Decision Boundary Optimization for Few-shot Class-Incremental Learning

Chenxu Guo\textsuperscript{1}, Qi Zhao\textsuperscript{1}, Shuchang Lyu\textsuperscript{1}, Binghao Liu\textsuperscript{1}, Chunlei Wang\textsuperscript{1}, Lijiang Chen\textsuperscript{1} and Guangliang Cheng\textsuperscript{*2}

\textsuperscript{1}Department of Electronic Information Engineering, Beihang University
\textsuperscript{2}Department of Computer Science, University of Liverpool

Abstract

Few-shot class-incremental learning (FSCIL) is gaining prominence in real-world machine learning applications, including image classification and face recognition. Existing methods often employ parameter freezing for the backbone and classify based on metric learning. However, these methods suffer from two significant problems. Firstly, training the backbone solely on base classes limits its performance on novel classes due to information loss. Secondly, conventional metric-based strategies for prototype generation tend to introduce confusion in decision boundaries during few-shot tasks. To address these challenges, we propose a novel approach called Decision Boundary Optimization Network (DBONet) for few-shot class-incremental learning. To tackle the first issue, DBONet incorporates an augmentation feature extractor along with a corresponding loss function. This augmentation feature extractor combines samples from different categories to capture richer features. For the second issue, we leverage limited sample representativeness information by introducing the Prototype Generation Module (PGM) into DBONet, enabling the generation of more representative prototypes. The prototypes produced by PGM significantly contribute to the accurate delineation of decision boundaries. Furthermore, we exploit intra-class information to enhance classification precision. Extensive experiments on CIFAR100, miniImageNet, and CUB200 datasets demonstrate that our proposed approach achieves new state-of-the-art results.

1. Introduction

Image recognition has garnered significant attention in recent years [23, 25, 27, 38, 41]. The ability of models to acquire new visual knowledge from limited samples has become a central focus for numerous researchers [10, 11, 42]. In practical applications, visual recognition systems often encounter challenges related to data privacy or device memory limitations, which pose difficulties in retraining the model using previously learned data [46]. To address this issue, class-incremental learning (CIL) has emerged as a dedicated approach, where the learning of each class is treated independently. The primary objective of CIL is to enable a learning system to continuously acquire new knowledge from novel classes while retaining a substantial portion of the previously learned knowledge [36].

Conventional class-incremental learning (CIL) tasks typically operate under the assumption that an ample amount of data for novel categories is available for model learning. However, in practical scenarios, it is often the case that there is an insufficiency of data pertaining to novel categories, thereby necessitating the learning system to effectively acquire knowledge using limited samples [45, 60]. These specific learning scenarios are referred to as Few-Shot Class-Incremental Learning (FSCIL) [46].

FSCIL exhibits remarkable parallels with the learning process observed in human beings. Humans continuously receive and assimilate new knowledge from their surroundings, and this learning occurs gradually over time [19]. Importantly, humans tend to retain the vast majority of the knowledge they have acquired, making it challenging to forget previously learned information. Furthermore, hu-
mans possess the remarkable ability to leverage their existing knowledge to comprehend novel content. This ability to build upon prior knowledge plays a vital role in the human learning process.

Deep learning models currently fall short of achieving human-level performance, particularly when trained on limited samples, which often leads to underfitting issues. Additionally, disregarding previous data while learning new classes can result in catastrophic forgetting of the model [36]. This problem is particularly pronounced when dealing with limited samples of novel categories compared to conventional continual incremental learning (CIL) tasks.

In recent times, several studies have employed metric learning strategies to preserve the model’s recognition ability on base classes. Existing FSCIL frameworks [5, 6, 9, 13, 37, 46, 55, 63] have made significant advancements. However, these methods primarily rely on optimizing the cross-entropy loss for a single category and cannot be effectively applied when dealing with dual category virtual samples [53, 15].

