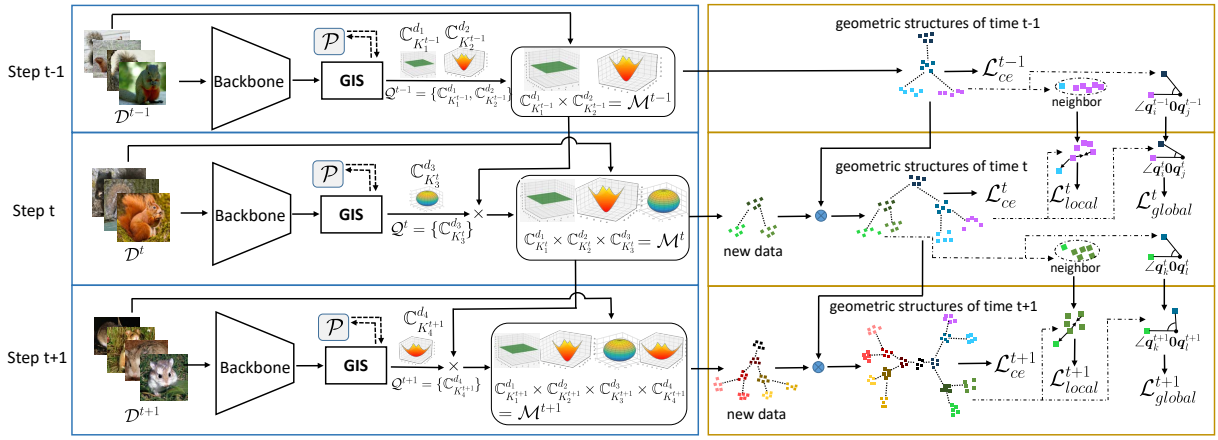

Figure 2. The framework of our method. $\mathcal{D}^t$ denotes the training data given at step t. GIS denotes the geometry incremental search scheme, and $\mathcal{P}$ denotes the submanifold pool. $\mathcal{Q}^t$ contains selected CCSs from $\mathcal{P}$ at step t, and $\mathcal{M}^t$ is the constructed mixed-curvature space, where $\times$ means the Cartesian product. We train our model under the cross-entropy loss $\mathcal{L}_{ce}^t$, and the newly designed $\mathcal{L}_{global}^t$ and $\mathcal{L}_{local}^t$ losses.

the mixed-curvature space is given by the sum of squared distances in CCSs,

$$\Psi^2(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \psi_{K_j}^2(\boldsymbol{x}_j, \boldsymbol{y}_j), \tag{5}$$

where $\psi_{K_j}^2(\cdot)$ is the squared distance in $\mathbb{C}_{K_j}^{d_j}$. For two vectors $\boldsymbol{q}, \boldsymbol{s} \in T_{\mathbf{0}}\mathcal{M}$, the angle between them is

$$\cos(\angle \boldsymbol{q}\mathbf{0}\boldsymbol{s}) = \frac{G(\boldsymbol{q},\boldsymbol{s})}{\sqrt{G(\boldsymbol{q},\boldsymbol{q})}\sqrt{G(\boldsymbol{s},\boldsymbol{s})}} = \frac{\sum_{j=1}^{m} g_j(\boldsymbol{q}_j,\boldsymbol{s}_j)}{\sqrt{\sum_{j=1}^{m} g_j(\boldsymbol{q}_j,\boldsymbol{q}_j)}\sqrt{\sum_{j=1}^{m} g_j(\boldsymbol{s}_j,\boldsymbol{s}_j)}} = \frac{\langle \boldsymbol{q},\boldsymbol{s}\rangle}{\|\boldsymbol{q}\|\|\boldsymbol{s}\|}. \tag{6}$$

4. Method

4.1. Formulation

Continual learning is cast as training a model from a stream of data $(\mathcal{D}^1, \cdots, \mathcal{D}^T)$, received at T distinct steps. $\mathcal{D}^t$ denotes the training data at step t, containing N_t pairs of the form of $(\boldsymbol{I}, y)$, where $\boldsymbol{I}$ is an instance (*e.g.*, an image) and y is the associated label from the label space $\mathcal{Y}^t$. Label spaces at different steps have no overlap, $\mathcal{Y}^1 \cap \cdots \cap \mathcal{Y}^T = \emptyset$. $\mathcal{D}^t$ is only available at step t. Our goal is to correctly classify instances from all seen classes. In other words, after the training process at step t, the model is evaluated in the label space $\cup_{i=1}^{t} \mathcal{Y}^i$.

We use a backbone $f_{\theta^t}(\cdot)$ to extract features from an instance $\boldsymbol{I}$, where θ^t denotes the parameter at step t. Then, $f_{\theta^t}(\boldsymbol{I}) \in \mathbb{R}^d$ is projected to $\boldsymbol{x}^t = (\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{m_t}^t)$ in the mixed-curvature space $\mathcal{M}^t$, where $\mathcal{M}^t = \times_{j=1}^{m_t} \mathbb{C}_{K_j^t}^{d_j}$ is produced by our geometry incremental search scheme $\text{GIS}(\mathcal{M}^{t-1}, \mathcal{D}^t)$ that expands $\mathcal{M}^{t-1}$ to get $\mathcal{M}^t$ using $\mathcal{D}^t$, and m_t is the number of CCSs.

