A Appendix

You may include other additional sections here.

A.1 Clarifications of three scenarios

- **Task-Incremental Learning (TIL)**: This is a continual learning scenario where the model is informed about the task that needs to be performed in advance. In this scenario, the model can be trained with task-specific components as it knows what it's being asked to do. A typical architecture for a model in this scenario is a multi-headed output layer, meaning each task has its own output units, while the rest of the network may be shared between tasks. The goal is to incrementally improve on a series of tasks, learning each one without forgetting the previous tasks.
- **Domain-Incremental Learning (DIL)**: In this scenario, the model does not know the task identity at test time. However, it only needs to solve the task at hand, without necessarily identifying which task it is. The structure of tasks remains consistent, but the input distribution may vary. The model needs to adapt to these changes in the input distribution to successfully perform the task. A real-world example might be a model learning to adapt to different environments without explicitly identifying the environment.
- **Class-Incremental Learning (CIL)**: This is a complex learning scenario where the model not only needs to solve each task it has encountered so far but also must infer which task it is currently facing. In other words, it should be able to classify and learn new classes of objects incrementally. The model is required to maintain knowledge of previously learned classes while still being able to learn new ones. This scenario embodies many real-world learning problems where new classes or categories are continually encountered, and old ones should not be forgotten.

A.2 Results on different Memory sizes

To evaluate the performance of our proposed method under varying memory sizes, we conducted experiments by adjusting the size of the memory set and comparing the results with those obtained using other memory selection methods. The experiments were conducted using the imbalanced class-domain incremental scenario of PACS, and the results are presented in Tab. 1.

The experimental results showed consistent performance improvements for our proposed FDBS method across all memory sizes tested. Our method outperformed all other memory selection methods in each case, with the magnitude of the improvement being more pronounced for smaller memory sizes. Furthermore, our proposed FDBS method can be further strengthened by combining it with Contrastive Learning Loss (IWL) to improve its performance

Table 1: Comparison of different memory selection methods on Imb C-DIL PACS for three different memory sizes. We present the final accuracy as mean and standard deviation over five independent runs

	Memory size		
Methods	100	500	2000
ER	16.47±2.39	20.34±2.56	24.37±1.34
GSS	15.73±1.63	17.67±1.95	23.28±1.39
CBRS	17.24±2.15	21.15±2.17	25.61±1.84
OCS	19.35±1.87	23.43±2.28	26.87±1.36
FDBS(ours)	19.76±1.96	25.56±2.61	29.12±2.48
FDBS+IWL(ours)	**21.22**±1.48	**26.34**±1.86	**30.28**±0.93

A.3 The effectiveness of IWL

We combined Contrastive Learning Loss (IWL) with ER and CBRS to evaluate the effectiveness of our IWL. The experiments were conducted on Imbalanced C-DIL DomainNet and the Balanced CIFAR-100. The results are presented in Tab. 2.

Our study has demonstrated that Contrastive Learning Loss (IWL) can significantly enhance the performance of simple memory sample selection methods. Specifically, IWL is capable of optimizing the feature space, thereby enabling model better classifying. Additionally, we have observed that our selection method, FDBS, achieves the best results when used in combination with IWL.

Table 2: We combined Contrastive Learning Loss (IWL) with ER and CBRS to evaluate the effectiveness of our IWL. The experiments were conducted on Imbalanced C-DIL DomainNet and Balanced CIFAR-100. We set the memory size as 5000. The final accuracy was presented as the mean and standard deviation over five independent runs

Methods/Datasets	Balanced CIFAR-100	Imb C-DIL DomainNet
ER	18.26 ± 1.78	6.24 ± 0.62
ER+IWL	18.79 ± 1.32	8.34 ± 0.54
CBRS	18.55 ± 1.68	6.13 ± 0.59
CBRS+IWL	19.13 ± 1.16	9.21 ± 0.63
FDBS	19.89 ± 1.54	10.25 ± 0.94
FDBS+IWL	**21.13** ± 0.94	**11.46** ± 0.71

A.4 Study the influence of hyperparameters

In our memory selection method, FDBS, we introduce two crucial hyperparameters: σ within the RBF kernel (**??**) and τ as defined in (**??**). To assess the impact of these hyperparameters, we conducted experiments specifically on the Imbalanced Class-Domain Incremental Learning (Imb C-DIL) scenario of PACS. The results of these experiments are presented in Appendix A.4.

In our approach, both σ and τ play pivotal roles in evaluating the influence of a memory point on an input point, based on their respective distances. Generally, a larger value for these hyperparameters signifies that the influence diminishes more rapidly as the distance between points increases. Through our experimentation, we observed that our model exhibits a higher sensitivity to variations in the value of τ than σ.