The semantic features across different categories typically exhibit a lack of generality, giving rise to three primary challenges when dealing with novel class samples in comparison to previous class samples. Firstly, the features of novel classes are often not fully expressed, resulting in the feature vectors’ norm being statistically smaller than that of the base class samples. Consequently, the model tends to confuse categories between novel and base classes, for instance, mistaking dolphins from the base class for sharks in the novel class. As a result, severe category confusion problems often occur. Secondly, the support set’s samples are often non-representative, a common issue in real-world applications like face recognition. Obtaining ideal photos with strong feature representation in few-shot conditions, such as frontal shots under good lighting, is challenging. To overcome this, we propose a prototype generation module (PGM) that produces superior class prototypes in latent space, mitigating the impact of potential outliers on class representation learning. Additionally, our analysis reveals varying intra-class variances for each class in the latent space. This difference in the spatial distribution of feature vectors between classes poses challenges when using the equal category decision method. To address this, we propose a novel classifier based on intra-class variance for a more accurate decision boundary in latent space.

To address the aforementioned challenges, we present a novel architecture DBONet, which comprises three key components: the augmentation feature extractor, the prototype generation module, and the intra-class variance classifier. Specifically, the augmentation feature extractor aims to obtain robust global features from images, thus achieving fully expressed features. The prototype generation module (PGM) aims to create more suitable prototypes that adapt well to conditions with limited samples. Finally, the intra-class variance classifier is introduced to tackle the decision boundary shift problem. By effectively eliminating above issues, our approach further enhances the overall performance and accuracy of the model.

The contributions of this paper can be summarized as follows:

- A novel end-to-end FSCIL learning framework, Decision Boundary Optimization Network (DBONet), is proposed to acquire highly representative features, thus achieving much better performance.
- A Prototype Generation Module (PGM) is proposed, whereby the utilization of representative samples enables the derivation of significantly improved class prototypes in the latent space.
- An intra-class variance classifier is employed to adaptively adjust the class decision boundary to reduce confusion between classes.
- Extensive experiments on three benchmark datasets demonstrate the state-of-the-art (SOTA) performance of the proposed method in FSCIL tasks.

2. Related Work

Class-incremental learning (CIL). The main content of CIL is making learning system can continually learn knowledge without forgetting [26, 28, 62]. The current methods can be generally divided into three types. Regularization Approaches [1, 20, 35, 58] mainly adopt regularization terms and classification loss to alleviate catastrophic forgetting. Some methods which using rehearsal and replay mechanism [16, 33, 36, 40] to prevent the forgetting of previous tasks. Another group of study [2, 14, 52] aim to mute bias from most recently learned task to tackle CIL tasks. In CIL tasks, novel classes usually have adequate samples. So many methods proposed for CIL task may suffer from reduced efficacy with the condition of limited novel samples.

Few-shot learning (FSL). The purpose of few-shot learning is to be able to learn valid information from limited samples, which usually requires pre-training or meta-learning to obtain a model that can quickly adapt to few-shot scene [34, 43, 50]. Related few-shot learning study primarily includes model-based, metric-base, optimization-based methods. Model-based methods [29, 30, 31] involve model architectures specifically tailored for fast learning. Metric-based methods [17, 22, 32, 42, 45, 49, 56] focus on how to pull support samples and query samples in latent space, while scaling up the distance between different classes. Optimization-based methods [11, 39, 44] learn an optimizer through meta learning which can quickly adapt to new categories with limited samples. Most of FSL study don’t consider differentiating between base classes and novel classes together.
Few-shot class-incremental learning (FSCIL). There are some finetune-based methods to tackle FSCIL tasks. Finetune-based methods optimize parameters with novel data using various balance mechanisms to solve catastrophic forgetting and overfitting problems [5, 9, 19, 37, 46, 63]. Recently, methods which freeze backbone during novel classes learning perform significant results. Zhang et al. [59] use a pseudo incremental learning method to train an attention-base module to enhance the model performance. Zhou et al. [61] propose forward compatible training by assigning virtual prototypes to compress the embedding of base classes and reserve space for novel classes. The above methods do not effectively solve the problem of confusion and scope imbalance in the latent space. We reckon that the decision boundary of the latent space can be further optimized by adjusting the prototype.