We perform distance-based classification for $\boldsymbol{x}^t$. At step t, suppose that there are n classes totally and the classifier of the l-th class is $\boldsymbol{W}_l^t \in \mathbb{R}^d$, classifiers of all classes are denoted by $\boldsymbol{W}^t = [\boldsymbol{W}_1^t, \cdots, \boldsymbol{W}_n^t]$. $\boldsymbol{W}_l^t$ is also projected to $\boldsymbol{w}_l^t \in \mathcal{M}^t$ in the mixed-curvature space. The probability that $\boldsymbol{x}^t$ belongs to the l-th class is

$$p(l|\boldsymbol{x}^t) = \frac{\exp\big(-\Psi^2(\boldsymbol{x}^t, \boldsymbol{w}_l^t)\big)}{\sum_{l'=1}^{n} \exp\big(-\Psi^2(\boldsymbol{x}^t, \boldsymbol{w}_{l'}^t)\big)}. \tag{7}$$

We update the backbone and the classifier by minimizing the cross-entropy loss $\mathcal{L}_{ce}^t$ for classification, and the newly designed angular-regularization loss $\mathcal{L}_{global}^t$ and neighbor-robustness loss $\mathcal{L}_{local}^t$ for preventing the change of geometric structures. Our framework is shown in Fig. 2.

4.2. Geometry Incremental Search Scheme

Notation. At step t, we omit the index t of the backbone f_θ, classifier $\boldsymbol{W}$, and curvature K if they are not trained and initialized by results from step $t-1$. For example, at step t, we use θ to denote the backbone parameter that is initialized by θ^{t-1}, and θ^t means it is well-trained at step t.

Before training, we build a submanifold pool of CCSs, $\mathcal{P} = \{\mathbb{C}_{K_1}^{d_1}, \mathbb{C}_{K_2}^{d_2}, \cdots, \mathbb{C}_{K_\xi}^{d_\xi}\}$, by sampling subsets of coordinates from $f_\theta(\boldsymbol{I}) \in \mathbb{R}^d$. Concretely,

(1) Each $\mathbb{C}_{K_j}^{d_j}$ corresponds to a subset of coordinates in $f_\theta(\boldsymbol{I})$, starting from the c_j^s-th dimension to the c_j^e-th dimension in $f_\theta(\boldsymbol{I})$, and the dimension $d_j = c_j^e - c_j^s + 1$. The correspondence rule (*i.e.*, c_j^s and c_j^e) is pre-defined and does not change with t. For example, we can randomly generate them with $c_j^s < c_j^e < d$.

(2) Representation on $\mathbb{C}_{K_j}^{d_j}$ is $\boldsymbol{x}_j = \exp_{\mathbf{0}}^{K_j}(f_\theta(\boldsymbol{I})[c_j^s : c_j^e])$, where $f_\theta(\boldsymbol{I})[c_j^s : c_j^e]$ is a d_j-dimension vector.

(3) The number ξ of CCSs in $\mathcal{P}$ does not change with t.

(4) Curvatures $\mathcal{K} = \{K_1, \cdots, K_\xi\}$ change with t. At the beginning of step t, $\mathcal{K}$ is initialized by curvatures from step

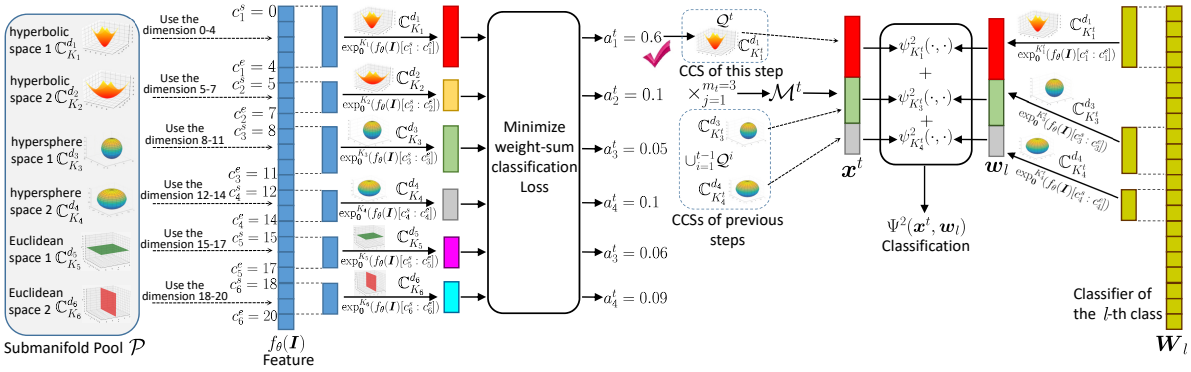

Figure 3. The illustration of the geometry incremental search. We build a submanifold pool $\mathcal{P}$ of six CCSs. We project feature $f_\theta(\boldsymbol{I}) \in \mathbb{R}^d$ to CCSs, and update weights a_j^t and curvatures K_j^t for CCSs by minimizing a weight-sum classification loss. The mixed-curvature space $\mathcal{M}^t$ is constructed by combining CCSs with high weights (contained in $\mathcal{Q}^t$) and CCSs selected at previous steps (contained in $\cup_{i=1}^{t-1}\mathcal{Q}^i$). We also project the classifier $\boldsymbol{W}_l \in \mathbb{R}^d$ to $\boldsymbol{w}_l \in \mathcal{M}^t$. The distance $\Psi^2(\boldsymbol{x}_t, \boldsymbol{w}_l)$ composed of distances $\psi^2_{K_j^t}$ in CCSs is used for classification.

