Dynamic Pseudo-Label Assignment and Consistent Prototypical Learning for Few-Shot Class-Incremental Learning

Supplementary Material

Introduction

We will provide more detailed information in the supplementary material. In particular, Section A is a description of the inference details, Section B is a hyperparametric sensitivity analysis, Section C is an addition of more experimental details, and Section D is an expansion of more ablation details. Also, we have anonymized and open-sourced the code of the experimental results for reproduction.

A. Inference Details

When performing inference on base tasks, it is only necessary to classify according to the global classification layer weight matrix W of the existing base classes. In incremental tasks, since the model needs to perform inference on all learned tasks, we need to combine $\widetilde{W}$ and P_v in actual inference. Specifically, when the test sample x is input into the model, its class label satisfies:

$$\begin{aligned} P(y_i|\varphi(x)) &= P\left(P_j^t|\varphi(x)\right) \\ &= \sum_{l=1,\cdots,L} P\left(P_j^t|P_{\nu_l},\varphi(x)\right) P(P_{\nu_l}|\varphi(x)) \end{aligned} \tag{1}$$

where $P(P_{\nu_l}|\varphi(x)) = \frac{s(\varphi(x),P_{\nu_l})}{\sum_{k=1}^{L} s(\varphi(x),P_{\nu_k})}$. Using Bayes' theorem to expand $P\left(P_j^t|P_{\nu_l},\ \varphi(x)\right)$, we obtain:

$$P\left(P_j^t|P_{\nu_l},\varphi(x)\right) = \frac{P\left(\varphi(x)|P_j^t,P_{\nu_l}\right) P\left(P_j^t|P_{\nu_l}\right)}{\sum_{P_{\nu_k}\in P_\nu} P\left(\varphi(x)|P_j^t,P_{\nu_k}\right) P\left(P_j^t|P_{\nu_k}\right)} \tag{2}$$

For ease of analysis and calculation, we assume that $P\left(\varphi(x)|P_j^t,P_{\nu_l}\right) = \lambda\mathcal{N}(\varphi(x);P_j^t,\Sigma) + (1-\lambda)\mathcal{N}((x);P_{\nu_l},\Sigma)$ follows a linear Gaussian mixture distribution. At the same time, $P\left(P_j^t|P_{\nu_l}\right) = \mathcal{N}(P_j^t;P_{\nu_l},\Sigma)$ follows a Gaussian distribution. Therefore, $P\left(P_j^t|P_{\nu_l},\varphi(x)\right)$ can be expressed as:

$$\frac{M_\lambda(n(P_j^t,\varphi(x)),n(P_{\nu_l},\varphi(x)))n(P_{\nu_l},P_j^t)}{\sum_{P_{\nu_k}\in P_\nu} M_\lambda(n(P_j^t,\varphi(x)),n(P_{\nu_k},\varphi(x)))n(P_{\nu_l},P_j^t)} \tag{3}$$

where $n(a,b) = \exp\left((\Sigma^{-1}a)^T b - \frac{1}{2}a\Sigma^{-1}a\right)$, $M_\lambda(a,b) = \lambda a + (1-\lambda)b$. Since we use cosine-based similarity when training the classifier in the base task, we normalize $\varphi(x)$, P_j^t, and P_{ν_l} in the actual calculation. At the same time, assuming that Σ is a unit matrix, Eq. 2 can be further expressed as:

$$\frac{M_\lambda\left(\exp\left((P_j^t)^T(P_{\nu_l}+\varphi(x))\right),\exp\left((P_{\nu_l})^T(P_j^t+\varphi(x))\right)\right)}{\sum_{P_{\nu_k}\in P_\nu} M_\lambda\left(\exp\left((P_j^t)^T(P_{\nu_k}+\varphi(x))\right),\exp\left((P_{\nu_k})^T(P_j^t+\varphi(x))\right)\right)} \tag{4}$$

In actual inference, the strength of the transfer of pseudo-class knowledge from the base task to the incremental tasks can be controlled by adjusting the value of λ. In the experiment, we set λ to 0.1 based on experience.

