Expanding Hyperspherical Space for Few-Shot Class-Incremental Learning

Yao Deng and Xiang Xiang*
Key Lab of Image Processing and Intelligent Control, Ministry of Education
School of Artificial Intelligence and Automation
Huazhong University of Science and Technology, Wuhan 430074, China

Abstract

In today's ever-changing world, the ability of machine learning models to continually learn new data without forgetting previous knowledge is of utmost importance. However, in the scenario of few-shot class-incremental learning (FSCIL), where models have limited access to new instances, this task becomes even more challenging. Current methods use prototypes as a replacement for classifiers, where the cosine similarity of instances to these prototypes is used for prediction. However, we have identified that the embedding space created by using the relu activation function is incomplete and crowded for future classes. To address this issue, we propose the Expanding Hyperspherical Space (EHS) method for FSCIL. In EHS, we utilize an odd-symmetric activation function to ensure the completeness and symmetry of embedding space. Additionally, we specify a region for base classes and reserve space for unseen future classes, which increases the distance between class distributions. Pseudo instances are also used to enable the model to anticipate possible upcoming samples. During inference, we provide rectification to the confidence to prevent bias towards base classes. We conducted experiments on benchmark datasets such as CIFAR100 and mini-ImageNet, which demonstrate that our proposed method achieves state-of-the-art performance.

1. Introduction

In recent years, deep convolutional neural networks (CNNs) have made significant advancements in various vision tasks [5,9,21,25]. However, these methods heavily rely on large-scale supervised datasets to train models capable of learning a limited number of object classes. In real-world scenarios, data is received in a continuous stream [8], with new classes of data emerging regularly [4]. Humans possess the ability to continually learn new knowledge while retaining past insights. Similarly, deep learning models must incrementally expand their knowledge and discriminate new classes, known as Class-Incremental Learning (CIL). The primary challenge of CIL is catastrophic forgetting [7], where the performance of the model on old classes decreases significantly when updated with new classes. Addressing catastrophic forgetting has become the main focus of deep learning researchers [11, 13, 31, 33, 38].

In addition, collecting and labeling data can be an expensive process, resulting in models that may only be trained on limited new data. This makes the task more challenging, particularly when addressing few-shot inputs in the incremental learning scenario known as Few-Shot Class-Incremental Learning (FSCIL), as illustrated in Figure 1.

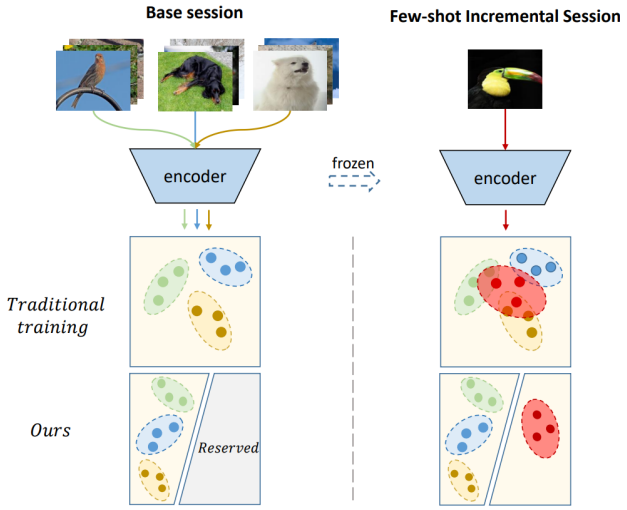


Figure 1. FSCIL setting and our scheme compared with traditional training. A model is needed to discriminate all classes, while session with non-overlapping classes arrive in sequence. Adequate data are available when training in base session, while instances of each class are insufficient in the following incremental session. The model should incorporate new classes without forgetting old ones. In contrast to the traditional training approach, our method separates the feature space into two distinct hyperspheres, namely the upper and lower hemispheres for base and future classes respectively, and utilizes cosine similarity to distinguish between them.

*Corresponding author (E-mail: xex@hust.edu.cn).

In this scenario, a model must incorporate new classes with only a limited number of instances while retaining previous knowledge, without experiencing catastrophic forgetting. Furthermore, it must also address the problem of overfitting when dealing with few-shot instances.

Several approaches [22, 37, 42] have been proposed to tackle the FSCIL challenge from a few-shot learning perspective, with the aim of mitigating overfitting during model updating. Among them, ProtoNet [22] has introduced the concept of prototypes, which can somewhat alleviate catastrophic forgetting in the case of limited new data. In ProtoNet, the feature extractor is frozen after training on the base session, and the average embedding of each class is obtained to replace the classifier as a prototype. The model discriminates classes based on the similarity, typically cosine similarity, between the embedding of the instances and the prototypes. This approach can maintain a better performance on instances from old classes as only the classifier is updated in the incremental session, and the prototypes of the base session remain unchanged. However, since the extractor is not trained on data from new classes, the embeddings of new classes may be mixed with old ones, resulting in poor discriminability of new classes. Additionally, the overlapping areas of the distributions of the old and new classes not only cause the performance of the model on the new class to degrade, but also the performance of the base session.