$t-1$. (Curvatures are then updated via Eq. (8))

At step t, we select CCSs from $\mathcal{P}$ to expand the geometry of the underlying space, according to their importance for classifying $\mathcal{D}_t$. We project the classifier $\boldsymbol{W}_l$ of the l-th class to these CCSs based on sampled coordinates, $\boldsymbol{w}_{lj} = \exp_{\mathbf{0}}^{K_j}(\boldsymbol{W}_l[c_j^s : c_j^e])$. We assign initial weight a_j to the j-th CCS, and all weights are collectively denoted by $\mathcal{A} = \{a_1, \cdots, a_m\}$. To identify proper CCSs, we minimize the following weight-sum classification loss,

$$\mathcal{A}^t, \mathcal{K}^t = \arg\min_{\mathcal{A},\mathcal{K}} \mathbb{E}_{\boldsymbol{I}\sim\mathcal{D}} \left[-\log \frac{\exp\left(-\sum_{j=1}^m a_j \cdot \psi^2_{K_j}(\boldsymbol{x}_j, \boldsymbol{w}_{yj})\right)}{\sum_{l'=1}^n \exp\left(-\sum_{j=1}^m a_j \cdot \psi^2_{K_j}(\boldsymbol{x}_j, \boldsymbol{w}_{l'j})\right)} \right], \tag{8}$$

where y is the ground truth of $\boldsymbol{I}$, $\mathcal{A}^t = \{a_1^t, \cdots, a_m^t\}$, and $\mathcal{K}^t = \{K_1^t, \cdots, K_m^t\}$. We select CCSs from $\mathcal{P}$ according to their weights a_j^t,

$$\mathcal{Q}^t = \{\mathbb{C}^{d_j}_{K_j^t} | a_j^t > \tau_1\}, \tag{9}$$

where τ_1 is a threshold. The geometry incremental search scheme expands $\mathcal{M}^{t-1}$ to get $\mathcal{M}^t$ by combining $\mathcal{Q}^t$ with CCSs selected at previous steps,

$$\mathcal{M}^t := \times_{\mathbb{C}^d_K \in \cup_{i=1}^t \mathcal{Q}^i} \mathbb{C}^d_K, \tag{10}$$

where $|\cup_{i=1}^t \mathcal{Q}^i| = m_t$. The representation in $\mathcal{M}^t$ is $\boldsymbol{x}^t = (\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{m_t}^t)$, where $\boldsymbol{x}_j^t = \exp_{\mathbf{0}}^{K_j^t}(f_\theta(\boldsymbol{I})[c_j^s : c_j^e]) \in \mathbb{C}^{d_j}_{K_j^t}$, and $\mathbb{C}^{d_j}_{K_j^t} \in \cup_{i=1}^t \mathcal{Q}^i$. Similarly, the classifier for the l-th task is $\boldsymbol{w}_l = (\boldsymbol{w}_{l1}, \cdots, \boldsymbol{w}_{lm_t}) \in \mathcal{M}^t$, where $\boldsymbol{w}_{lj} = \exp_{\mathbf{0}}^{K_j^t}(\boldsymbol{W}_l[c_j^s : c_j^e]) \in \mathbb{C}^{d_j}_{K_j^t}$, and $\mathbb{C}^{d_j}_{K_j^t} \in \cup_{i=1}^t \mathcal{Q}^i$. Based on $\mathcal{M}^t$, we train θ and $\boldsymbol{W}$ to maximize the performance. Note that, although the dimension of $\boldsymbol{x}^t$ may grow with t, the dimension of both the feature $f_\theta(\boldsymbol{I})$ and the classifier $\boldsymbol{W}_l$ are fixed as d. The architecture and parameter number of the backbone are fixed during training, and we just add classifiers for new classes. Thus, our resource consumption does not grow much in the data stream. This process is shown in Fig. 3.

4.3. Geometric Structure Preserving

To alleviate catastrophic forgetting, we preserve captured geometric structures of old data, through which instances of the same class are close to each other and far from those of different classes. In doing so, we store a few instances of old classes in a memory buffer $\mathcal{B}$, and introduce the angular-regularization loss $\mathcal{L}_{global}$ for global structures and the neighbor-robustness loss $\mathcal{L}_{local}$ for local structures.

4.3.1 Angular regularization loss

We use angles between instances to characterize the global geometric structures of data. The idea is that if angles between old data are preserved, the geometric structures are also preserved. At step t, we measure angles of instances in the memory buffer, and changes of the angles compared with those at step $t-1$ are penalized.

Concretely, for any two instances $\boldsymbol{I}, \boldsymbol{I}' \in \mathcal{B}$, we obtain their representations $\boldsymbol{x}^t, \boldsymbol{x}'^t \in \mathcal{M}^t$. Their representations $\boldsymbol{x}^{t-1}, \boldsymbol{x}'^{t-1} \in \mathcal{M}^{t-1}$ are also computed using $f_{\theta^{t-1}}$ and $\mathcal{M}^{t-1}$. We project $\boldsymbol{x}^t, \boldsymbol{x}'^t, \boldsymbol{x}^{t-1}, \boldsymbol{x}'^{t-1}$ to tangent spaces,