B. Hyperparameter Sensitivity Analysis

In order to explore the sensitivity of the model performance to each hyperparameter, we conducted a sensitivity analysis on the CUB200 dataset using the average accuracy as the evaluation metric, and the results are shown in Figure 1, Figure 2, and Figure 3. For the hyperparameter α, the

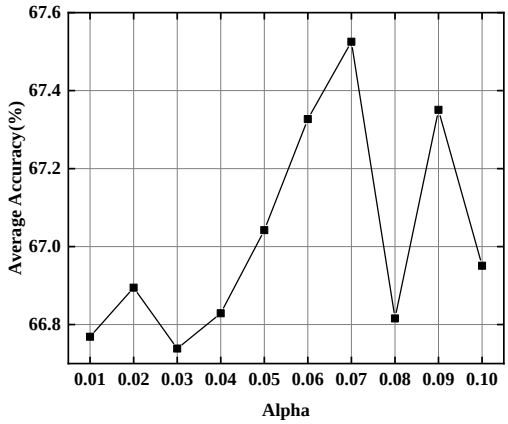

Figure 1. Influence of hyperparameter α on average accuracy of DPCL on CUB200 dataset.

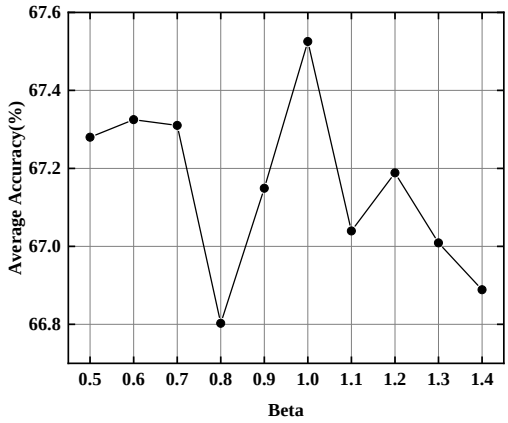

Figure 2. Influence of hyperparameter β on average accuracy of DPCL on CUB200 dataset.

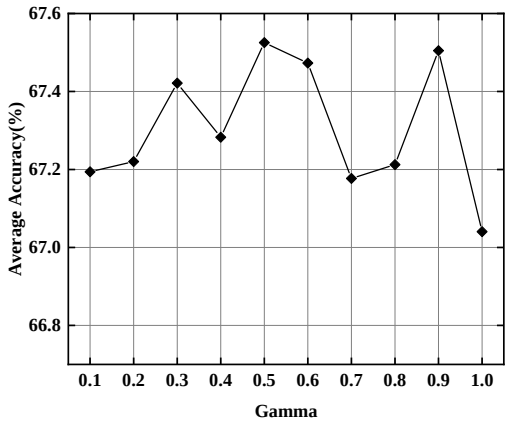

Figure 3. Influence of hyperparameter γ on average accuracy of DPCL on CUB200 dataset.

scanning was performed in the interval $[0.01,\ 0.1]$ with a step size of 0.01. The results show that the model performance peaks at $\alpha = 0.07$. For the hyperparameter β, it was explored in the interval $[0.5,\ 1.4]$ in steps of 0.1, and the results showed that it performed best at $\beta = 1.0$. The hyperparameter γ was evaluated in steps of 0.1 in the interval $[0.1,\ 1.0]$, with the optimal value occurring at $\gamma = 0.5$.

The analysis reveals that the optimal values (*i.e.*, $\alpha = 0.07$, $\beta = 1.0$, $\gamma = 0.5$) are in perfect agreement with our selected configuration, confirming the validity of the current hyperparameter selection.

C. More Experiment Details

C.1. Implementation Details

In terms of dataset partitioning, we strictly follow the benchmark settings proposed by [9]. All datasets adopt standard preprocessing procedures and data enhancement strategies, including random resized crop, horizontal flipping, and color jittering, to enhance the generalization ability of the models.