Prototypical learning (PL). The basic assumption of prototypical learning is that each category has a potential center, and samples between different categories are approximately separable in latent space [24, 42]. Yue et al. [57] leverage prototype which can preserve semantic structure for Unsupervised Domain Adaptation. Chen et al. [4] perform fine-grained image classification with comparing part of images and prototypes. Li et al. [21] use limited support samples as prototype to guide few-shot image segmentation. There are also some study which aim to exploit potential of the prototype sufficiently. Yang et al. [54] propose a new approach to handle outlier data by utilizing trained prototypes and an assumed Gaussian distribution. Deng et al. [8] argue that the prototype should be treated as a distribution instead of a point in the latent space. Inspired by above mentioned prototypical learning researches, we propose to sufficiently explore the role of prototypes to tackle the FSCIL tasks.

3. Method

3.1. Problem Formulation

In an N-way K-shot FSCIL task, let training set streams as $D_0, D_1, D_2, \ldots, D_n$, totally $(n+1)$ sessions, and the corresponding samples label sets are $Y_0, Y_1, Y_2, \ldots, Y_n$. The test set streams is $D'_0, D'_1, D'_2, \ldots, D'_n$, and the corresponding samples label sets are $Y'_0, Y'_1, Y'_2, \ldots, Y'_n$. $D_0$ is the base session training sets with sufficient samples. The training label sets from different sessions are disjoint, which can be formulated as $Y_i \cap Y_j = \emptyset$ for $i \neq j$. In other words, the model can only access a set of specific categories of training data at different sessions. At the end of each session, the model needs to evaluate on the previous data and current data together. So the test set label space in $j$-th session can be summarized as $Y'_j = Y_0 \cup Y_1 \cup Y_2 \cup \ldots \cup Y_{i-1} \cup Y_i$. The novel sessions in training set usually have only a limited amount of samples. Without loss of generality, we use N classes, and each class has K samples. Taking min/Imagenet dataset as an example, in the session 0, there are 60 base classes, and each class has 500 training samples. Other sessions contain 5 ways $\times$ 5 shot, totally 25 training samples for each session.

3.2. Augmented Feature Training

Our training pipeline mainly comprises two stages. In the first stage, we train the encoder using feature augmentation to obtain a global feature representation of each image. In the second stage, we categorize samples into two types based on their representativeness and then train the prototype generation module to obtain a prototype for each class.

In the previous work CEC [59], encoders and attention mechanisms were trained using a feature enhancement method involving image rotation. Building upon this, we introduce an innovative approach to enrich the feature space further, leveraging manifold mixup augmentation [48]. The architecture of our proposed method is illustrated in Fig. 2. The $h(\cdot)$ indicates the pre-encoder which consists of the first three layers of ResNet [12] backbone, and the $g_a(\cdot)$ and $g_n(\cdot)$ indicate the stable feature extractor and the augmentation feature extractor. This can be shown as following:

$$E(x) = \text{Concat}(g_a(h(x)), g_n(h(x)))$$  \hspace{1cm} (1)

where $E(x)$ is the output global feature. In the following, we will use the $E(\cdot)$ to represent the inference function of the DBONet.

As depicted in Fig. 2, the model processes an input image and generates two distinct feature vectors using the stable feature extractor and the augmentation feature extractor. We refer to these feature vectors as the ”stable feature” and the ”augmented feature” of the image, respectively. Subsequently, these two feature vectors are concatenated to create a comprehensive global feature representation for the image.

Our study aims to optimize high-level semantic features by fusing diverse categories of image features. To achieve this, we follow a three-step process: 1) We input pairs of images from different classes into the pre-encoder, resulting in two dense feature maps. 2) The two feature maps obtained from different categories are fused using a randomly selected weight coefficient $\lambda$ from the interval [0.45, 0.55]. 3) These fused feature maps are then passed through two feature extraction modules $g_a$ and $g_n$, dedicated to extracting stable and augmented features, respectively.

$$F' = \lambda * h(x_i) + (1 - \lambda) * h(x_j)$$ \hspace{1cm} (2)

The process involves taking images $x_i$ and $x_j$ from different base categories and passing them through the third layer of ResNet to obtain dense features $h(x_i)$ and $h(x_j)$. These dense features are fused to create the combined feature $F'$. Subsequently, the fused dense feature is fed into an
augmentation feature extractor to generate a global feature representation. Finally, the global feature representation is multiplied with matrix $W$ as described in Eq. 3 to obtain the logits output, and the loss is calculated using Eq. 4.