$$\begin{cases} \boldsymbol{q}^t = \mathrm{Log}_{\mathbf{0}}^{\mathcal{K}^t}(\boldsymbol{x}^t) \\ \boldsymbol{q}'^t = \mathrm{Log}_{\mathbf{0}}^{\mathcal{K}^t}(\boldsymbol{x}'^t) \end{cases}, \begin{cases} \boldsymbol{q}^{t-1} = \mathrm{Log}_{\mathbf{0}}^{\mathcal{K}^{t-1}}(\boldsymbol{x}^{t-1}) \\ \boldsymbol{q}'^{t-1} = \mathrm{Log}_{\mathbf{0}}^{\mathcal{K}^{t-1}}(\boldsymbol{x}'^{t-1}) \end{cases}, \tag{11}$$

where $\boldsymbol{q}^t, \boldsymbol{q}'^t \in T_{\mathbf{0}}\mathcal{M}^t$ and $\boldsymbol{q}^{t-1}, \boldsymbol{q}'^{t-1} \in T_{\mathbf{0}}\mathcal{M}^{t-1}$. Their angles in the tangent spaces are computed by

$$\begin{aligned} \cos(\angle \boldsymbol{q}^t \mathbf{0} \boldsymbol{q}'^t) &= \frac{\langle \boldsymbol{q}^t, \boldsymbol{q}'^t \rangle}{\|\boldsymbol{q}^t\| \|\boldsymbol{q}'^t\|}, \\ \cos(\angle \boldsymbol{q}^{t-1} \mathbf{0} \boldsymbol{q}'^{t-1}) &= \frac{\langle \boldsymbol{q}^{t-1}, \boldsymbol{q}'^{t-1} \rangle}{\|\boldsymbol{q}^{t-1}\| \|\boldsymbol{q}'^{t-1}\|}, \end{aligned} \tag{12}$$

The angular-regularization loss penalizes the difference between $\angle \boldsymbol{q}^t \mathbf{0} \boldsymbol{q}'^t$ and $\angle \boldsymbol{q}^{t-1} \mathbf{0} \boldsymbol{q}'^{t-1}$ by

$$\mathcal{L}^t_{global} = \sum_{(\boldsymbol{I}_i, \boldsymbol{I}_j \in \mathcal{B})} l_\delta\big(\cos(\angle \boldsymbol{q}^t \mathbf{0} \boldsymbol{q}'^t), \cos(\angle \boldsymbol{q}^{t-1} \mathbf{0} \boldsymbol{q}'^{t-1})\big), \tag{13}$$

where l_δ is the Huber loss [34],

$$l_\delta(a, b) = \begin{cases} \frac{1}{2}(a-b)^2, & \text{if } |a-b| \leq 1, \\ |a-b| - \frac{1}{2}, & \text{otherwise} \end{cases}. \tag{14}$$

4.3.2 Neighbor-robustness loss

The neighbor-robustness loss preserves the local structures by realizing within-class compactness and between-class separability. At step t, for any $\boldsymbol{I}, \boldsymbol{I}' \in \mathcal{B}$, the squared distance $\Psi^2(\boldsymbol{x}^{t-1}, \boldsymbol{x}'^{t-1})$ between their representations $\boldsymbol{x}^{t-1}$ and $\boldsymbol{x}'^{t-1}$ are computed.

We measure neighbors of $\boldsymbol{I}$ at step $t-1$ as

$$\begin{cases} \mathcal{N}_w^{t-1}(\boldsymbol{I}) = \{\boldsymbol{I}' | \Psi^2(\boldsymbol{x}^{t-1}, \boldsymbol{x}'^{t-1}) < \tau_2 \text{ and } y = y'\} \\ \mathcal{N}_b^{t-1}(\boldsymbol{I}) = \{\boldsymbol{I}' | \Psi^2(\boldsymbol{x}^{t-1}, \boldsymbol{x}'^{t-1}) < \tau_2 \text{ and } y \neq y'\} \end{cases}, \tag{15}$$

where $\mathcal{N}_w^{t-1}(\boldsymbol{I})$ is the within-class neighbors of $\boldsymbol{I}$, $\mathcal{N}_b^{t-1}(\boldsymbol{I})$ denotes the between-class neighbors of $\boldsymbol{I}$, and τ_2 is a threshold. Next, we calculate the within-class and between-class coefficients between $\boldsymbol{I}, \boldsymbol{I}'$ as

$$w^{t-1} = \begin{cases} 1, & \text{if } \boldsymbol{I} \in \mathcal{N}_w^{t-1}(\boldsymbol{I}') \text{ or } \boldsymbol{I}' \in \mathcal{N}_w^{t-1}(\boldsymbol{I}) \\ 0, & \text{otherwise} \end{cases}, \tag{16}$$

$$b^{t-1} = \begin{cases} 1, & \text{if } \boldsymbol{I} \in \mathcal{N}_b^{t-1}(\boldsymbol{I}') \text{ or } \boldsymbol{I}' \in \mathcal{N}_b^{t-1}(\boldsymbol{I}) \\ 0, & \text{otherwise} \end{cases}. \tag{17}$$

Based on w^{t-1} and b^{t-1}, an affinity coefficient is computed by $e^{t-1} = w^{t-1} - b^{t-1}$. We summarize the above process as $e^{t-1} = E\big(\mathcal{N}_w^{t-1}(\boldsymbol{I}), \mathcal{N}_w^{t-1}(\boldsymbol{I}')\big)$.