τ	$\sigma = 0.5$
0.1	27.23 ± 1.89
0.5	**28.64** ± 1.44
1	27.58 ± 1.46
5	26.18 ± 1.23
10	24.89 ± 1.13

Table 3: σ fixed while varying τ

σ	$\tau = 0.5$
0.1	28.50 ± 1.65
0.5	**28.64** ± 1.44
1	28.34 ± 1.32
5	28.2 ± 1.35
10	27.49 ± 1.26

Table 4: τ fixed while varying σ

A.5 Collaborative Learning with other memory-based methods

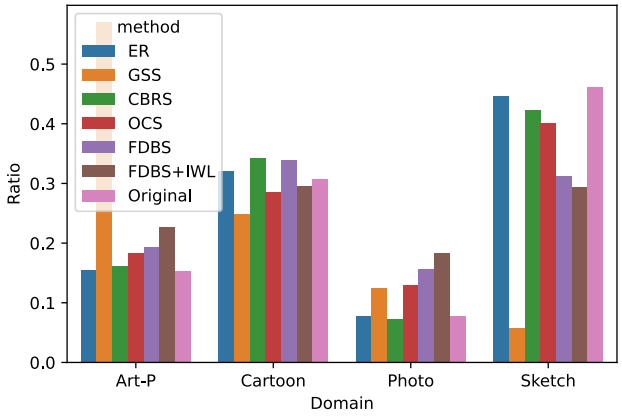
In our evaluation, we consider two notable continual learning methods, PodNet**?** and AFC**?**, both of which incorporate specialized distillation techniques reliant on a memory set. We integrate our Feature-Distance Based Sample Selection (FDBS) method to replace their original selection methods, which were either random or based on herding. Our experiments encompass two distinct scenarios: Balanced CIFAR-100 and the imbalanced Class-Domain Incremental Learning (imb C-DIL) of DomainNet. The results of these experiments are presented in Table Tab. 5. Remarkably, our memory selection method consistently enhances the performance of these continual learning methods both on balanced and imbalanced scenarios.

A.6 The distribution of our memory set

To gain deeper insights into the efficacy of our memory selection method, we examine the distribution of our memory set. Our experiments focus on the challenging task of imbalanced Domain-Incremental Learning using the PACS dataset, which comprises four distinct domains (e.g., photo, art painting, cartoon, and sketch). Following training, we analyze the distribution of our memory

Methods	Split-CIFAR100	Imb C-DIL DomainNet
PodNet	19.57 ± 1.48	8.75 ± 0.73
PodNet + FDBS(ours)	**20.93** ± 1.72	10.32± 0.82
AFC	19.43 ± 1.67	7.69 ± 0.64
AFC + FDBS(ours)	20.69 ± 1.54	**10.65**± 0.49

Table 5: Combining FDBS with Other Memory-Based Methods: Experiments on Balanced Split CIFAR-100 and Imbalanced Class-Domain Incremental Learning on DomainNet (Memory Size: 5000).The final accuracy was presented as the mean and standard deviation over five independent runs.



(a) Intra-Class Selection

Figure 1: The ratio of different domains within the memory set compared to the original scenario.

set, shedding light on how our method has shaped the representation of critical data points within this dynamic learning environment. The results of this analysis are presented in Tab. 6. And the ratio of different domain is shown in Fig. 1.

Methods such as ER and CBRS opt for random image selection, aiming to maintain a distribution akin to the original dataset. In contrast, our method prioritizes increasing intra-class diversity, thereby influencing a more balanced distribution of stored images. This approach plays a crucial role in improving the overall performance of continual learning. Additionally, the integration of our Contrastive Learning Loss (IWL) further enhances the feature space within our memory set. This refinement proves instrumental in effectively capturing images from minority domains, contributing to a more robust and balanced representation of data.

Methods /Domains	Photo	Art Painting	Cartoon	Sketch
Our Scenario	500	1000	2000	3000
ER	78	155	320	447
GSS	125	570	248	57
CBRS	73	162	342	423
OCS	130	183	286	401
FDBS(ours)	156	193	339	312
FDBS+IWL(Ours)	183	227	296	294

Table 6: Comparison of Memory Set Composition Across Methods in Imbalanced Domain-Incremental Learning (imb DIL) Scenario of PACS. We set the memory size as 1000.

A.7 Results on Balanced class-incremental learning scenario

To assess the effectiveness of our proposed approach in the context of classic balanced class-incremental learning, we conducted an experiment referred to **Cifar 100-B0** as detailed in **?**. In

this experiment, we partitioned the original Cifar 100 dataset into 10 and 20 distinct tasks, with each task encompassing a set of 5 distinct classes. The memory size is set as 2000. The result is presented in Tab. 7. Even in the classic class-incremental learning scenario, our proposed method can still significantly improve the previous state-of-the-art method.