In the model training configuration, we adopt different backbone networks and training strategies according to the characteristics of the dataset. For CIFAR100 and miniImageNet, we use a randomly initialized ResNet-12 network. In our actual implementation, CIFAR100 is trained for 210 epochs with 300 episodes per epoch, while miniImageNet is trained for 310 epochs with 300 episodes per epoch. The optimization process uses the Stochastic Gradient Descent (SGD) and introduces the Nesterov momentum (with a coefficient of 0.9). The initial learning rate is set to 0.1, and the Warmup–CosineAnnealing–Cooldown scheduler is applied to decay the learning rate. For the CUB200 dataset, we used the pre-trained ResNet-18 model provided by [14] as the backbone network. The training period is set to 100 epochs, and each epoch contains 30 episodes. The optimizer also uses SGD with Nesterov momentum (0.9), and the initial learning rate is set to a low 0.002. Concurrently, the learning rate is adjusted by a Warmup-CosineAnnealing-Cooldown scheduler.

The DPCL method proposed in this paper is implemented based on the PyTorch deep learning framework. All experiments are done on a computing platform equipped with Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz, 2 NVIDIA GeForce RTX 4090 GPUs.

C.2. Algorithm Details

We divide the model training process into two stages: base task training (Algorithm 1) and incremental task training (Algorithm 2).

Algorithm 1 Base Task of DPCL Training

Input: Hyperparameters α, β, γ, Base task dataset D_0, Virtual prototypes P_ν
Parameter: Model parameters F_θ
Output: Trained model for base task

Initialize model parameters F_θ.
repeat
  for $e \in \{1, \cdots, \text{episodes}\}$ **do**
    Resample D_0 using N_0-way K_0-shot.
    Splitting Support Sets and Query Sets.
    Generating pseudo-samples.
    Calculate prototypes P_i^0 using support set.
    Compute base class prototype loss $\mathcal{L}_{\text{base proto}}$.
    Compute base class classification loss $\mathcal{L}_{\text{base cls}}$.
    Assign pseudo-labels to pseudo-classes using a dynamic pseudo-label assignment mechanism based on P_ν.
    Compute pseudo-class classification loss $\mathcal{L}_{\text{pseudo cls}}$.
    Get the total loss:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{base proto}} + \beta\mathcal{L}_{\text{base cls}} + \gamma\mathcal{L}_{\text{pseudo cls}}$$

    Update F_θ based on the total loss $\mathcal{L}$.
  end for
until the predefined number of epochs is reached.

In the base task training stage, the model parameters are initialized and undergo multiple epochs of iterative optimization. Each epoch consists of several episodes. In each episode, we perform N_0-way K_0-shot resampling on the base task dataset. This divides it into a support set and a query set. We then generate pseudo-samples. Subsequently, we calculate the prototypes of the base classes. The pseudo-samples are processed using a dynamic pseudo-label assignment mechanism based on virtual prototypes.

Table 1. Comparison with SOTA methods on CIFAR100 for FSCIL. * denotes reproduced results.

Methods	Accuracy in each session (%)									Average
	0	1	2	3	4	5	6	7	8	Acc.
Topic [9]	64.10	55.88	47.07	45.16	40.11	36.38	33.96	31.55	29.37	42.62
CEC [13]	73.07	68.88	65.26	61.19	58.09	55.57	53.22	51.34	49.14	59.53
C-FSCIL [3]	77.47	72.40	67.47	63.25	59.84	56.95	54.42	52.47	50.47	61.64
MetaFSCIL [2]	74.50	70.10	66.84	62.77	59.48	56.52	54.36	52.56	49.97	60.79
CLOM [16]	74.20	69.83	66.17	62.39	59.26	56.48	54.36	52.16	48.25	60.34
FACT [14]	74.60	72.09	67.56	63.52	61.38	58.36	56.28	54.24	52.10	62.24
SoftNet [12]	72.62	67.31	63.05	59.39	56.00	53.23	51.06	48.83	46.63	57.57
NC-FSCIL [11]	82.52	76.82	73.34	69.68	66.19	62.85	60.96	59.02	56.11	67.50
GKEAL [15]	74.01	70.45	67.01	63.08	60.01	57.30	55.50	53.39	51.40	61.35
TEEN [10]	74.92	72.65	68.74	65.01	62.01	59.29	57.90	54.76	52.64	63.10
WaRP [4]	80.31	75.86	71.87	67.58	64.39	61.34	59.15	57.10	54.74	65.82
SAVC* [8]	78.60	72.95	68.73	64.59	61.41	58.46	56.29	54.40	52.19	63.07
ALFSCIL [5]	80.75	77.88	72.94	68.79	65.33	62.15	60.02	57.68	55.17	66.75
OrCo* [1]	79.82	63.06	62.33	60.15	58.74	56.62	55.37	54.21	51.12	60.16
DyCR [7]	75.73	73.29	68.71	64.80	62.11	59.25	56.70	54.56	52.24	63.04
ADBS [6]	79.93	75.22	71.11	65.99	62.46	58.38	55.96	53.72	51.15	63.77
DPCL (ours)	**83.50**	**78.79**	**74.41**	**70.17**	**67.04**	**63.73**	**61.18**	**59.12**	**56.54**	**68.27**
	+0.98	**+0.91**	**+1.07**	**+0.49**	**+0.85**	**+0.88**	**+0.22**	**+0.10**	**+0.43**	**+0.78**