To address the limitations of existing methods, recent advances [18, 39] in the field of FSCIL have sought to design more sparse embedding spaces. Building upon this research, we have endeavored to further increase the sparsity of the embedding space, which necessitates a more extensive feature space. Our analysis and experiments indicate that when employing the commonly used activation function *relu* and cosine similarity [29], the model is only capable of mapping features to a very limited region of the hyperspherical space. Specifically, the embedding space is confined to a small region of the complete hypersphere due to the non-negativity constraints of the *relu* activation function. Since the feature space of a model that requires continuous updating is already too narrow to occupy the entire hypersphere, the space becomes even more crowded as new classes are added. Achieving embedding distribution across the complete hypersphere could significantly improve the sparsity of the feature space.

Besides, in incremental learning, the number of classes keeps increasing with the arrival of new tasks, which poses a challenge for the prototype-based approach since the embeddings of base classes will mix with those of new classes. This mixing can negatively impact the model's performance on both old and new classes. To address this issue, it is important to limit the space allocated to base classes and reserve enough space for future classes. However, linear classifiers cannot allocate space for base and novel classes separately in a general feature space. To overcome this problem, we propose to use cosine similarity and embed the features in a hypersphere rather than a general space. This hypersphere has a specific shape that allows us to allocate space for base and novel classes separately. We aim to reserve as much space for future classes as for the base features, which can be allocated as upper and lower hemispheres, respectively. To achieve a sparse embedding representation, we use regularization techniques to split the base and unseen classes and specify the space occupied by each.

The present study introduces the Expanding Hyperspherical Space (EHS) for (FSCIL) to address the problem of limited space in the embedding representation of the model. In order to achieve this, we employ an odd-symmetric activation function to ensure that the output of the feature extractor has both positive and negative values, and we use a base vector to partition the embedding space into upper and lower hemispheres. We also employ some regulations in the loss function to allocate embeddings of base classes to the upper hemisphere and pseudo-instances to the lower hemisphere to enable the model to anticipate future classes. During inference, we rectify the samples in the lower hemisphere, reducing their confidence in being predicted as base classes. Our experiments demonstrate that the proposed method outperforms the previous state-of-the-art (SOTA) methods on benchmark datasets.

2. Related Works

2.1. Few-Shot Learning

In order to learn with inadequate data as humans do, Few-shot Learning (FSL) methods [2, 3, 6, 14, 16, 22, 24, 28, 34–36] can be classified into two categories: model optimization and metric-based methods. Optimization-based methods [2, 3, 6, 16] aim to facilitate rapid adaptation of models to few-shot data, as traditional gradient descent techniques are challenging to apply in this context. For example, model-agnostic meta-learning [6] proposes increasing the gradient update steps to improve the model's fast adaptation capability. In contrast, metric-based methods [14, 22, 24, 28, 34–36] focus on the distance metric between novel query samples and the base knowledge representations. ProtoNet [22] introduced the notion of a prototype, which replaces the classifier with the average embedding of each class. The distance between prototypes and instance features is used to determine the class to which a sample belongs. Similarly, FRN [30] utilizes feature maps reconstruction to achieve few-shot learning.

2.2. Class-Incremental Learning

Class-incremental Learning (CIL) is an approach to learn new classes without forgetting previously learned ones from

a continuous stream of data. CIL methods [1, 11, 13, 27, 33, 40] can be broadly categorized into two groups: replay and regularization. Replay methods rely on replaying past experiences to alleviate forgetting. For instance, DER [33] uses dark knowledge to distill past experiences sampled throughout the training process. GCR [27] prioritizes the most significant gradient changes between sessions and replays them in a specific sequence. On the other hand, regularization-based methods build a regularized feature space for both old and new classes. LwF [13] uses distillation loss of the new model output and fine-tuning to train the model on new tasks. EWC [11] mitigates catastrophic forgetting by finding a common solution between two adjacent sessions through L_2 loss applied to the model weights.

2.3. Few-Shot Class-Incremental Learning

Few-shot Class-incremental Learning (FSCIL) is a newly proposed learning scenario that focuses on handling incremental learning with limited samples. To address this challenge, researchers have proposed various methods [15, 18, 26, 32, 37, 39]. Tao *et al*. [26] proposed a neural gas network that preserves the topology of features from different classes to minimize the forgetting problem. Mazumder *et al*. [15] adopt a freezing mechanism to prevent catastrophic forgetting and reserve some model parameters for learning future classes to mitigate overfitting. CEC [37] applies a graph model to the classifiers to balance the relationship between the weights of old and new classes. FACT [39] uses multiple virtual prototypes to reserve space for future classes in the embedding space. ALICE [18] adopts a margin trick to learn a sparse feature representation for future classes.

3. Prototypical Networks in FSCIL

In this section, we begin by presenting the FSCIL scenario, and subsequently introduce the baseline of prototype-based techniques along with its drawbacks.

