Supplementary Material for Dual Consolidation for Pre-Trained Model-Based Domain-Incremental Learning

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye(✉), Lijun Zhang, De-Chuan Zhan
School of Artificial Intelligence, Nanjing University
National Key Laboratory for Novel Software Technology, Nanjing University
{zhoudw,caizw,yehj,zhanglj,zhandc}@lamda.nju.edu.cn

Abstract

In the main paper, we present a method to prevent forgetting in domain-incremental learning through representation and classifier consolidation. The supplementary material provides additional details on the algorithm and experimental results mentioned in the main paper, along with extra empirical evaluations and discussions. The organization of the supplementary material is as follows:

- Section I provides the pseudocode of DUCT.
- Section II presents detailed experimental results, including the running time comparison, detailed performance on different task orders, and results with other pre-trained weights.
- Section III introduces the details about the datasets adopted in the main paper, including the number of tasks and images and differently-ordered sequences of tasks.
- Section IV introduces the compared methods adopted in the main paper.
- Section V presents a series of additional experiments to further validate the robustness of DUCT.

I. Algorithm Pipeline

Following the notation presented in the main body, we summarize the training steps in Algorithm 1.

First of all, we prepare the dataset and load the pre-trained model weights, then calculate the mean representation for each incoming class, as shown in Lines 2∼3. Next, we fine-tune the model and extract the task vector, which is then used to apply the model merging technique, as described in Line 5. After that, we keep the merged backbone frozen and retrain the classification layer to align it with the consolidated representation. Finally, we apply classifier transport to the classifiers to transfer semantic information, as outlined in Lines 7∼8.

†Correspondence to: Han-Jia Ye (yehj@lamda.nju.edu.cn)

Algorithm 1 DUCT for DIL

Input: Incremental datasets: $\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\}$,
Pre-trained embedding: $\phi_0(\mathbf{x})$;
Output: Incrementally trained model;

1: **for** $b = 1, 2 \cdots, B$ **do**
2: Get the incremental training set $\mathcal{D}^b$;
3: Extract class centers via Eq. 5;
4: Optimize the model via Eq. 2 ;
5: Consolidate representations via Eq. 7;
6: Retrain new classifiers via Eq. 8;
7: Solve OT via Eq. 9;
8: Consolidate old classifiers via Eq. 10;
return the updated model;