$$l = fc(g_a(F')) = W^T g_a(F')$$

where $l$ is the logit output and $W$ is the classifier weight for each class.

$$L_B = BICELoss(l, \lambda_{y_1}, \lambda_{y_2}, y_1, y_2)$$

$$= -\sum_{i=1,2} \lambda_{y_i} l_{y_i} - \log(\sum_{j=1,2} \lambda_{y_j} e^{l_{y_j}} + \sum_{i \neq y_1, j \neq y_2} e^{l_i})$$

where $y_1$ and $y_2$ are the labels of the samples which are fused, and $\lambda_{y_1}$ and $\lambda_{y_2}$ ($\lambda_{y_1} + \lambda_{y_2} = 1$) are the corresponding fusion weight for the sample feature vectors. $l_{y_1}$ and $l_{y_2}$ are the logit output value for the class $y_1$ and $y_2$. The goal of $BICELoss$ is to enhance the recognition of virtual categories by effectively separating them from other class prototype vectors within the latent space.

In addition, we also optimize the pre-encoder $h(\cdot)$ and the stable feature extractor $g_a(\cdot)$ with using traditional cross entropy loss. So our training pipeline in stage 1 can be formulated as Algorithm 1.

### 3.3. Cosine Variance Classifier

Consider a simple toy example, illustrated in Fig. 3, where two distinct categories exhibit Gaussian distributions with varying variances in the latent space. The ideal classification boundary, represented by the green dashed line in Fig. 3, should accurately separate the two categories. However, when utilizing an unweighted Euclidean distance measurement for classification, the decision boundary of the two distributions is depicted as the red line in Fig. 3, which lies at the midpoint between the two distributions. Unfortunately, this red line classification boundary leads to an increased classification error. To address this issue, we propose the need for further optimization of the decision boundary within the latent space. Our objective is to minimize the classification error as much as possible, ensuring an accurate separation of categories and improving the overall performance of the classification model.

In the latent space, the feature vectors of each class of image occupy a certain range. In the experiment, we find that the range of different categories is not always allocated according to the average distance, and the range occupied by different categories in latent space is also different.
3.4. Prototype Generation Module

We propose a prototype generation module to generate prototype vectors closer to the potentially general center for each category, and calculate the average cosine variance for each category to measure the coverage of the category on the latent hypersphere space. To split representative and weakly representative samples from training set, we need to use the Eq. 8 to calculate the probability that one sample belongs to its label category. And by comparing the probability and threshold $\epsilon$, we can select representative samples. And then, we label the samples of the base class 1 if it is representative, else 0. After that, we use the encoder $E(\cdot)$ to extract the feature vectors of all samples of the training set. And we train a prototype generation module with this feature set, the function of the module is to generate a more precise prototype vector based on the representative samples of the support set for a class. Here we use the transformer architecture [47] for the prototype generation module, the training process is shown in Algorithm 2.

$$p(y = i|x) = \frac{S(E(x); \mu_i, \sigma_i, 1)}{\sum_{j=1}^{C} S(E(x); \mu_j, \sigma_j, 1)} \quad (8)$$

where $C$ is the number of total classes.
Table 1: Comparison with the state-of-the-art on CIFAR100 dataset. The * denotes result report in corresponding paper.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average class-wise accuracy (%)</th>
<th>PD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Ft-CNN*</td>
<td>64.10</td>
<td>39.61</td>
</tr>
<tr>
<td>iCaRL* [36]</td>
<td>64.10</td>
<td>53.28</td>
</tr>
<tr>
<td>EEIL* [3]</td>
<td>64.10</td>
<td>53.11</td>
</tr>
<tr>
<td>Rebalancing* [14]</td>
<td>64.10</td>
<td>50.05</td>
</tr>
<tr>
<td>TOPIC* [46]</td>
<td>64.10</td>
<td>58.88</td>
</tr>
<tr>
<td>Decoupled-Cosine* [49]</td>
<td>74.55</td>
<td>67.43</td>
</tr>
<tr>
<td>CEC* [59]</td>
<td>73.07</td>
<td>68.88</td>
</tr>
<tr>
<td>Fact* [61]</td>
<td>74.60</td>
<td>72.09</td>
</tr>
</tbody>
</table>