Methods	10 steps	20 steps
iCaRL*[?]	65.27 ± 1.02	61.20 ± 0.83
BiC*[?]	68.80 ± 1.20	66.48 ± 0.32
PodNet*[?]	58.03 ± 1.27	53.97 ± 0.85
AFC[?]	61.25 ± 1.38	54.76 ± 0.79
WA*[?]	69.46 ± 0.29	67.33 ± 0.15
WA + **FDBS**(ours)	**71.35** ± 0.56	**70.18** ± 0.38
WA + **MSCL**(ours)	**73.71** ± 0.27	**72.34** ± 0.19

Table 7: Results for classic class-incremental learning on CIFAR-100. Results marked with '*' are obtained directly from **?**. The memory size is set to 2000.

A.8 ALGORITHM OF OUR METHOD

Algorithm 1 Train a batch at time step t

Input: F, g ,$\mathbb{S}^{mem}$, $\mathbb{S}_t^{str}$, b, K, $\boldsymbol{D}^{mem}$ as shown in **??**, $\mathbf{F}^{mem}$ stores the features of the memory set, N_b is the batch size.

for b steps **do**
  sample batch $I, \mathbf{X}^m, \boldsymbol{y}^m$ of size N_b from $\mathbb{S}^{mem}$ $\{I$: the index of the samples in $S^{mem}\}$
  $\mathbf{X}^{str}, \boldsymbol{y}^{str} = \mathbb{S}_t^{str}$
  $\mathbf{F}^m, \hat{\boldsymbol{y}}^m = g \circ F(\mathbf{X}^m)$
  $\mathbf{F}^{str}, \hat{\boldsymbol{y}}^{str} = g \circ F(\mathbf{X}^{str})$
  $\alpha = 0.1 + 0.9 * 0.99^t$
  Current Loss : $L_{cur} = \ell(\hat{\boldsymbol{y}}^{str}, \boldsymbol{y}^{str})$
  Replay Loss : $L_r = \ell(\hat{\boldsymbol{y}}^m, \boldsymbol{y}^m)$
  Update $\mathbf{F}^{mem}[I] = \mathbf{F}^m$
  Update $\boldsymbol{D}^{mem}[I] = dist(\mathbf{F}^m, \mathbf{F}^{mem})$
  Compute $\boldsymbol{a}$ based on **??**
  $\boldsymbol{D} = dist(\mathbf{F}^{str}, \mathbf{F}^{mem})$ as **??**
  Compute $\boldsymbol{w}$ based on **??** and **??**
  $L_{IWL} = L_{IWL}(\boldsymbol{w})$ as **??**
  Total Loss : $L = \alpha L_{cur} + (1 - \alpha)L_r + L_{IWL}$
  Update: F, g : Adam.step( )
  FDBS($\mathbb{S}^{mem}$,$\mathbb{S}_t^{str}$,$\boldsymbol{w}$,$\boldsymbol{D}$,M,K,$\boldsymbol{D}^{mem}$,$\mathbf{F}^{mem}$) as shown in Algorithm 2
end for

Algorithm 2 FDBS at time step t

Input: $\mathbb{S}^{mem}$, $\mathbb{S}_t^{str}$, $\boldsymbol{w}$, $\boldsymbol{D}$, M, K, $\boldsymbol{D}^{mem}$, $\mathbf{F}^{mem}$

$\mathbf{X}^{mem}, \boldsymbol{y}^{mem} = \mathbb{S}^{mem}$;
for each data $i, (x_i, y_i)$ in $\mathbb{S}_t^{str}$ **do**
  $K = K + 1$
  if $len(\mathbb{S}^{mem}) < M$ **then**
    store (x_i, y_i) in $\mathbb{S}^{mem}$
  else
    $p = \boldsymbol{w}_i * M/K$
    $r = random.rand()$
    if $r < p$ or $y_i \notin \mathbb{S}^{mem}$ **then**
      $c = most\_frequent(\boldsymbol{y}^{mem})$
      $I = index(\boldsymbol{y}^{mem} == c)$
      $k = random.choice(I)$
      $\mathbf{X}^{mem}[k], \boldsymbol{y}^{mem}[k] = x_i, y_i$;
      $\mathbf{F}^{mem}[k] = F(x_i)$
      $\boldsymbol{D}^{mem}[k] = \boldsymbol{D}[i, :]$
    else
      ignore (x_i, y_i)
    end if
  end if
end for