3.1. Few-Shot Class-Incremental Learning

FSCIL combines the settings of few-shot learning and class-incremental learning. In the base session, the model receives training set $\mathcal{D}^0 = \{(\mathbf{x_i}, y_i)\}_{i=0}^{|\mathcal{D}^0|}$ with *sufficient* training samples $\mathbf{x_i} \in \mathbb{R}^D$ of base classes $y_i \in Y_0$. Y_0 is the label space of base session. Then in the incremental sessions, the model receive a sequence of *insufficient* training sets $\{\mathcal{D}^1, \mathcal{D}^2, ...\mathcal{D}^T\}$, where $\mathcal{D}^t = \{(\mathbf{x_i}, y_i)\}_{i=0}^{|\mathcal{D}^t|=NK}$. $y_i \in Y_t$ where Y_t is the label space of session t and these label spaces do not contain any overlap. $Y_i \cap Y_j = \varnothing$ for $i \neq j$ and model only access to $\mathcal{D}^t$ in session t. Each incremental session can be defined as a N-way K-shot few shot leaning problem to reflect the insufficiency of the instances. At inference, the trained model on the current task t must classify test samples of current and old tasks *i.e.*, $\{0, ...t-1\}$. A model should learn new classes and maintain the ability to classify old classes when facing a new session t, *i.e.*, minimize the empirical risk over all testing set:

$$\sum_{(\mathbf{x_i}, y_i) \in \mathcal{D}^0 \cup ... \mathcal{D}^t} \ell(\mathcal{M}(\mathbf{x_i}), y_i), \tag{1}$$

where ℓ is a loss function that measures the discrepancy between prediction of model $\mathcal{M}$ and ground-truth label. The prediction is calculated from the similarity between embedding and weights of linear classifier: $\mathcal{M}(\mathbf{x}) = W^T f(\mathbf{x})$, where $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ and $W \in \mathbb{R}^{d \times |Y_0 \cup ... Y_t|}$. W can be divided as: $W = \{w_0, ...w_{|Y_0 \cup ... Y_t|}\}$, where w_c is the classifier of class c

3.2. Prototypical Network

As a FSCIL method, ProtoNet [22] trains model with cross-entropy loss in the base session. In the incremental session, the embedding extractor f is frozen and used to extract the average embedding of each class, which is called prototype:

$$p_c = \frac{1}{N_c} \sum_{\{(\mathbf{x_i}, y_i) \in \mathcal{D}^t | y_i = c\}} f(\mathbf{x_i}). \tag{2}$$

N_c is the instance number of class c. Each prototype can be viewed as the common embedding of each class and utilized to replace classifier weight: $w_c = p_c$. Then the similarities between samples and prototypes are denoted as the prediction: $P^T f(\mathbf{x})$, where $P = \{p_0, ...p_{|Y_0 \cup ... Y_t|}\}$. Instead of linear classifier, some works use cosine similarity as model prediction:

$$\mathcal{M}(\mathbf{x}) = \frac{P^T}{||P||} \frac{f(\mathbf{x})}{||f(\mathbf{x})||}. \tag{3}$$

It makes model focus on angles between normalized embeddings, which distributed on the unit hypersphere of d dimension.

3.3. Ignorance of Activation Function

It should be noted that embeddings do not necessarily distribute uniformly across the entire unit hypersphere. The popular activation function *relu* is advantageous and widely used in many models. However, its output is always non-negative, which causes the elements of embeddings to be non-negative as well. This limitation means that the embeddings can only occupy a very small portion of the entire hypersphere space. For embedding vectors with d dimensions, they are only able to utilize $\frac{1}{2^d}$ of the entire space. A visual representation of this phenomenon is provided in Fig. 2 for the two-dimensional case.

The clustering of features in a limited portion of the hypersphere poses challenges for prototype-based methods.

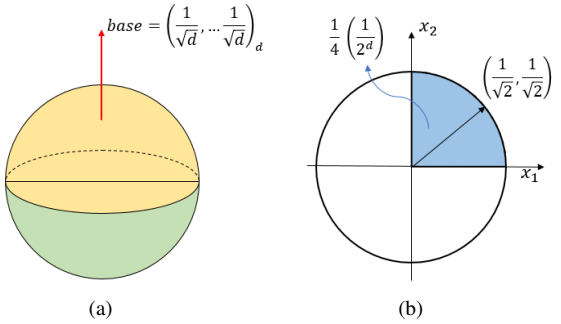


Figure 2. Hypersphere space with base vector (a) and case in two-dimension (b). Through the cosine similarity between instances and base vector, the embedding space can be divide into upper (yellow) and lower (green) hemispheres. (b) shows case in two-dimension that only part of hypersphere is utilized when using *relu* as activation function.

This is because when the variance of the embedding distribution is large for certain classes and the distance between prototypes is close, instances may be incorrectly classified. Moreover, in incremental sessions, the prototypes of newly extracted classes tend to cluster near those of the base session, resulting in overlapping embedding distributions of base and new classes, and leading to confusion in classifying samples in the overlapping region. Utilizing the unoccupied feature space can help overcome these issues by achieving sparser distribution of prototypes, and increasing the distance between prototypes of different classes and between old and new classes.

4. Expanding Hyperspherical Space for FSCIL

To address the issue of mutual overlap between the distributions of each class and to maximize the cosine similarity between the samples and their corresponding prototypes, we want to ensure the sparsity of the feature space. This is achieved through the use of an odd-symmetric activation function which maintains the integrity and homogeneity of the hypersphere. During the training of the base session, regularization is utilized to distinguish between the base and unseen classes, and to allocate space for each, with a margin reserved between them. During inference, we apply rectification to those samples that are distributed in the reserved space for instances in incremental sessions, which reduces the model's confidence in predicting them as belonging to the base classes.

We first introduce the selection of the activation function, and then discuss how to partition the embedding space and the rectification method at inference.