Table 2: Comparison with the state-of-the-art on miniImageNet dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average class-wise accuracy (%)</th>
<th>PD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Ft-CNN*</td>
<td>61.31</td>
<td>27.22</td>
</tr>
<tr>
<td>iCaRL* [36]</td>
<td>61.31</td>
<td>46.32</td>
</tr>
<tr>
<td>EEIL* [3]</td>
<td>61.31</td>
<td>46.58</td>
</tr>
<tr>
<td>Rebalancing* [14]</td>
<td>61.31</td>
<td>48.31</td>
</tr>
<tr>
<td>TOPIC* [46]</td>
<td>61.31</td>
<td>50.09</td>
</tr>
<tr>
<td>Decoupled-Cosine* [49]</td>
<td>70.37</td>
<td>65.45</td>
</tr>
<tr>
<td>CEC* [59]</td>
<td>72.00</td>
<td>66.83</td>
</tr>
<tr>
<td>Fact* [61]</td>
<td>72.56</td>
<td>69.63</td>
</tr>
</tbody>
</table>

Table 3: Comparison with the state-of-the-art on CUB200 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average class-wise accuracy (%)</th>
<th>PD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Ft-CNN*</td>
<td>68.68</td>
<td>43.70</td>
</tr>
<tr>
<td>iCaRL* [36]</td>
<td>68.68</td>
<td>52.65</td>
</tr>
<tr>
<td>EEIL* [3]</td>
<td>68.68</td>
<td>53.63</td>
</tr>
<tr>
<td>TOPIC* [46]</td>
<td>68.68</td>
<td>62.49</td>
</tr>
<tr>
<td>Decoupled-Cosine* [49]</td>
<td>75.52</td>
<td>70.95</td>
</tr>
<tr>
<td>CEC* [59]</td>
<td>75.85</td>
<td>71.94</td>
</tr>
<tr>
<td>Fact* [61]</td>
<td>75.90</td>
<td>73.23</td>
</tr>
</tbody>
</table>

Table 4: Average class-wise accuracy of base and novel categories.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR100 base</th>
<th>CIFAR100 novel</th>
<th>miniImageNet base</th>
<th>miniImageNet novel</th>
<th>CUB200 base</th>
<th>CUB200 novel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average class-wise accuracy (%)</td>
<td>PD</td>
<td>Average class-wise accuracy (%)</td>
<td>PD</td>
<td>Average class-wise accuracy (%)</td>
<td>PD</td>
</tr>
<tr>
<td>Decoupled-Cosine</td>
<td>72.2</td>
<td>10.9</td>
<td>69.3</td>
<td>10.03</td>
<td>76.1</td>
<td>13.73</td>
</tr>
<tr>
<td>CEC</td>
<td>71.7</td>
<td>15.3</td>
<td>69.8</td>
<td>14.35</td>
<td>76.4</td>
<td>16.19</td>
</tr>
<tr>
<td>Fact</td>
<td>75.4</td>
<td>17.43</td>
<td>71.0</td>
<td>16.65</td>
<td>78.24</td>
<td>23.265</td>
</tr>
<tr>
<td>DBONet</td>
<td>77.11</td>
<td>23.785</td>
<td>73.7</td>
<td>22.59</td>
<td>79.23</td>
<td>25.705</td>
</tr>
</tbody>
</table>

and 2 weakly representative samples to further exploit the representation capture ability of the PGM.

And then we use the $L_p$ mentioned in Algorithm 2 to constrain the prototype to groundtruth, and the groundtruth is set as the average feature vector of all samples of this class. Through above training process, we can train a PGM to obtain a accurate prototype of one class as much as possible with the condition of limited samples. We empirically set the hyperparameter $\gamma$ as 0.25 and the representativeness threshold $\epsilon$ as 0.03.