The model updates its parameters by weighting and combining three optimization objectives: base prototype loss, base classification loss, and pseudo-class classification loss.

Algorithm 2 Incremental Task of DPCL Training

Input: Incremental task dataset D_{t+1}
Parameter: Encoder $\varphi(\cdot)$, classifier $\widetilde{W}$
Output: Trained model for incremental task

1: Frozing Encoder $\varphi(\cdot)$
2: **for** $i \in \{1, \cdots, |C_{t+1}|\}$ **do**
3: Obtaining prototypes P_i^t using $\varphi(\cdot)$ and $x_i \in D_{t+1}$.
4: Expand classifier by adding new prototype vectors P_i^{t+1} for the incremental classes:

$$\widetilde{W} = \widetilde{W} \cup P_i^{t+1}$$

5: **end for**

During the incremental task training phase, we freeze the encoder obtained from the base task training. For each new class in the incremental task dataset, we use the frozen encoder to compute its corresponding new class prototype. The model extends the classifier by appending the new class prototype vector to the existing classifier weight matrix. This approach enables the model to recognize new classes without updating the encoder, which effectively avoids the forgetting of base classes.

C.3. Result on CIFAR100

The performance comparison results of FSCIL on the CIFAR100 dataset are shown in Table 1. Our proposed DPCL method achieves the highest accuracy across all 9 sessions and attains the best average accuracy of 68.27%.

C.4. Result on CUB200

The performance comparison results of FSCIL on the CIFAR100 dataset are shown in Table 2. It should be noted that since C-FSCIL [3] was not evaluated on CUB200, we excluded it from comparisons on this dataset. For CUB200, DPCL exhibits performance comparable to existing SOTA methods during the early stages of incremental learning. And in the later stages of incremental learning, DPCL had the best results. From a holistic perspective, DPCL achieves a higher average accuracy than the best SOTA methods.

D. More Ablation Details

To maintain consistency with the ablation experiments in Sec. 4.4, all experiments in this section were conducted on CUB200.

Table 2. Comparison with SOTA methods on CUB200 for FSCIL. * denotes reproduced results.

