PACE: Prototypical Adaptive Contrastive Embeddings

Cao Yu
Singapore Management University
yu.cao.2022@msc.smu.edu.sg

Dylan Hillier
Singapore Management University
das.hillier.2023@phdcs.smu.edu.sg

Neel Modha
Singapore Management University
neelmodha.2022@mitb.smu.edu.sg

Li Xingyun
Singapore Management University
xingyun.li.2022@mitb.smu.edu.sg

Abstract

In this paper, our team discusses the approach we take to solve the Class Incremental Learning (CIL) challenge on a 100-class bird dataset. This CIL challenge requires the classification of fine-grained bird classes, necessitating the use of a finely tuned model that can retain high-level class-specific nuances across phases of training while only using few samples from each class. Our solution, Prototypical Adaptive Contrastive Embeddings (PACE), leverages a two-staged approach: in the initial stage we fine-tune adaptive embedding heads using a contrastive loss. We store prototypes of the classes using k-means centroids and pass these to future stages as part of the history; this is then followed by a second stage in which we train a logistic regression head on the class prototypes after passing them through the adaptors. Our comprehensive evaluation shows that PACE surpasses competing methods, achieving 79.8% in last accuracy and 85.6% in average accuracy, as evidenced by our number 1 ranking on the private leaderboard.

1. Introduction

Class Incremental Learning refers to the process where a machine learning model gradually learns new classes or tasks over time without forgetting previously learned information.

In traditional machine learning, models are trained on a fixed dataset with a fixed set of classes. However, in real-world applications, data continuously evolves, and new classes or tasks may emerge. Class incremental learning addresses this scenario by allowing a model to incrementally learn new classes without retraining on the entire dataset or forgetting the previously learned information.

1.1. Project Task

In the task, we are presented with a 100-class bird dataset, which is to be trained across 10 phases, with each phase containing 10 classes. This is the subset of the CUB-200 dataset[7]. This task involves two key challenges: class incremental learning and fine-grained image classification. In addition, we are limited to using models pre-trained or fine-tuned on ImageNet-1K, the combined size of all the models cannot exceed 600MB, and we must use at most 5 samples for replay from each class.

1.2. Challenges of Class Incremental Learning

Class incremental learning, where a deep learning model learns new classes over time without forgetting the previously learned ones, presents several challenges. The key challenge here is catastrophic forgetting, where as a model learns new classes, there is a risk of forgetting or overwriting previously learned information. It happens when the model updates its parameters to accommodate new knowledge, causing a degradation in performance on previous tasks.

1.3. Challenges of Fine-Grained Image Classification

The challenges of CIL discussed above are further exacerbated by the fact that our task involves fine-grained image classification. Fine-grained image classification refers to the task of categorizing images into highly specific or fine-grained classes, often within a broader category or superclass. Unlike general image classification, which might classify images into broader categories like ”dogs” or ”cars,” fine-grained classification focuses on distinguishing between subcategories that share similar visual features, such as birds with small changes in their features, as part of our task. Fine-grained image classification datasets moreover are typically less data rich, making the challenge even harder given the high intra-class variance and subtle visual differences between classes.

2. Related Works

2.1. Transformers for Fine-grained Image Classification

Fine-grained image classification is a distinct subset of image classification focused on distinguishing between categories that are closely similar and often easily confused due to their subtle differences. In TransFG [3], the authors address the limitations of CNNs in capturing nuanced differences among fine-grained classes by employing self-attention mechanisms. By dividing images into regions, they show that the transformer architecture used in TransFG hierarchically processes and focuses on crucial features, enabling the model to discern subtle differences. Results from their experiments show that Transformer-based architectures have an advantage over traditional CNN-based models for fine-grained classification, therefore we repeat this finding and broadly speaking see better performance of transformer-based approaches.