4.1. Odd-Symmetric Activation Function

The activation function *relu* is frequently used in various models, but its output is always non-negative. Consequently, the feature vectors produced by the extractor are restricted to a very limited portion of the hypersphere, representing only a fraction of the available feature space ($\frac{1}{2^d}$). This issue becomes more pronounced as the feature dimension d increases, and a smaller proportion of the features can be distributed throughout the hypersphere.

In order to increase the distance between the individual prototypes and expand the space occupied by embeddings to the entire hypersphere, we adopt the odd-symmetric activation function *tanh*. Unlike *relu*, which only outputs non-negative values, the output of *tanh* is distributed symmetrically on both sides of the origin, with a value range of $[-1, 1]$. Therefore, by using *tanh* as the activation function, embeddings can be distributed over the entire hypersphere, thereby improving the sparsity of the feature space.

4.2. Specifying for Base Classes

To optimize the distribution of embeddings and achieve a clear separation between the features of base classes and future classes, it is necessary to adjust the embedding distribution within the entire hypersphere. This will ensure a greater distance between the prototypes of different classes, minimizing the overlap between their distributions. Specifically, we aim to distribute the embeddings of base classes in the upper hemisphere and those of future classes in the lower hemisphere. To achieve this, we set a base vector, denoted by $base = (\frac{1}{\sqrt{d}}, \ldots \frac{1}{\sqrt{d}})_d$, as a reference vector to differentiate between the upper and lower hemispheres, shown in Fig. 2. The similarity between a feature and the base vector is then calculated as:

$$sim_B(\mathbf{x}) = base^T \frac{f(\mathbf{x})}{|f(\mathbf{x})|}. \tag{4}$$

The similarity between the feature and the base vector is used to determine whether the feature belongs to the upper or lower hemisphere. A positive similarity indicates the upper hemisphere and a negative similarity indicates the lower hemisphere. To enforce the distribution of base classes in the upper hemisphere, a regularization is used, which is formulated as follows:

$$\mathcal{L}_B = max(m_B - sim_B(\mathbf{x}), 0), \tag{5}$$

where m_b denotes the margin from the demarcation line of upper and lower hemisphere. If $sim_B > m_B$, the feature distribution is on the upper hemisphere side of margin, and the loss is 0. When $sim_B < m_B$, the smaller the similarity to the base vector, the further the feature is from the upper hemisphere, and the greater the loss.

4.3. Reserving for Future Classes

Our objective is to equip the model with the capability to anticipate future instances and effectively map them to the lower hemisphere of the feature space. In order to achieve

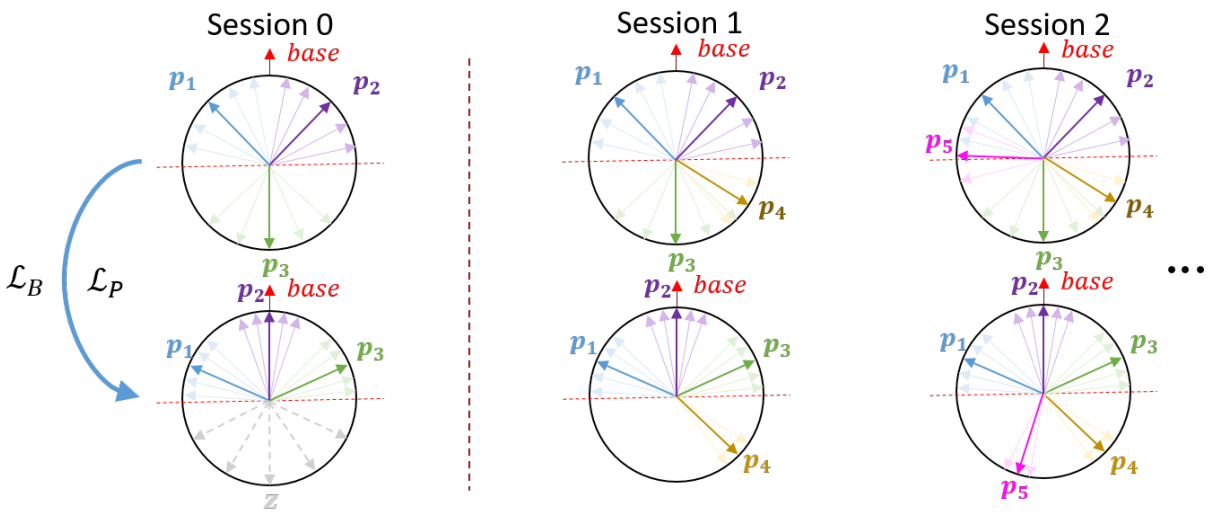


Figure 3. An illustration of feature distributions of simply cross-entropy loss trained model and the loss with two regularization terms added. The light color arrows represent examples of different class features on the hyperspherical feature space. The dark color arrows represent the average feature prototype of corresponding classes, with mixed up pseudo instances. Our two regularization terms reserve space for future classes in base session.

this, we utilize the generation of pseudo-samples, which can mimic the characteristics of the instances that the model may encounter in the future.