Methods	Accuracy in each session (%)											Average Acc.
	0	1	2	3	4	5	6	7	8	9	10	
Topic [9]	68.68	62.49	54.81	49.99	45.25	41.40	38.35	35.36	32.22	28.31	26.28	43.92
CEC [13]	75.85	71.94	68.50	63.50	62.43	58.27	57.73	55.81	54.83	53.52	52.28	61.33
MetaFSCIL [2]	75.90	72.41	68.78	64.78	62.96	59.99	58.30	56.85	54.78	53.82	52.64	61.93
CLOM [16]	79.57	76.07	72.94	69.82	67.80	**65.56**	63.94	62.59	60.62	60.34	59.58	67.17
FACT [14]	75.90	73.23	70.84	66.13	65.56	62.15	61.74	59.83	58.41	57.89	56.94	64.42
SoftNet [12]	78.11	74.51	71.14	62.27	65.14	62.27	60.77	59.03	57.13	56.77	56.28	63.95
NC-FSCIL [11]	**80.45**	75.98	72.30	**70.28**	68.17	65.16	**64.43**	**63.25**	60.66	60.01	59.44	67.28
GKEAL [15]	78.88	75.62	72.32	68.62	67.23	64.26	62.98	61.89	60.20	59.21	58.67	66.35
TEEN [10]	77.26	76.13	72.81	68.16	67.77	64.40	63.25	62.29	61.19	60.32	59.31	66.63
WaRP [4]	77.74	74.15	70.82	66.90	65.01	62.64	61.40	59.86	57.95	57.77	57.01	64.66
SAVC* [8]	79.83	75.09	71.99	68.01	67.42	64.20	63.86	62.51	61.48	60.97	60.34	66.88
ALFSCIL [5]	79.79	76.53	73.12	69.02	67.62	64.76	63.45	62.32	60.83	60.21	59.30	67.00
OrCo* [1]	74.79	66.12	64.99	63.39	62.12	59.71	59.38	58.46	56.93	57.79	56.84	61.86
DyCR [7]	77.50	74.73	71.69	67.01	66.59	63.43	62.66	61.69	60.57	59.69	58.46	65.82
ADBS [6]	79.99	75.89	72.53	68.33	67.92	64.75	64.10	62.93	61.31	60.88	59.65	67.12
DPCL (ours)	79.63	**76.73**	**73.72**	69.42	**68.29**	65.33	63.48	62.57	**61.82**	**61.39**	**60.40**	**67.53**
	-0.82	**+0.20**	**+0.60**	**-0.86**	**+0.12**	**-0.23**	**-0.95**	**-0.68**	**+0.34**	**+0.42**	**+0.06**	**+0.24**

Table 3. The impact of pseudo-sample generating position.

Generating Position	Accuracy in each session (%)											Average Acc.
	0	1	2	3	4	5	6	7	8	9	10	
Input layer	79.63	**76.73**	**73.72**	**69.42**	**68.29**	**65.33**	**63.48**	**62.57**	**61.82**	**61.39**	**60.40**	**67.53**
Intermediate layer	79.57	76.58	73.62	69.19	68.11	65.05	63.46	61.89	61.20	61.14	60.15	67.27
Output layer	**79.85**	76.07	73.11	68.51	67.32	64.57	63.40	61.95	60.86	60.65	59.84	66.92

D.1. The effects of pseudo-sample generating position

As shown in Table 3, generating pseudo-samples directly at the input layer (raw-pixel) yields the highest accuracy across nearly all sessions, outperforming pseudo-samples synthesized at intermediate or output layers. Mixing at the raw-pixel level preserves richer low-level cues, which leads to pseudo-samples with substantially higher entropy and diversity. Such diversity is essential for constructing pseudo-classes that faithfully mimic the variability of future unseen classes, thereby improving the robustness of incremental learning.

In contrast, synthesizing pseudo-samples at deeper layers consistently underperforms. High-level representations are strongly shaped by semantic compression and are tightly aligned with the base-class manifold. As a result, feature-space mixing tends to produce pseudo-samples that collapse toward existing class prototypes, reducing their novelty and weakening their effectiveness in modeling potential incremental classes. This explains the widening performance gap in later sessions, where generalization to incremental classes becomes increasingly critical.

D.2. Selection of the number of base classes

Using a larger number of base classes to synthesize each pseudo-class consistently improves performance, with the best results obtained when ten base classes are mixed, as shown in Table 4. This trend aligns with the core motivation of DPCL: pseudo-classes should exhibit high information entropy to better simulate the uncertainty and diversity of future novel classes. When only a few base classes (e.g., two) are used, the synthesized pseudo-samples inherit

Table 4. The impact of different base class counts on the synthesis of each pseudo-class.