4. Experiments

4.1. Datasets and implementation details

We follow TOPIC [46] and use datasets CIFAR100 [18], miniImageNet [7] and Caltech-UCSD Birds-200-2011(CUB200) [51] to evaluate our methods. CIFAR100 has 100 categories of images. Each category has 500 training images and 100 test images. Size of each image is 32 $\times$ 32 pixels. miniImageNet also has 60,000 images with 100 classes. And per class has 500 training images and 100 testing images. Size of each image is 84 $\times$ 84 pixels. CUB200 is a fine-grained image classification benchmark, which consists of 200 different species of birds. It contains 5994 images for training and 5794 images for testing. Each image has a size of 224 $\times$ 224 pixels. In FSCIL task, we follow the setting as the same as previous.
Table 5: Ablation study on three benchmark datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Average class-wise accuracy (%)</th>
<th>PD↓</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>baseline</td>
<td>77.13</td>
<td>72.05</td>
</tr>
<tr>
<td></td>
<td>gₚₐ</td>
<td>77.67</td>
<td>73.14</td>
</tr>
<tr>
<td></td>
<td>gₚₐ+PGM</td>
<td>77.67</td>
<td>73.33</td>
</tr>
<tr>
<td></td>
<td>gₚₐ+PGM+IC</td>
<td>77.81</td>
<td>73.62</td>
</tr>
<tr>
<td>minImageNet</td>
<td>baseline</td>
<td>73.82</td>
<td>70.56</td>
</tr>
<tr>
<td></td>
<td>gₚₐ</td>
<td>74.24</td>
<td>70.92</td>
</tr>
<tr>
<td></td>
<td>gₚₐ+PGM</td>
<td>74.24</td>
<td>71.04</td>
</tr>
<tr>
<td></td>
<td>gₚₐ+PGM+IC</td>
<td>74.53</td>
<td>71.55</td>
</tr>
<tr>
<td>CUB200</td>
<td>baseline</td>
<td>77.82</td>
<td>74.16</td>
</tr>
<tr>
<td></td>
<td>gₚₐ</td>
<td>78.13</td>
<td>74.98</td>
</tr>
<tr>
<td></td>
<td>gₚₐ+PGM</td>
<td>78.13</td>
<td>75.23</td>
</tr>
<tr>
<td></td>
<td>gₚₐ+PGM+IC</td>
<td>78.66</td>
<td>75.33</td>
</tr>
</tbody>
</table>

Figure 4: Ablation study on three benchmarks. The results show that our proposed module has incremental improvements for FSCIL task. Compared to baseline model, we improve by 3.87%, 4.08%, 3.18% on the last session class-wise average accuracy for CIFAR100, minImageNet and CUB200, respectively. And we achieve 3.19%, 3.37%, 2.34% decrease on performance drop for CIFAR100, minImageNet and CUB200.

works [46, 59, 61]. For CIFAR100, there are 60 categories as the base data and the remaining 40 categories as the novel data. The model learns novel data in the manner of 5way × 5shot, with a total of 8 sessions. And for minImageNet, we divide it into 60 and 40 categories, with the same setting of CIFAR100. For CUB200, we divide it into 100 and 100 categories, respectively. The model learns novel data in the manner of 10way × 5shot, with a total of 10 sessions. We use ResNet20 as the backbone for the CIFAR100 dataset, and the corresponding pre-encoder $h(\cdot)$ structure is consistent with the first two residual layers of ResNet20, $gₚ(\cdot)$ and $gₙ(\cdot)$ structure are consistent with the last layer of ResNet. We train 100 epochs with an initial learning rate of 0.01. We adopt ResNet18 as the backbone architecture for the minImageNet dataset. The pre-encoder function $h(\cdot)$ aligns structurally with the first three layers of ResNet18, while the structures of $gₚ(\cdot)$ and $gₙ(\cdot)$ are analogous to the last layer of ResNet18. The training process consists of 100 epochs, initialized with a learning rate of 0.01. For the CUB200 dataset, we maintain network consistency with the architecture used for minImageNet. The training process spans 100 epochs, commencing with an initial learning rate of 0.001. Our implementation is based on the PyTorch library, and we employ SGD with momentum as the optimization algorithm, alongside milestones as the scheduler.