Motivated by [39], we aim to fuse the features of two samples to generate new features, which can be used as features for a hypothetical sample that might appear in the future. In order to accomplish this, we divide the extractor into two components, denoted as $f(\mathbf{x}) = g(h(\mathbf{x}))$. To combine the features of two different input samples belonging to different classes, we merge the embeddings of the intermediate layer of the extractor as follows:

$$\mathbf{z} = g[\gamma h(\mathbf{x_i}) + (1-\gamma)h(\mathbf{x_j})], \tag{6}$$

where $y_i \neq y_j$ and $\gamma \in [0,1]$ is sampled from Beta distribution. Similarly, we calculate the similarity between pseudo-sample and the negative base vector:

$$sim_P(\mathbf{z}) = -base^T \frac{\mathbf{z}}{|\mathbf{z}|}. \tag{7}$$

Then loss that constrains the distribution of the future classes in the lower hemisphere is:

$$\mathcal{L}_P = max(m_P - sim_P(\mathbf{z}), 0), \tag{8}$$

where m_P indicates the margin on the lower side of the upper and lower hemisphere dividing line. By generating virtual instances, we can anticipate the potential distribution of new classes that may emerge in the future. This is demonstrated in the lower left corner of Fig. 3. The loss function $\mathcal{L}_P$ is designed to move the mixed instance $\mathbf{z}$ towards the opposite direction of the base vector until it reaches the lower hemisphere. As a result, the embedding space is divided into upper and lower hemispheres, which respectively accommodate the base classes and the future classes. This approach ensures that there is nearly half of the hypersphere of embedding space left for the possible emergence of future classes.

4.4. The Overall Loss

To summarize, our approach involves three loss functions. First, the cross entropy loss $\mathcal{L}_{CE}$ is used to learn the base data. Second, the loss $\mathcal{L}_B$ helps to distribute the embeddings of base classes in the upper hemisphere. Finally, the loss $\mathcal{L}_P$ helps to reserve the lower hemisphere for future classes. These three loss functions are integrated to form the overall loss function of the proposed method.

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha\mathcal{L}_B + \beta\mathcal{L}_P. \tag{9}$$

4.5. Rectification at Inference

By assigning base session features and incremental session features to the upper and lower hemispheres of the hypersphere, respectively, we can identify the session of a given sample based on its similarity to the base vector, and thus predict its label more accurately.

$$P^r(y_i) = \begin{cases} P(y_i) \times \eta, & sim_B(\mathbf{x_i}) < 0 \; and \; y_i \in Y_0 \\ P(y_i), & otherwise \end{cases}, \tag{10}$$

where $P(y_i)$ is the prediction for class y_i, and $\eta \in [0,1]$ is penalty factor. In cases where the similarity of a sample

Method	Accuracy in each session(%)									PD↓	Relative Improvement
	0	1	2	3	4	5	6	7	8		
Finetune	64.10	39.61	15.37	9.80	6.67	3.80	3.70	3.14	2.65	61.45	+00.00
iCaRL [19]	64.10	53.28	41.69	34.13	27.93	25.06	20.41	15.48	13.73	50.37	+11.08
Rebalancing [11]	64.10	53.05	43.96	36.97	31.61	26.73	21.23	16.78	13.54	50.56	+10.89
TOPIC [26]	64.10	55.88	47.07	45.16	40.11	36.38	33.96	31.55	29.37	34.73	+26.72
FSLL+SS [15]	66.76	55.52	52.20	49.17	46.23	44.64	43.07	41.20	39.57	27.19	+34.26
CEC [37]	73.07	68.88	65.26	61.19	58.09	55.57	53.22	51.34	49.14	23.93	+37.52
PASS [41]	79.25	71.77	68.77	63.67	59.80	57.12	54.41	52.93	50.96	28.29	+31.62
ALICE [18]	79.00	70.50	67.10	63.40	61.20	59.20	58.10	56.30	54.10	24.90	+36.55
C-FSCIL [10]	77.47	72.40	67.47	63.25	59.84	56.95	54.42	52.47	50.47	27.00	+34.45
FACT [39]	74.60	72.09	67.56	63.52	61.38	58.36	56.28	54.24	52.10	22.50	+38.95
SAVC [23]	78.63	73.01	69.62	65.26	61.49	59.24	57.34	55.09	53.19	25.68	+35.77
EHS[1]	73.98	70.11	66.66	62.75	60.11	57.33	55.59	53.75	51.59	**22.39**	**+39.06**
EHS[2]	71.27	67.40	63.87	60.40	57.84	55.09	53.10	51.45	49.43	**21.84**	**+39.61**

Table 1. Top 1 classification accuracy of each session on CIFAR100 dataset. We calculate performance dropping rate (PD) and show the relative performance improvement of PD towards Finetune.

with the base vector is below 0, this indicates that the sample's feature distribution is located in the lower hemisphere of the hypersphere, which suggests that the sample is less likely to belong to the classes in the base session. Consequently, the confidence of the base class needs to be reduced to account for this.

5. Experiments

5.1. Implementation Details

Datasets. Following the past works, we mainly conduct comparison and ablation study on two datasets, CIFAR-100 [12] and *mini*ImageNet [20]. CIFAR-100 contains 60,000 images from 100 classes. Following the split of past works [26], we divide them into 60 base classes for base session and 40 few-shot classes for 8 incremental sessions. In the base session, each class has 600 images with a resolution of 32 $\times$ 32, while new classes are formulated into eight 5-*way* 5-*shot* incremental tasks. Similarly, we also split the *mini*ImageNet dataset into 60 base classes and 40 few-shot classes for 8 incremental sessions with the resolution of 84 $\times$ 84.