2.2. Representation Learning for Class Incremental Learning

In *iCaRL* [5] the authors introduce a dual strategy involving both representation learning and classifier expansion. It is of particular relevance that they also investigate improvements in representations of previous classes. Firstly, they build a set of exemplars for each class chosen to be close to the class mean. They then utilise these to update a mean prototype for each class that is used for classification. This is similar to the process by which we use k-means centroids – the major difference being that since we freeze our embeddings we can directly pass the centroidal representation, which is significantly compressed. Nevertheless, this makes it difficult for us to adapt the base embedding model. The other major feature of their work is the use of a distillation loss to constrain the model during adaptation. While we do not use such a constraint, our use of residuals ensures that the model only learns to adapt the existing embeddings to maximize contrast and arguably plays a similar role.

In Foster [9] the authors investigate the transfer of knowledge between different phases of learning. In particular, they apply strategies involving feature boosting and compression, as well as knowledge distillation, to augment class representations. As with the comparison to iCaRL, we do not directly use such knowledge distillation, however, our use of residuals plays a similar role. We can also compare the feature boosting to our use of contrastive loss which aims to adapt the embeddings by changing features to be more discriminative between classes.

Similarly in *Revisiting CIL* [10], the authors introduce an approach for using prototypes from pre-trained embedding models as class representations that are then used to classify based on distance – they call this approach *Simple CIL*. Through experiments on benchmark datasets, they show that *Simple CIL* is able to outperform existing methods based on the strength of pre-trained embedding models alone. This reinforces our results in simply adapting pre-trained embeddings with k-means history – the main difference being that we still train a logistic regression classifier on top of this. The authors further consider a method for adapting these existing embeddings they call *ADAM* – they freeze the pre-trained embeddings and add an additional model fine-tuned only on the first phase of the data. They are only able to do this on the first incremental dataset due to the necessity of keeping the validity of the feature prototypes, in contrast to our approach which learns adaptors that adapt the class embeddings themselves. In any case, the paper shows the generalizability and adaptivity of pre-trained models, and the fact that this *already have sufficient discriminative power for fine-grained classification tasks.*

2.3. Sample Efficient Fine-tuning

Our work directly adapts SetFit [6] in which the authors explore the adaptation of pre-trained language models in few shot settings without the use of prompts. The relevance here is that we similarly aim to leverage powerful pre-trained embeddings. Their key insight is to use contrastive learning to increase the sample efficiency of the fine-tuning process. Using the same technique of pairwise comparisons between classes we are able to increase the size of our training data by a square factor. The contrastive approach is thus more sample efficient and can help prevent overfitting to particular samples. In our case, we can use it to directly adapt a class embedding without having to add a new classification head for calculating a classification loss. This obviates the use of additional parameters during this fine-tuning. Additionally, the authors also inspire the notion of using lightweight logistic regression heads as a final fine-tuning step.

3. Method

3.1. Baseline setup

As with the authors of [10] it becomes apparent to us that pre-trained vision models (PTM) largely possess the discriminative capacity to perform well on the fine-grained classification task. As such we establish a baseline only using the memory budget of 5 samples per class, and training a logistic regression classifier on this history during each phase. The classifier is dropped from each previous phase while doing this.

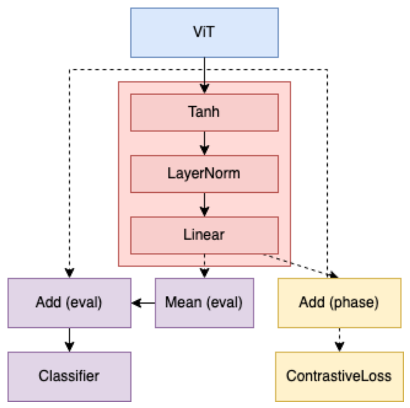

Figure 1. Architecture of FT Head

We discover that we achieve better performance using the ImageNet classifier head and feeding this into the logistic regression head over feeding in the featural representation. The finding is consistent across different PTMs.