Finally, the neighbor-robustness loss forces the squared distance $\Psi^2(\boldsymbol{x}^t, \boldsymbol{x}'^t)$ at step t to comply with the affinity coefficient at step $t-1$,

$$\mathcal{L}^t_{local} = \sum_{(\boldsymbol{I}, \boldsymbol{I}' \in \mathcal{B})} E\big(\mathcal{N}_w^{t-1}(\boldsymbol{I}), \mathcal{N}_w^{t-1}(\boldsymbol{I}')\big) \Psi^2(\boldsymbol{x}^t, \boldsymbol{x}'^t), \tag{18}$$

through which local structures become discriminative.

4.4. Training

Our goal is to train the parameter θ of the backbone f_θ and the classifier $\boldsymbol{W}$ in the stream of data. Given new data $\mathcal{D}^t$ at step t, we first produce the mixed-curvature space $\mathcal{M}^t$ via the geometry incremental search scheme. Then, we train θ and $\boldsymbol{W}$ under the guidance of a cross-entropy loss $\mathcal{L}^t_{ce}$, the angular-regularization loss $\mathcal{L}^t_{global}$, and the neighbor-robustness loss $\mathcal{L}^t_{local}$. The cross-entropy loss $\mathcal{L}^t_{ce}$ is

$$\mathcal{L}^t_{ce} = \mathbb{E}_{\boldsymbol{I} \sim (\mathcal{D}^t \cup \mathcal{B})} \left[-\log p(\widehat{y} | \boldsymbol{I}) \right]. \tag{19}$$

Algorithm 1 Training process of the proposed method.

Input: Data stream $(\mathcal{D}^1, \cdots, \mathcal{D}^T)$. Randomly initialized backbone f_θ and classifier $\boldsymbol{W}$. Memory buffer $\mathcal{B} = \emptyset$.
Output: Updated backbone f_{θ^T} and classifier $\boldsymbol{W}^T$.

```
while t < T do
    Initialize θ and W by parameters from step t − 1.
    while k < MaxIteration do
        Optimize weights A and curvatures K of CCSs via Eq. (8), using data from D^t.
    end while
    Produce the underlying space M^t for D^t via the geometry incremental search scheme via Eq. (9) and Eq. (10).
    while k < MaxIteration do
        Update the backbone f_θ and classifier W by minimizing the classification loss L^t_ce, angular-regularization loss L^t_global, and neighbor-discriminative loss L^t_local in Eq. (20).
    end while
    Obtain f_{θ^t} and W^t.
    Randomly select a few instances from D^t and add them to B.
end while
Return f_{θ^T} and W^T.
```

Overall, the loss function at step t is given by

$$\mathcal{L}^t = \mathcal{L}^t_{ce} + \lambda_1 \mathcal{L}^t_{global} + \lambda_2 \mathcal{L}^t_{local}, \tag{20}$$

where λ_1 and λ_2 are the trade-off hyperparameters. After the training process, we obtain the backbone θ^t and the classifier $\boldsymbol{W}^t = [\boldsymbol{W}^t_1, \cdots, \boldsymbol{W}^t_n]$. Then we randomly select a few instances from $\mathcal{D}^t$ and add them into $\mathcal{B}$. The pseudo-code of the training process is summarized in Algorithm 1.

5. Experiments

5.1. Settings

Datasets. We evaluate our method on the CIFAR-100 [25] and Tiny-ImageNet [26] datasets that are commonly used in continual learning. CIFAR-100 has 100 classes, and Tiny-ImageNet consists of 200 classes.

We test our method on two groups of settings following standard protocols: whether the model is pre-trained on a large number of classes [47, 54]. (1) We pre-train our model in a large number of classes. For CIFAR-100, we test our method on three settings: C100-B50-S5, C100-B50-S10, and C100-B40-S20. For Tiny-ImageNet, we test our method on three settings: T200-B100-S5, T200-B100-S10, and T200-B100-S20. For instance, C100-B50-S5 means that we first pre-train the model using the first 50 classes, and the following classes are split into 5 steps that each has 10 classes. (2) We do not pre-train our model. We use the CIFAR-100 dataset and test our method on three settings: C100-B0-S5, C100-B0-S10, C100-B0-S20. The second group of settings are more challenging.

Method	C100-B50-S5	C100-B50-S10	C100-B40-S20
iCaRLCNN [36]	40.19	38.87	34.26
iCaRLNME [36]	49.14	45.31	40.53
EEIL [11]	50.21	47.60	42.23
LUCIR [21]	54.71	50.53	48.00
PASS [54]	55.67	49.03	48.48
IL2A [53]	54.98	45.07	45.74
HFA [18]	55.55	52.11	46.12
Ours	**56.03**	**54.31**	**49.32**

Table 1. Final accuracy (%) on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings.

Method	T200-B100-S5	T200-B100-S10	T200-B100-S20
iCaRLCNN [36]	23.17	20.71	20.28
iCaRLNME [36]	34.43	33.24	27.51
EEIL [11]	35.00	33.67	27.64
IL2A [53]	36.58	34.28	28.34
HFA [18]	36.11	33.65	31.37
Ours	**38.10**	**37.99**	**34.85**

Table 2. Final accuracy (%) on the T200-B100-S5, T200-B100-S10, and T200-B100-S20 settings.