Evaluation Protocol. Following prevailing works [26], we measure the Top-1 accuracy after the i-th session as $\mathcal{A}$. Besides, we also measure the degree of forgetting with performance dropping rate (PD), *i.e.*, PD= $\mathcal{A}_0 - \mathcal{A}_B$, where $\mathcal{A}_0$ denotes accuracy after base session and $\mathcal{A}_0$ stands fo accuracy after the last session. We use the identical training splits (including base and incremental sessions) for every compared method for a fair comparison.

Training Details. We deploy all the models with PyTorch [17]. For CIFAR-100, we adopt ResNet20 [9] as backbone, while for others we use ResNet18. The model is trained with a batch size of 256 for 1000 epochs, and we use SGD with momentum for optimization. The learning rate starts from 0.1 and decays with cosine annealing. Both balanced weights α and β are set as 2.5 in EHS[1] and 3 in EHS[2], while the penalty factor η is set as 0.8.

5.2. Comparison with State-of-the-Art

We compare our method with several proposed FSCIL methods. Finetune is regarded as the lower-bound of the FSCIL setting. We compare relative improvement of PD of all methods with respect to Finetune. Firstly, We compare with classic CIL methods: iCaRL [19] and Rebalancing [11]. For SOTA FSCIL methods, TOPIC [26], FSLL+SS [15], CEC [37], ALICE [18], PASS [41], C-FSCIL [10], FACT [39] and SAVC [23] are comopared with our algorithm. Among them, **PASS [41], ALICE [18], FACT [39] and SAVC [23] are prototype-based methods** and use cosine classifiers as we do. Besides, FACT and SAVC are both methods enhancing reserved space. For our method, EHS[2] slightly sacrifices the accuracy of base session on the basis of EHS[1], so that the base features and features in incremental sessions are more precisely distributed in the upper and lower hemispheres. Specifically, we increase the values of both $\mathcal{L}_B$ and $\mathcal{L}_P$ from 2.5 to 3 to enhance the trend of updating the model to obtain a more sparse feature representation for FSCIL.

Results on CIFAR100. We can infer from Tab. 1 that our method outperforms the current SOTA methods. For iCaRL [19] and Rebalancing [11], which can alleviating catastrophic forgetting in CIL setting, the relative improvement of PD toward Finetune is only 11.08 and 10.89. It

Method	Accuracy in each session(%)									PD↓	Relative Improvement
	0	1	2	3	4	5	6	7	8		
Finetune	61.31	27.22	16.37	6.08	2.54	1.56	1.93	2.60	1.40	59.91	+00.00
iCaRL [19]	61.31	46.32	42.94	37.63	30.49	24.00	20.89	18.80	17.21	44.10	+15.81
Rebalancing [11]	61.31	47.80	39.31	31.91	25.68	21.35	18.67	17.24	14.17	47.14	+12.77
TOPIC [26]	61.31	50.09	45.17	41.16	37.48	35.52	32.19	29.46	24.42	36.89	+23.02
FSLL+SS [15]	68.85	63.14	59.24	55.23	52.24	49.65	47.74	45.23	43.92	24.93	+34.98
CEC [37]	72.00	66.83	62.97	59.43	56.70	53.73	51.19	49.24	47.63	24.37	+35.54
PASS [41]	79.68	70.16	67.04	63.12	61.33	58.56	55.97	54.83	53.04	26.64	+33.27
ALICE [18]	80.60	70.60	67.40	64.50	62.50	60.00	57.80	56.80	55.70	24.90	+35.01
C-FSCIL [10]	76.40	71.14	66.46	63.29	60.42	57.46	54.78	53.11	51.41	24.99	+34.92
FACT [39]	72.56	69.63	66.38	62.77	60.60	57.33	54.34	52.16	50.49	22.07	+37.84
SAVC [23]	81.10	76.25	72.45	68.99	66.55	63.10	59.97	58.25	57.21	23.89	+36.02
EHS[1]	71.25	66.65	62.84	59.65	56.90	54.14	51.63	50.05	49.06	**22.19**	**+37.72**
EHS[2]	69.43	64.86	61.30	58.21	55.49	52.77	50.22	48.61	47.67	**21.76**	**+38.15**

Table 2. Top 1 classification accuracy of each session on *mini*ImageNet dataset. We calculate performance dropping rate (PD) and show the relative performance improvement of PD towards Finetune.

shows that simply CIL methods are affected by overfitting due to few-shot instances. TOPIC [26], FSLL+SS [15] and C-FSCIL [10] do not use prototype to replace classifer and update model in the inremental session. They get better performance than simply CIL approaches but worse than those prototype-based methods, which may because the effect of overfitting and catastrophic forgetting is not completely eliminated. Although CEC [37] does not use the prototype neither, it introduces an additional graph neural network to combine classifiers learned on individual sessions for incremental learning, which brings a lower performance dropping rate (23.93). Using the prototype and cosine classifier as well, our approach achieves lower performance dropping rate (22.39 and 21.84).