Base-Class Number	Accuracy in each session (%)											Average Acc.
	0	1	2	3	4	5	6	7	8	9	10	
2	79.13	75.45	72.36	68.06	66.99	64.36	62.61	61.71	60.69	60.67	59.80	66.53
5	79.33	76.36	73.43	69.00	67.87	65.11	63.12	62.05	61.40	60.87	59.99	67.14
10	**79.63**	**76.73**	**73.72**	**69.42**	**68.29**	**65.33**	**63.48**	**62.57**	**61.82**	**61.39**	**60.40**	**67.53**
15	79.33	76.01	73.01	68.36	67.12	64.41	62.52	61.57	61.01	60.56	59.60	66.68
20	79.09	76.05	72.81	68.14	67.12	64.68	63.28	61.81	60.90	60.80	59.86	66.78

Table 5. The impact of the number of pseudo-classes.

Pseudo-Class Number	Accuracy in each session (%)											Average Acc.
	0	1	2	3	4	5	6	7	8	9	10	
0	78.76	75.14	71.94	67.29	66.38	63.31	61.80	60.09	59.57	58.95	58.02	65.57
1	79.11	74.97	71.74	67.19	66.46	63.13	62.54	61.07	60.31	59.76	59.04	65.94
3	79.30	76.28	72.93	68.20	67.49	64.46	63.10	62.07	60.99	60.79	59.92	66.87
5	**79.63**	**76.73**	**73.72**	**69.42**	**68.29**	**65.33**	**63.48**	**62.57**	**61.82**	**61.39**	**60.40**	**67.53**
10	78.75	75.37	72.57	68.16	66.85	63.59	62.96	61.33	60.87	60.20	59.37	66.36

strong biases from those limited classes, resulting in low-entropy representations that provide weaker regularization. Increasing the number of contributing base classes enriches the visual variability and reduces class-specific dominance, thereby producing more discriminative and better-separated pseudo-classes.

However, further increasing the number beyond ten leads to a slight performance degradation. This is because excessive mixing can overly homogenize the pseudo-samples, blurring their structure and pushing them toward an average representation that no longer forms meaningful pseudo-class boundaries. These results collectively indicate that a moderate number of diverse base classes yields the most effective pseudo-samples, striking a balance between diversity and structural coherence in pseudo-class construction.

D.3. The number of pseudo-classes

The results in Table 5 indicate that introducing an appropriate number of pseudo-classes consistently improves performance, with the best results achieved when five pseudo-classes are used. This outcome reflects the role of pseudo-classes in DPCL. A moderate number of pseudo-classes provides a sufficiently rich set of high-entropy synthetic categories, which strengthens the model's ability to anticipate the variability of novel classes during incremental learning. When too few pseudo-classes are used, the diversity of the pseudo-class space remains limited and the model receives weaker regularization, which restricts its capability to maintain discriminative boundaries across sessions.

A further increase in the number of pseudo-classes leads to a performance drop. Excessive pseudo-classes gradually dilute the structural coherence of the pseudo-class space and create redundant or overly similar synthetic categories. These categories do not provide additional useful signals and instead introduce noise that interferes with the optimization of both base and incremental tasks. The results suggest that five pseudo-classes achieve the most suitable balance between diversity and stability in pseudo-class construction.

D.4. The effects of pseudo-label assignment

The results in Table 6 show that dynamic pseudo-label assignment achieves consistently better performance than both manual setting and similarity matching. Manual assignment forces each pseudo-class to follow a fixed label structure, which increases the optimization burden during the base task and restricts the model's ability to flexibly organize the pseudo-class space. Similarity matching performs even worse because instance-level matching causes the pseudo-samples to collapse toward a single virtual prototype as training progresses, which weakens class separability and narrows the pseudo-class distribution.

Dynamic assignment avoids these issues by matching pseudo-classes to virtual prototypes at the class level and by ensuring balanced usage of prototypes. This strategy preserves diversity among pseudo-classes and prevents pro-

Table 6. The impact of pseudo-label assignment methods.