4.2. Comparison with the State-of-the-art Methods

We report the performance over benchmark datasets CIFAR100, minImageNet and CUB200 in Tab. 1, Tab. 2 and Tab. 3. Compared with current SOTA results, we obtained 3.68%, 3.33% and 0.86% improvements at last session class-wise average accuracy on CIFAR100, minImageNet and CUB200 datasets, respectively. In order to maintain consistency in model capacity, we halve the convolution kernel width of the last layer of ResNet, so as not to bring about excess parameters and computations. The last layer width of ResNet20 for CIFAR100 is 64, which we reduced to 32, minImageNet and CUB200 use ResNet18, and the width of the last layer is 512, which we changed it to 256. The performance drop is the result that the session 0 accuracy subtract the last session accuracy. Our performance improvement is also reflected in the term of performance drop, which means that our classification accuracy improvement is not only dependent on the base classes.

We also calculate the average accuracy of the base class and the novel class separately, and we can see that our model mainly improves classification performance on the novel classes compared with base classes. Compared to the Fact [61], we have improved the classification accuracy of the base classes by 1-2%, and the accuracy of the novel classes can be improved by 2-6%, which is primarily attributed to our innovative approach involving feature aug-
mentation training and prototype vector optimization. Compared to previous freezing backbone methods, our proposed method achieves better performance balance on base classes and novel classes.

![Figure 5: Channel width of the last layer (g_a, g_s) study on three benchmark datasets.](image)

![Figure 6: Performance with different hyperparameters γ on three benchmark datasets.](image)

![Figure 7: t-SNE visualization on miniImageNet dataset.](image)

### 4.3. Ablation study

On the basis of building the baseline method, we gradually add the required modules. The results are shown in Tab. 5 and Fig. 4. g_a in Tab. 5 represents the use of augmented features and BICELoss in Eq. 4 to train the g_a extractor module, while baseline uses conventional samples and cross entropy loss for training. PGM in Tab. 5 denotes the use of prototype generation module to generate prototype vectors. Finally, the IC in Tab. 5 indicates the classifier based on cosine variance is added. To assess the effectiveness of our method, we conduct ablation experiments on all three datasets. To ensure a fair comparison, we train g_a without utilizing manifold mixup features by substituting it with a convolutional layer of the same structure. We train the alternative model using cross-entropy loss and then concatenate the feature vectors obtained from the last two layer modules. This demonstrates the contribution of manifold mixup features in our overall approach.

### 4.4. Analysis

To investigate the impact of prototype dimension, we conduct an experiment with different output dimensions of two feature extractors across three datasets. The results are presented in Fig. 5. Notably, as the number of dimension increased, our model exhibited additional performance improvements. In Fig. 5, the variable d represents the sum of the last layer width of the feature extractors g_a(·) and g_s(·).

We conduct a comparative experiment (Fig. 6) to investigate the impact of parameter γ. Setting γ to 1 result in a significant decrease in accuracy, indicating that excessive introduction of intra-class variance information confuses the embedded spatial classification boundary. Conversely, with γ set to 0.25, we achieve the highest classification accuracy in the final parameter session, reaching a better equilibrium state.

As shown in the Fig. 7, we visualize the feature vectors of test set samples of categories 0, 5, 10, · · ·, 95 (we treat the first category as 0) by using the t-SNE algorithm. In order to facilitate the presentation of the results, we only select 50 samples from each category. Fig. 7 (a) shows the output result of the conventional network structure introduced in Sec. 4.3, and Fig. 7 (b) presents the output result of the augmentation feature extractor. From the results we can observe that the distribution of samples of novel class are more dispersed compared with most of the base classes. The features by the augmentation feature extractor are often more compact, especially on categories 65, 70, 75 and 95.

### 5. Conclusion

In this paper, we propose a framework based on decision boundary optimization to apply in few-shot class-incremental learning, and use the manifold mixup for feature augmentation to further improve the feature extraction ability. The PGM produces much better class prototype to reduce the classification error caused by the decision boundary shift. The classifier considering the intra-class variance further obtains a more accurate decision boundary in the latent space. Our extensive experiments show that proposed method achieves SOTA performance.

**Acknowledgements.** This work was supported by National Natural Science Foundation of China under Grants 62072021.
References


[55] Sung Whan Yoon, Do-Yeon Kim, Jun Seo, and Jaekyun Moon. Xrernet: Learning to extract task-adaptive representa-