Results on *mini*ImageNet. Similar to the performances on CIFAR100, our model also shows improvement on the *mini*ImageNet dataset, which is shown in Tab. 2. FSCIL methods still outperform approaches only proposed for CIL. Nevertheless, except for FACT [39], the performances of prototype-based methods do not differ much from other methods. For example, the performance dropping rate of ALICE [18] and C-FSCIL [10] are very close (24.90 and 24.99).Our EHS[2] (21.76) outperforming FACT [39] (22.07). Even without sacrificing performance in the base session, our performance (22.19) is very close to FACT [39] (22.07) and far superior to other methods (at least 8%).

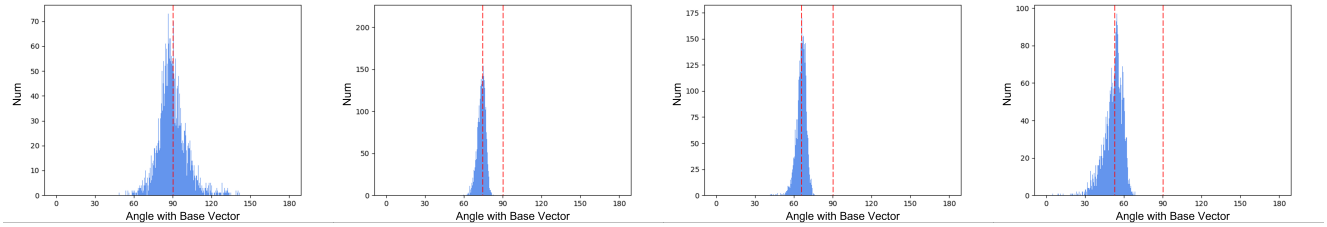
5.3. Ablation Study

In Tab. 3, we conduct ablation study on CIFAR100 and *mini*ImageNet. Our proposed EHS method involves two loss terms, and the choice of activation function also plays a crucial role in our learning paradigm.

Activation Function	$\mathcal{L}_B$	$\mathcal{L}_P$	CIFAR100		*mini*ImageNet	
			PD	Δ_{imp}	PD	Δ_{imp}
relu	×	×	25.72	+0.00	25.57	+0.00
tanh	×	×	23.55	+2.17	22.54	+3.03
tanh	✓	×	22.80	+2.92	22.43	+3.14
tanh	×	✓	24.53	+1.19	22.74	+2.83
tanh	✓	✓	22.39	**+3.33**	22.19	**+3.38**

Table 3. Ablation studies on CIFAR100 and *mini*ImageNet. $\mathcal{L}_B$ and $\mathcal{L}_P$ are terms of regularization in our loss. PD means performance dropping rate. Δ_{imp} denotes relative performance improvement toward simply prototype method.

We utilized the *tanh* activation function, which is an odd-symmetric activation function, to extend the embedding space to a complete hypersphere. By applying $\mathcal{L}_B$, we pushed embeddings of base classes to the hemisphere, and $\mathcal{L}_P$ reserved the lower hemisphere for pseudo instances, making it more suitable for incoming novel classes. Our experiments show that replacing the commonly used *relu* activation function with *tanh* greatly decreased the performance dropping rate by 2.17 on CIFAR100. This replacement increased the completeness of the feature space and made it more sparse for the distribution of embeddings. The addition of $\mathcal{L}_B$ further improved the sparsity of the feature space and reduced catastrophic forgetting by 2.17 compared to simply prototype methods and 0.75 compared to using *tanh* alone on CIFAR100. **However, only using $\mathcal{L}_P$ does not significantly improve the model's ability to eliminate forgetting. Since there is no regularization for the true feature distribution of the base session, pushing the pseudo features to the lower hemisphere alone was insufficient to reduce the overlapping area of base classes and novel classes.** The optimal performance is achieved by combining



(a) without $\mathcal{L}_B$ (b) margin=0.1 (c) margin=0.2 (d) margin=0.3

Figure 4. The distributions of features in base session corresponding to different margins (m_p). The distance between two red dashed lines reflects the degree of feature shift affected by $\mathcal{L}_B$.

both loss terms, $\mathcal{L}_B$ and $\mathcal{L}_P$, as shown in the last line of Table 3. This is because the interaction between these two loss terms constrains the base classes to the upper hemisphere of the feature space and pushes the features corresponding to the pseudo instances to the lower half of the hypersphere, effectively reducing the overlapping area between base and novel classes. The use of $\mathcal{L}_B$ and $\mathcal{L}_P$ brings more improvement on CIFAR100 while *tanh* plays a more important role on *mini*ImageNet.

5.4. Further Analysis

In this section, we will analyze the effect of margin on feature distribution in our method. For different margins (m_p), the effect of $\mathcal{L}_B$ on the distribution of features in base session is shown in Fig. 4.

The distribution of embeddings when $\mathcal{L}_B$ is not used to specify feature distribution for base classes is shown in (a). In this case, embeddings are found at the junction of the upper and lower hemispheres, and the angle between average embeddings and the base vector is $90°$. However, when $\mathcal{L}_B$ with margin is employed in (b), (c), and (d), the shift in features is evident, and the distribution of all base classes becomes closer to the base vector as the average angle decreases. As a result, all features in the base session are distributed in the upper hemisphere of the hypersphere, which is the region where the angle is less than $90°$.