Experimental Details. ResNet-18 [20] is used as the backbone and trained from scratch in our experiments. In the training process, we set the batchsize as 64 and use the Adam optimizer with 0.001 initial learning rate. We train the model for 30 epochs for each step. We set the threshold τ_1 as $\frac{1}{n}$ for the geometry incremental search scheme, where n is the number of classes. For the threshold τ_2, we set it as the mean of squared distances of instances with the same class. The size of the memory buffer is the same as existing replay-based methods [11, 21, 36]. For the first group of settings, we randomly select 20 instances for each class. For the second group of settings, we evaluate our method with the size of the memory buffer $\mathcal{B}$ as 200 and 500.

In the submanifold pool $\mathcal{P}$, CCSs are sampled with size 16, 32, 64, 128, and 256. For simplicity, we sequentially sample coordinates from features for CCSs with the same size. Take CCSs with a dimension of 16 as an example, the first CCS uses dimensions $1-16$, and the second one uses dimensions $17-32$. Since the dimension of features from ResNet-18 is 512, there are $\frac{512}{16}+\frac{512}{32}+\frac{512}{64}+\frac{512}{128}+\frac{512}{256}=62$ CCSs in $\mathcal{P}$ totally. We initialize curvatures of half of CCSs as -1, and initialize curvatures of the rest of CCSs as 1. The initial weight a_j is assigned as $\frac{1}{n}$.

Evaluation Metric. We use the final accuracy, average accuracy, average forgetting, and average incremental accuracy [11, 53] to evaluate our method.

5.2. Main Results

Final accuracies on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings are shown in Tab. 1. Final accuracies on the T200-B100-S5, T200-B100-S10, and T200-B100-S20 settings are shown in Tab. 2. Final accuracies on the C100-B0-S5, C100-B0-S10, and C100-B0-S20 settings are shown in Tab. 3. Average accuracies and average forgetting on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings are shown in Tab. 4. Average forgetting on the C100-B0-S20 setting is shown in Fig. 4. Accuracy curves on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings are shown in Fig. 5. Our method achieves better performance than compared continual learning methods, having higher accuracies and lower forgetting. These results demonstrate the effectiveness of our method.

Take C100-B50-S10 as an example (in Tab. 1), our method finally achieves 54.12%, 3.59% and 9.05% higher than the replay-based methods LUCIR [21] and IL2A [53], and 8.81% higher than the regularization-based method iCaRL [36]. This shows the effectiveness of exploring the geometric structure for continual learning. Moreover, our performance is 2.01% higher than that of HFA [18] that uses a fixed hyperbolic geometry for the data stream. In contrast, our method expands the geometry of the underlying space to match the growing geometric structures of the data stream, leading to better performance.

Size of $\mathcal{B}$	Method	C100-B0-S5	C100-B0-S10	C100-B0-S20
200	ER [37]	21.94 ± 0.83	14.23 ± 0.12	9.90 ± 1.67
	GEM [30]	19.73 ± 0.34	13.20 ± 0.21	8.29 ± 0.18
	AGEM [13]	17.97 ± 0.26	9.44 ± 0.29	4.88 ± 0.09
	iCaRL [36]	30.12 ± 2.45	22.38 ± 2.79	12.62 ± 1.43
	FDR [5]	22.84 ± 1.49	14.85 ± 2.76	6.70 ± 0.79
	GSS [1]	19.44 ± 2.83	11.84 ± 1.46	6.42 ± 1.24
	DER++ [8]	27.46 ± 1.16	21.76 ± 0.78	15.16 ± 1.53
	HAL [12]	13.21 ± 1.24	9.67 ± 1.67	5.67 ± 0.91
	ERT [9]	21.61 ± 0.87	12.91 ± 1.46	10.14 ± 1.96
	RM [4]	32.23 ± 1.09	22.71 ± 0.93	15.15 ± 2.14
	Ours	$\mathbf{32.75 \pm 0.42}$	$\mathbf{25.87 \pm 1.01}$	$\mathbf{19.09 \pm 0.34}$
500	ER [37]	27.97 ± 0.33	21.54 ± 0.29	15.36 ± 1.15
	GEM [30]	25.44 ± 0.72	18.48 ± 1.34	12.58 ± 2.15
	AGEM [13]	18.75 ± 0.51	9.72 ± 0.22	5.97 ± 1.13
	iCaRL [36]	35.95 ± 2.16	30.25 ± 1.86	20.05 ± 1.33
	FDR [5]	29.99 ± 2.23	22.81 ± 2.81	13.10 ± 3.34
	GSS [1]	22.08 ± 3.51	13.72 ± 2.64	7.49 ± 4.78
	DER++ [8]	38.39 ± 1.57	36.15 ± 1.10	21.65 ± 1.44
	HAL [12]	16.74 ± 3.51	11.12 ± 3.80	9.71 ± 2.91
	ERT [9]	28.82 ± 1.83	23.00 ± 0.58	18.42 ± 1.92
	RM [4]	39.47 ± 1.26	32.52 ± 1.53	23.09 ± 1.72
	Ours	$\mathbf{40.85 \pm 0.91}$	$\mathbf{33.55 \pm 0.53}$	$\mathbf{28.78 \pm 0.87}$

Table 3. Final accuracy (%) on the C100-B0-S5, C100-B0-S10, and C100-B0-S20 settings.