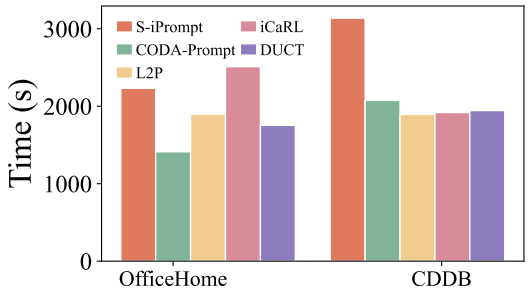
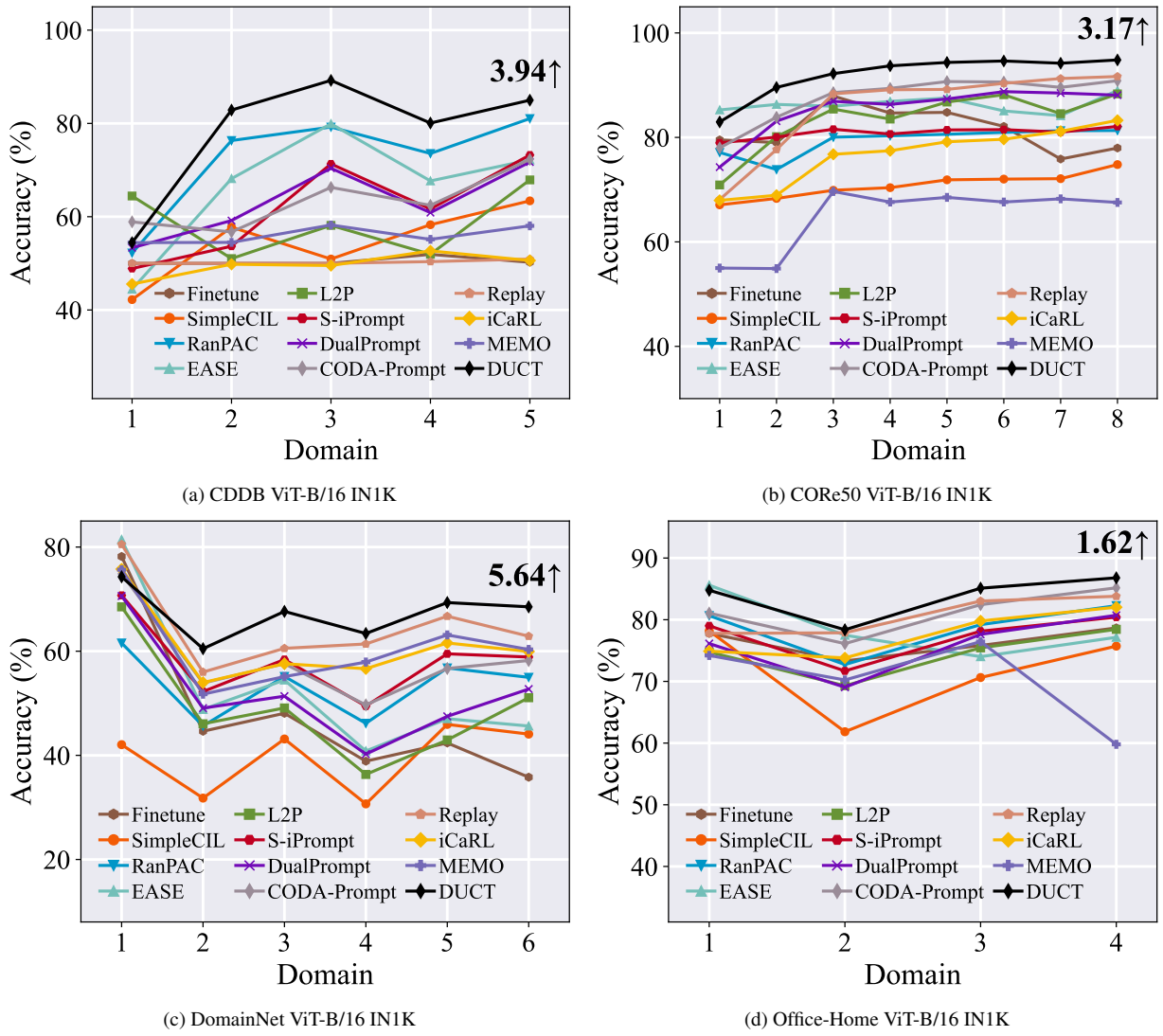


Figure 1. Running time comparison among different methods. DUCT **shows the best performance while having competitive training costs.**

II. Supplied Experimental Results

In this section, we supply additional experiments to show the effectiveness of DUCT, including the running time comparison, the detailed performance among different task orders, and more results with other pre-trained weights.



(a) CDDB ViT-B/16 IN1K

(b) CORe50 ViT-B/16 IN1K

(c) DomainNet ViT-B/16 IN1K

(d) Office-Home ViT-B/16 IN1K

Figure 2. Incremental performance of different methods with ViT-B/16 IN1K. We report the performance gap after the last incremental stage between DUCT and the runner-up method at the end of the line.

II.1. Running time comparison

In this section, we report the running time comparison among different compared methods. As shown in Figure 1, DUCT costs competitive running time against other compared methods while having the best performance.