3.2. Memory

Our first improvement is to improve the memory representation. Rather than taking samples from each class randomly, we leverage unsupervised clustering approaches in order to build better class representations. In particular, to take the pre-trained embeddings from the PTM and use K-means clustering on the embeddings from each class, resulting in five centroids as our new memory for that class. We note that this enables our model to achieve very strong performance even without any fine-tuning on the per-phase data, implying that the k-means representation is representative of each class. Furthermore, this representation is entirely dependent on the choice of PTM. We suspect that the varying performance of different PTMs is likely due to our utilization of the geometric properties inherent in the class embedding space of each PTM. It is feasible that a model with discriminative geometric properties in its class embedding space may have had less use in representing features that may have been used by other methods. We can consider these k-means centroid samples to be analogous to the *prototypes* in [5]

3.3. Adaptive Finetuning Heads

While the logistic regression is already able to efficiently learn from the k-means representation of each class, in order to maximize the efficient use of data in each phase we explore methods for adapting the embeddings. Our approach is to use a lightweight residual fine-tuning head (FT head), as pictured in figure 1. This head is adapted from the head in [8].

Our training process largely follows the practices established in [6]. We build a balanced set of positive and negative contrastive samples X_+, X_- in the following way. For each sample x_1, x_2 from the same class we add a 'positive' triple $(x_1, x_2, 1)$ to X_+. Thus this has $n_c \times n_c - 1$ instances. We then sample x_3 from the remaining classes to acquire a 'negative' triple for each positive triple and add $(x_1, x_3, 0)$ to X_-. We train over the set $X = X_- \bigcup X_+$, where our contrastive loss is just squared cosine similarity loss:

$$\mathcal{L}_{\text{cont}}(x_1, x_2, y) = \begin{cases} (1-\cos(x_1, x_2))^2, & \text{if } y = 1 \\ (\cos(x_1, x_2))^2, & \text{if } y = 0 \end{cases} \tag{1}$$

3.4. Aggregation

In order to utilise these embeddings for the logistic regression classifier, we must combine the different adaptors from each phase. We thus experiment with a number of different *aggregation* methods. In practice, we simply take the mean of the residual representations $\sum_i^N (\text{FT}_i(\text{PTM}(x)) - \text{PTM}(x))/N$ and add this to the pre-trained embedding, which we refer to as *add* aggregation.

3.5. PACE

Our overall methodology is pictured in figure 2, and consists in the following:

1. In each phase we feed images through the PTM and obtain embeddings.
2. We build a k-means centroid representation of each class from these embeddings and store them in the history
3. We then contrastively train a lightweight residual head to modify these base embeddings at inference time, *only* using the data from that phase. This is denoted by FT_i
4. In the final phase we train a new logistic regression classifier over the entire history/memory, aggregating the representations from each phase as our input to the classifier.

4. Experiments

Several experiments are conducted from different perspectives to evaluate the proposed method. We mainly focus on the top 1 accuracy during different phases to understand the performance. In some experiments, we also calculate the average accuracy (AA) over 10 phases and the accuracy of the last phase (LA) to compare the overall performance in different methods.

4.1. CIL Methods Comparison

Figure 3 shows the accuracy of 10 phases by using different class incremental learning methods. For a fair comparison, we use the same pre-trained model, the RepViT network pre-trained on ImageNet1K dataset, to extract the image embedding. The proposed method outperforms others, it tackles the catastrophic forgetting issue quite well.