Method	C100-B50-S5		C100-B50-S10		C100-B40-S20	
	AIA↑ (%)	AF↓ (%)	AIA↑ (%)	AF↓ (%)	AIA↑ (%)	AF↓ (%)
iCaRLCNN [36]	51.31	42.13	48.28	45.69	44.61	43.54
iCaRLNME [36]	60.86	24.90	53.86	28.32	51.01	35.53
EEIL [11]	60.40	23.36	55.77	26.65	52.57	32.40
LUCIR [21]	63.75	21.00	60.68	25.12	58.27	28.65
PASS [54]	63.84	25.20	59.87	30.25	58.07	30.61
IL2A [53]	66.19	29.57	58.2	39.75	58.01	48.66
HFA [18]	66.96	23.54	63.7	25.45	59.02	29.76
Ours	**67.44**	**21.94**	**65.93**	**23.76**	**61.85**	**24.71**

Table 4. Average incremental accuracy (denoted by 'AIA') and average forgetting (denoted by 'AF') on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings.

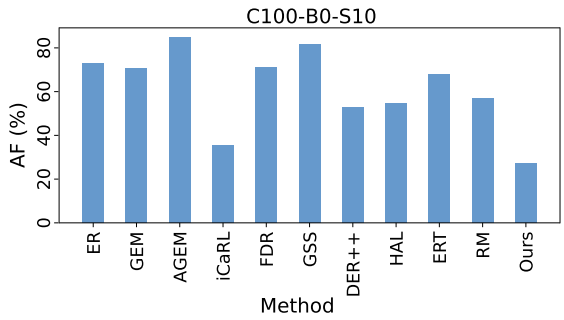

Figure 4. Average forgetting (AF, %) on the C100-B0-S10 setting.

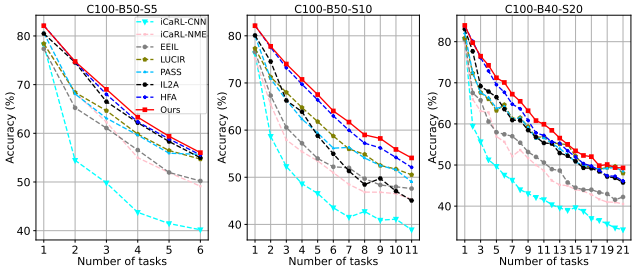

Figure 5. Accuracy curves on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings.

In the second group of settings (in Tab. 3), our method achieves good performance again on both the two sizes of the memory buffer. With the steps in continual learning increasing, learning a good model becomes more and more challenging. In this case, our improvements become larger and larger with the increase of steps compared with existing methods. For example, when the size of the memory buffer is 200, compared with RM [4], our improvements are 0.52%, 3.16%, and 3.94% on the C100-B0-S5, C100-B0-S10, C100-B0-S20 settings, respectively. This shows that exploring date geometry benefits to continual learning, efficiently alleviating the catastrophic forgetting problem, especially in challenging settings.

5.3. Ablation Study

Our method includes three important components: the geometry incremental search scheme, the angular-regularization loss function, and the neighbor-robustness loss function. We conduct ablation experiments on the CIFAR-100 dataset to evaluate their effectiveness. We denote the geometry incremental search scheme, angular-regularization loss function, and the neighbor-robustness loss function as 'GIS', 'Global', and 'Local', respectively. We first evaluate using the Euclidean geometry, that is, none of the three components is used, and the model is only trained by the cross-entropy loss in the Euclidean space using the memory buffer. Then, we add the geometry incremental search scheme to the model. Next, we remove the geometry incremental search scheme and add the two proposed loss functions. Finally, we evaluate the two loss functions by combining the geometry incremental search scheme with one of them. Results are shown in Tab. 5.

Method	C100-B50-S5	C100-B50-S10	C100-B40-S20
Euclidean	50.63	49.56	43.60
GIS	51.62	49.86	46.15
Global+Local	52.63	50.50	44.45
GIS+Global	52.49	50.33	47.31
GIS+Local	53.94	53.40	46.84
Ours	**56.03**	**54.31**	**49.32**

Table 5. Final accuracy (%) of ablation results on the C100-B50-S5, C100-B50-S10, and C100-B40-S20 settings.

We observe that all the three components have contributions to our improvement. Comparing 'GIS' with 'Baseline' and 'Ours' and 'Global+Local', the geometry incremental search scheme brings about 3% improvements on the C100-B50-S5 and C100-B40-S20 settings. 'Ours' has better performance than 'GIS+Local', and 'GIS+Global' has better performance than 'GIS', showing the angular-regularization loss function helps solve the catastrophic forgetting problem in continual learning. Similarly, the neighbor-robustness loss function also benefits to continual learning, especially in C100-B50-S10. These results show that exploring data geometry via the designed components is an effective and promising way for continual learning.

6. Conclusion

In this paper, we have presented that exploring data geometry benefits to continual learning, through which the forgetting issue of old data can be prevented by preserving geometric structures of data and learning from new data benefits from expanding the geometry of the underlying space. The introduced geometry incremental search scheme can produce suitable geometry for new data, avoiding data distortions. The proposed angular-regularization loss function can well preserve global structures of old data, and the neighbor-robustness loss can make local structures more discriminative. Experiments on multiple settings show that our method achieves better performance than continual learning methods designed in Euclidean space. In the current method, the submanifold pool has a finite number of CCSs, which may limit the ability of the mixed-curvature space. In the future, we are going to study more flexible schemes, where the submanifold pool has an infinite number of CCSs with arbitrary dimensions.

Acknowledgements. This work was supported by the Natural Science Foundation of China (NSFC) under Grants No. 62172041 and No. 62176021, and Shenzhen National Science Foundation (Shenzhen Stable Support Fund for College Researchers), No. 20200829143245001.