Pseudo-Label Assignment	Accuracy in each session (%)											Average Acc.
	0	1	2	3	4	5	6	7	8	9	10	
Manual setting	79.46	75.97	73.08	68.57	67.91	64.82	63.08	61.91	60.99	60.88	59.82	66.95
Similarity matching	79.14	74.93	71.76	67.36	66.63	63.51	62.30	60.97	60.42	59.99	59.09	66.01
Dynamic assignment	**79.63**	**76.73**	**73.72**	**69.42**	**68.29**	**65.33**	**63.48**	**62.57**	**61.82**	**61.39**	**60.40**	**67.53**

Table 7. The impact of orthogonality on virtual prototypes

Orthogonality	Accuracy in each session (%)											Average Acc.
	0	1	2	3	4	5	6	7	8	9	10	
True	**79.63**	**76.73**	**73.72**	**69.42**	**68.29**	**65.33**	**63.48**	**62.57**	**61.82**	**61.39**	**60.40**	**67.53**
False	78.75	75.42	72.48	68.02	67.03	64.23	62.88	61.00	60.91	60.20	59.23	66.38

totype collapse while reducing interference with base-class learning. As a result, the model benefits from clearer pseudo-class boundaries and more stable regularization, which leads to stronger generalization in later incremental sessions.

D.5. The effects of virtual prototypes' orthogonality

Table 7 shows that enforcing orthogonality among virtual prototypes leads to consistently higher accuracy. When the prototypes remain orthogonal, each pseudo-class is guided toward a distinct and well-separated reference direction, which preserves the structural diversity required by DPCL and prevents different pseudo-classes from drifting into overlapping regions during assignment. Without orthogonality, the virtual prototypes become entangled in the embedding space, and the pseudo-classes lose clear separation, which weakens the effectiveness of pseudo-label learning and reduces incremental generalization.

References

[1] Noor Ahmed, Anna Kukleva, and Bernt Schiele. Orco: Towards better generalization via orthogonality and contrast for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28762–28771, 2024. 3, 4

[2] Zhixiang Chi, Li Gu, Huan Liu, Yang Wang, Yuanhao Yu, and Jin Tang. Metafscil: A meta-learning approach for few-shot class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14146–14155, 2022. 2, 5, 3, 4

[3] 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 (CVPR)*, pages 9057–9067, 2022. 2, 5, 3

[4] Do-Yeon Kim, Dong-Jun Han, Jun Seo, and Jaekyun Moon. Warping the space: Weight space rotation for class-incremental few-shot learning. In *The Eleventh International Conference on Learning Representations*, 2023. 3, 4

[5] Jiashuo Li, Songlin Dong, Yihong Gong, Yuhang He, and Xing Wei. Analogical learning-based few-shot class-incremental learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(7):5493–5504, 2024. 3, 4

[6] Linhao Li, Yongzhang Tan, Siyuan Yang, Hao Cheng, Yongfeng Dong, and Liang Yang. Adaptive decision boundary for few-shot class-incremental learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(17): 18359–18367, 2025. 3, 4

[7] Zicheng Pan, Xiaohan Yu, Miaohua Zhang, Weichuan Zhang, and Yongsheng Gao. Dycr: A dynamic clustering and recovering network for few-shot class-incremental learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):7116–7129, 2025. 3, 4

[8] 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 (CVPR)*, pages 24183–24192, 2023. 3, 4

[9] 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 (CVPR)*, pages 12180–12189, 2020. 2, 3, 4

[10] Qi-Wei Wang, Da-Wei Zhou, Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Few-shot class-incremental learning via training-free prototype calibration. In *Proceedings of the International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2023. 3, 4

[11] Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. Neural collapse inspired feature-

classifier alignment for few-shot class-incremental learning. In *ICLR*, 2023. 3, 4

[12] Jaehong Yoon, Sultan Madjid, Sung Ju Hwang, Chang-Dong Yoo, et al. On the soft-subnetwork for few-shot class incremental learning. In *The Eleventh International Conference on Learning Representations*, 2023. 3, 4

[13] 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 (CVPR)*, pages 12455–12464, 2021. 3, 4

[14] 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 (CVPR)*, pages 9046–9056, 2022. 2, 3, 4

[15] Huiping Zhuang, Zhenyu Weng, Run He, Zhiping Lin, and Ziqian Zeng. Gkeal: Gaussian kernel embedded analytic learning for few-shot class incremental task. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7746–7755, 2023. 3, 4

[16] Yixiong Zou, Shanghang Zhang, Yuhua Li, and Ruixuan Li. Margin-based few-shot class-incremental learning with class-level overfitting mitigation. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2022. 3, 4