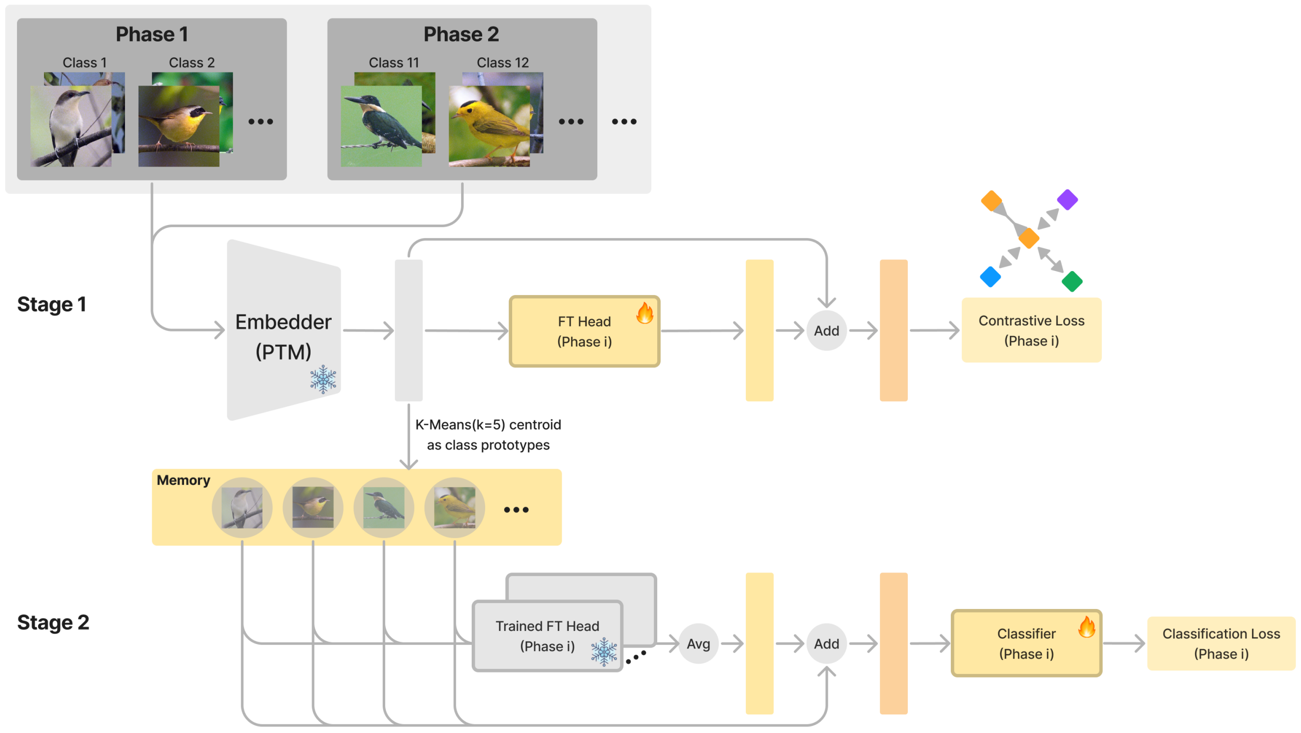

Figure 2. Overview of PACE architecture.

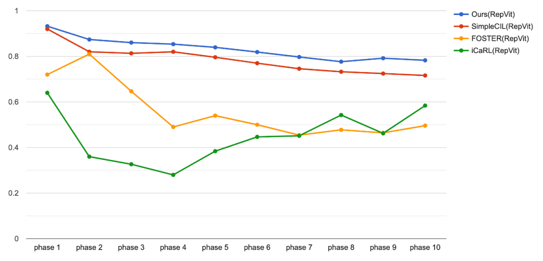

Figure 3. Accuracy comparison of different CIL methods during 10 phases.

4.2. Embedding Models Comparison

To understand the influence of embedding methods, we take several pre-trained models in the proposed CIL strategy and compare the accuracy over 10 phases. Most of the pre-trained embedding models in the experiments are trained on ImageNet1K dataset. The figure 4 shows that pre-trained embedding methods have strong effects on the accuracy of each phase. We note that when the pre-trained model is trained on a larger scale dataset, the overall performance is much better (the VitB16In21kFtin1k denotes the ViT base with 16 patches, trained on ImageNet21K dataset, then fine-tuned on the ImageNet1K dataset). Indeed this result matches the state of the art reported in [10]. More generally, the results between ResNet50 and ViT-Base-16 models are very close, hence there is little support for the claim that ViT models are inherently more suitable for this task, compared with classic CNN architectures.