As the margin value in $\mathcal{L}_B$ increases, the features tend to move closer to the base vector, specifically towards the "upper pole" of the hypersphere. With a small margin, the model's feature distribution becomes more compact due to the regularization effect, resulting in a reduced range of angle between embeddings and base vector ((a) $\rightarrow$ (b)). However, with a large margin, it becomes more challenging to optimize $\mathcal{L}_B$, especially for a model with gradient descent update. This leads to a different degree of sample shift towards the "upper pole," which is reflected in the larger distribution area shown in (c) $\rightarrow$ (d).

5.5. Discussion

As discussed in Section 5.4, our proposed loss is effective in handling feature shift and leaves enough space in the lower hemisphere for the distribution of new class features. However, there is still room for improvement as there is an unused area near the base vectors in the upper hemisphere. It may be beneficial to reserve this space for novel classes by restricting the maximum similarity between features and base vectors in our loss $\mathcal{L}_B$. Alternatively, increasing the margin without setting an upper bound will cause the embeddings to move closer to the base vector, leading to the compression of the feature space and a smaller area for feature distribution. Ideally, if we can reserve most of the hypersphere for future classes, it would be more advantageous. **In addition, compared to *relu*, using *tanh* may cause problems such as gradient vanishing during the training process. Future work will include the seek of a odd-symmetric activation function that combines the advantages of *relu*.**

6. Conclusion

In recent years, there has been an increasing interest in few-shot class-incremental learning (FSCIL). This paper presents a novel approach that addresses the limitations of existing prototype-based methods. One of the key issues with these methods is that embeddings are only distributed over a small portion of the complete hypersphere space, which limits their ability to handle incremental learning. To address this, we propose using an odd-symmetric activation function instead of the commonly used *relu*, which allows for a more complete and sparse embedding space. Additionally, we allocate space for features of base and incremental classes, with the help of pseudo instances generated by mixed base samples. Our experimental results demonstrate that this approach achieves State-of-the-Art performance.

Acknowledgement. This research was supported by the Natural Science Fund of Hubei Province under Grant 2022CFB823, the HUST Independent Innovation Research Fund under Grant 2021XXJS096, the Alibaba Innovation Research program under Grant CRAQ7WHZ11220001-20978282, and grants from the Key Lab of Image Processing and Intelligent Control, Ministry of Education, China.

References

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

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018. 2

[3] Sébastien Arnold, Shariq Iqbal, and Fei Sha. When maml can adapt fast and how to assist when it cannot. In *International Conference on Artificial Intelligence and Statistics*, pages 244–252. PMLR, 2021. 2

[4] Yu-Ying Chou, Hsuan-Tien Lin, and Tyng-Luh Liu. Adaptive and generative zero-shot learning. In *International conference on learning representations*, 2021. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 2

[7] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[8] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):1–36, 2017. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[10] Michael Hersche, Geethan Karunaratne, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. Constrained few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9057–9067, 2022. 6, 7

[11] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1, 3, 6, 7

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

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

[14] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 438–455. Springer, 2020. 2

[15] Pratik Mazumder, Pravendra Singh, and Piyush Rai. Few-shot lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2337–2345, 2021. 3, 6, 7

[16] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[18] Can Peng, Kun Zhao, Tianren Wang, Meng Li, and Brian C Lovell. Few-shot class-incremental learning from an open-set perspective. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV*, pages 382–397. Springer, 2022. 2, 3, 6, 7

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 6, 7

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 6

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[22] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 2, 3

[23] Zeyin Song, Yifan Zhao, Yujun Shi, Peixi Peng, Li Yuan, and Yonghong Tian. Learning with fantasy: Semantic-aware virtual contrastive constraint for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24183–24192, 2023. 6, 7

[24] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018. 2

[25] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 1

[26] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12183–12192, 2020. 3, 6, 7

[27] Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. Gcr: Gradient coreset based replay buffer

selection for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 99–108, 2022. 3

[28] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 2

[29] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049, 2017. 2

[30] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8012–8021, 2021. 2

[31] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 1

[32] Xiang Xiang, Yuwen Tan, Qian Wan, Jing Ma, Alan Yuille, and Gregory D Hager. Coarse-to-fine incremental few-shot learning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*, pages 205–222. Springer, 2022. 3

[33] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021. 1, 3

[34] Han-Jia Ye, Yi Shi, and De-Chuan Zhan. Identifying ambiguous similarity conditions via semantic matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16619, 2022. 2

[35] Han-Jia Ye, De-Chuan Zhan, Yuan Jiang, and Zhi-Hua Zhou. Heterogeneous few-shot model rectification with semantic mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3878–3891, 2020. 2

[36] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12203–12213, 2020. 2

[37] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12455–12464, 2021. 2, 3, 6, 7

[38] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13208–13217, 2020. 1

[39] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9046–9056, 2022. 2, 3, 5, 6, 7

[40] Qinhao Zhou, Xiang Xiang, and Jing Ma. Hierarchical task-incremental learning with feature-space initialization inspired by neural collapse. *Neural Processing Letters*, pages 1–17, 2023. 3

[41] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021. 6, 7

[42] Kai Zhu, Yang Cao, Wei Zhai, Jie Cheng, and Zheng-Jun Zha. Self-promoted prototype refinement for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6801–6810, 2021. 2