References

[1] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11816–11825, 2019. 7

[2] Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. Hyperbolic busemann learning with ideal prototypes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 103–115, 2021. 1

[3] G. Bachmann, Gary Bécigneul, and Octavian-Eugen Ganea. Constant curvature graph convolutional networks. In *International Conference on Machine Learning (ICML)*, pages 486–496, 2020. 1, 3

[4] Jihwan Bang, Heesu Kim, Young Joon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8214–8223, 2021. 7, 8

[5] Ari S. Benjamin, David Rolnick, and Konrad P. Körding. Measuring and regularizing networks in function space. In *International Conference on Learning Representations (ICLR)*, 2019. 7

[6] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017. 1

[7] Samuel R Buss and Jay P Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics (TOG)*, 20(2):95–126, 2001. 3

[8] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15920–15930, 2020. 2, 7

[9] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. *International Conference on Pattern Recognition (ICPR)*, pages 2180–2187, 2021. 7

[10] James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31(59-115):2, 1997. 3

[11] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European Conference on Computer Vision (ECCV)*, pages 241–257, 2018. 7

[12] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip H. S. Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. 1, 2, 7

[13] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 7

[14] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Tafazzoli Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2534–2543, 2021. 1

[15] Pengfei Fang, Mehrtash Harandi, and Lars Petersson. Kernel methods in hyperbolic spaces. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10645–10654, 2021. 1

[16] Zhi Gao, Yuwei Wu, Mehrtash Harandi, and Yunde Jia. Curvature-adaptive meta-learning for fast adaptation to manifold data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 45(2):1545–1562, 2023. 2, 3

[17] Zhi Gao, Yuwei Wu, Yunde Jia, and Mehrtash Harandi. Curvature generation in curved spaces for few-shot learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8671–8680, 2021. 1

[18] Zhi Gao, Yuwei Wu, Yunde Jia, and Mehrtash Harandi. Hyperbolic feature augmentation via distribution estimation and infinite sampling on manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 7

[19] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 3

[20] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 7

[21] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 831–839, 2019. 7

[22] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6418–6428, 2020. 1

[23] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 114:3521 – 3526, 2017. 1, 2

[24] Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. In *European Conference on Computer Vision (ECCV)*, pages 219–236, 2022. 2

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 6

[26] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 6

[27] Shen Li, Jianqing Xu, Xiaqing Xu, Pengcheng Shen, Shaoxin Li, and Bryan Hooi. Spherical confidence learning for face recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15629–15637, 2021. 1

[28] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 40:2935–2947, 2018. 2

[29] Teng Long, Pascal Mettes, Heng Tao Shen, and Cees GM Snoek. Searching for actions on the hyperbole. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1150, 2020. 1

[30] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6467–6476, 2017. 2, 7

[31] Aaron Lou, Maximilian Nickel, Mustafa Mukadam, and Brandon Amos. Learning complex geometric structures from data with deep riemannian manifolds. 2021. 1

[32] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7765–7773, 2018. 1, 2

[33] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems (NeurIPS)*, 2017. 3

[34] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3967–3976, 2019. 6

[35] Guodong Qi, Huimin Yu, Zhaohui Lu, and Shuzhao Li. Transductive few-shot classification on the oblique manifold. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8392–8402, 2021. 1

[36] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, G. Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017. 2, 7

[37] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations (ICLR)*, 2019. 7

[38] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016. 3

[39] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018. 2

[40] Kirill Shevkunov and Liudmila Prokhorenkova. Overlapping spaces for compact graph representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11665–11677, 2021. 1, 2, 3

[41] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2990–2999, 2017. 2

[42] Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2020. 2, 3

[43] Li Sun, Zhongbao Zhang, Junda Ye, Hao Peng, Jiawei Zhang, Sen Su, and Philip S. Yu. A self-supervised mixed-curvature graph neural network. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 2, 3

[44] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, and Yihong Gong. Bi-objective continual learning: Learning 'new' while consolidating 'known'. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 5989–5996, 2020. 2

[45] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 184–193, 2021. 2

[46] Shen Wang, Xiaokai Wei, Cícero Nogueira dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew O. Arnold, Bing Xiang, Philip S. Yu, and Isabel F. Cruz. Mixed-curvature multi-relational graph neural network for knowledge graph completion. In *International Conference of World Wide Web (WWW)*, pages 1761–1771, 2021. 1, 2, 3

[47] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 171–181, 2022. 6

[48] Zhirong Xu, Shiyang Wen, Junshan Wang, Guojun Liu, Liang Wang, Zhi Yang, Lei Ding, Yan Zhang, Di Zhang, Jian Xu, and Bo Zheng. AMCAD: adaptive mixed-curvature representation based advertisement retrieval system. In *IEEE International Conference on Data Engineering (ICDE)*, pages 3439–3452, 2022. 2

[49] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3013–3022, 2021. 1, 3

[50] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6980–6989, 2020. 2

[51] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, pages 3987–3995, 2017. 2

[52] Shuai Zhang, Yi Tay, Wenqi Jiang, Da-cheng Juan, and Ce Zhang. Switch spaces: Learning product spaces with sparse gating. *arXiv preprint arXiv:2102.08688*, 2021. 2, 3

[53] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. Class-incremental learning via dual augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14306–14318, 2021. 1, 2, 7

[54] Fei Zhu, Xu-Yao Zhang, Chuan Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5867–5876, 2021. 2, 6, 7