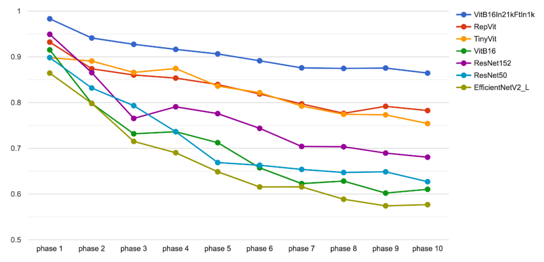

Figure 4. Accuracy comparison of different embedding models during 10 phases.

We also visualise the image features extracted by two pre-trained models, to comprehend if the networks can pay attention to the key area of the images. From figure 5, the better pre-trained model (RepViT) can make the focused key area even smaller, which benefits the differentiating of intra-variances.

4.3. Weak-label Boosting

In the project settings, we have two extra unlabeled datasets, the validation dataset in the public leaderboard stage which

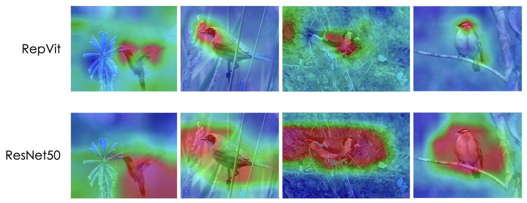

Figure 5. Visualisation of image features between two embedding models.

Method	LA(%)	AA(%)
Ours(RepVit)	78.3	83.3
Ours(RepVit)+Val	77.6	82.5
Ours(RepVit)+Val+Test	79.4	84.2

Table 1. Comparison of different weak-label data for boosting.

has about 5 images per class and the test dataset in the private leaderboard stage which has about 25 images per class. The weak-label boosting experiment is conducted to explore if the extra weak-labeled data can boost the overall performance. Figure 6 illustrates the effects of using different scales of weak-labeled data. With more weak-labeled data, the overall performance is boosted by about 1% (as shown in table 1). Although the scale of the experiment is quite limited, it could still give a hint that the model has better tolerance when it has enough correct labels.

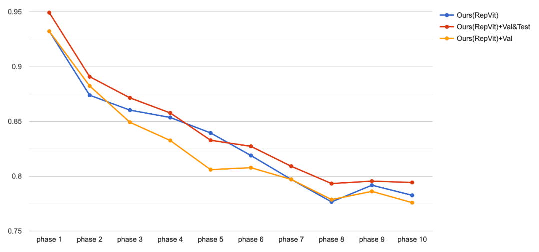

Figure 6. Accuracy comparison of different weak-label data for boosting.

4.4. Ablation Study

To understand the importance of different components in the proposed CIL strategy, three ablation studies are conducted.

4.4.1 History Building Methods

Three different ways of building history prototypes are explored in this section. Firstly, just use 5 random samples (image embeddings by a pre-trained model) in each class as the prototypes. In each phase, the new classifier is trained over these history prototypes and prototypes in the current

Method	LA(%)	AA(%)
Random(5 samples)	73.4	79.0
K-means(k=5, w/o fine-tune)	76.4	82.2
K-means(k=5, w/ fine-tune)	78.3	83.3

Table 2. Comparison of different history building methods.

phase. Secondly, we use 5 centroids of k-means cluster as 5 prototypes of each class (in the pre-trained model image embedding space). Lastly, we also use 5 centroids of k-means cluster as prototypes, but the pre-trained model image embedding space is fine-tuned in the project dataset. The results are shown in table 2. Figure 7 shows the last strategy of building history attains higher accuracy.

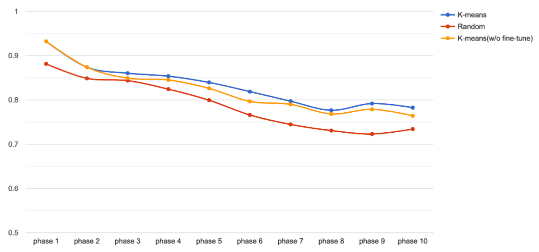

Figure 7. Accuracy comparison of different history building methods.

4.4.2 Embedding Aggregation Methods

In this experiment, it figures out the effects of different aggregation approaches. Figure 8 shows that simply adding the embedding of the pre-train model with the mean of embedding over different phases, can perform a slightly good performance during 10 phases.

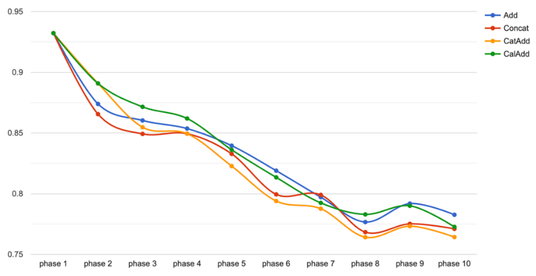

Figure 8. Accuracy comparison of different embedding aggregation methods.

4.4.3 Classification Head

In the study, four different classification heads are exploited, table 3 displays the results. The regression approach can achieve reasonable performance, as shown in figure 9.

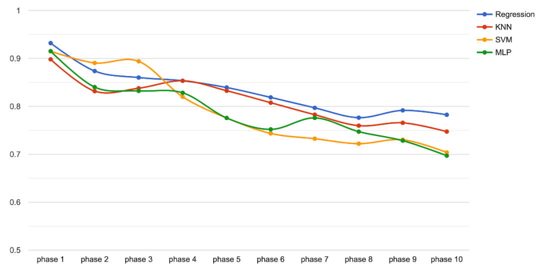

Figure 9. Accuracy comparison of different classification heads.

Method	LA(%)	AA(%)
MLP	69.7	78.9
SVM	70.4	79.3
KNN	74.8	81.2
Regression	78.3	83.3

Table 3. Comparison of different classification heads.

5. Conclusion

In this work, we introduce a two-stage prototype-based class incremental learning method which can adapt to new phases by optimizing adaptations to embeddings via contrastive loss. Our experiments show the importance and influence of the choice of PTM, as well as the significant impact of using prototypes. Finally, we find that contrastive fine-tuning can have a weak impact on improving performance.

In our experiments we also investigate the difference between the embeddings of ViT and CNN embedding models and find no evidence that ViT models are better for fine-grained classification tasks, although we do find indicative evidence that the particular model we use (RepViT) has a stronger representation. Additionally we investigate the use of weak-labeled data and find little impact.

In our ablation studies we find in particular that the use of prototypes helps mitigate catastrophic forgetting by providing a compact representation of the data that can be fit in memory, and used to train a classification head. Our fine-tuning approach has less clear benefits. While it does consistently increase performance, especially in earlier stages of the training, we find that it is quite sensitive to hyperparameters, and given that our residual representations are averaged over it is unclear how well these benefits are retained in the case of more classes.

As such future work (for instance with adaptors like LoRa[4]) could be done to improve the adaptation process. The challenge of this is maintaining the validity of the k-means based representation of the classes while changing the underlying embedding models. Other future work could involve investigating the use of generative data augmentation and class compression, and exploring the performance of the techniques in settings with even more phases.

In the competition, PACE outperforms methods proposed by other teams, which enables us to be number 1 in the final private leaderboard. Furthermore, we find that it is roughly as performant as state-of-the-art methods, especially when freed from the constraints of using pre-trained models only from ImageNet1k[1]. Indeed we are able to reach about 86.2% in LA and 90.5% in AA with ViT-Base distilled from ImageNet21k[2] – this is the same as reported in [10] for their *ADAM* approach but with a weaker base model. Furthermore, our approach is trainable with a very minimal computing budget, and we are able to train all ten phases in around ten minutes using only a 16GB MPS processor.

References

[1] 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. 6

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6

[3] Ju He, Jie-Neng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, and Changhu Wang. Transfg: A transformer architecture for fine-grained recognition, 2021. 2

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 6

[5] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017. 2, 3

[6] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts, 2022. 2, 3

[7] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 1

[8] Ao Wang, Hui Chen, Zijia Lin, Hengjun Pu, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. *arXiv preprint arXiv:2307.09283*, 2023. 3

[9] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning, 2022. 2

[10] Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need, 2023. 2, 4, 6