II.2. Performance of different task orders

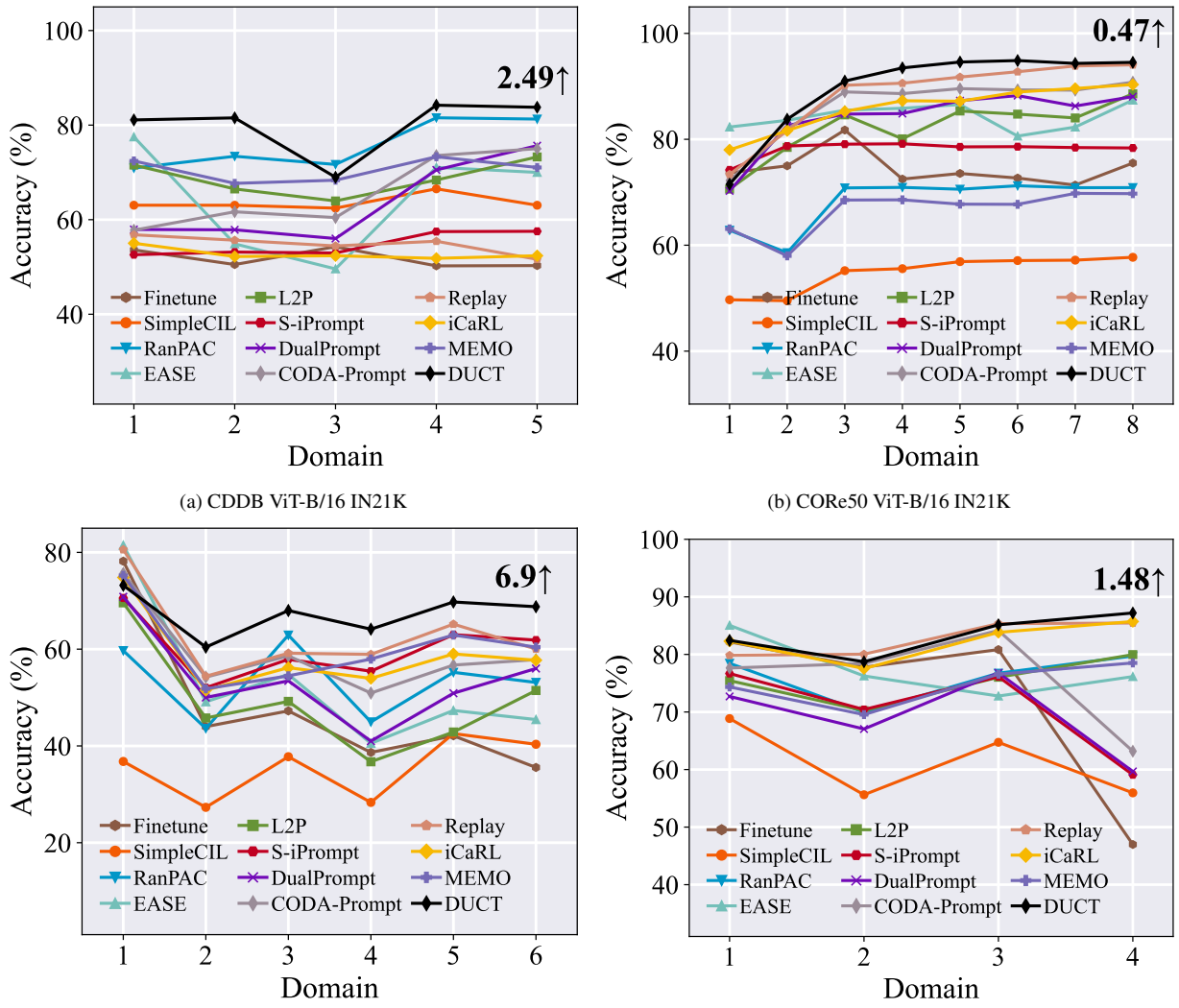
In the main paper, we conduct experiments on the benchmark datasets with five task orders and report the average performance. In this section, we report the performance on each order in Table 1, 2, 3, 4, 5. The task sequences are reported in Section III.2.

II.3. Different backbones

In the main paper, we mainly report the performance using ViT-B/16-IN1K. In this section, we present performance on the remaining benchmark in Figure 2 and all the experimental results on ViT-B/16-IN21K in Figure 3, demonstrating the model's robustness to changes in backbones.

III. Dataset Details

In this section, we introduce the details about datasets, including the dataset information (*i.e.*, the number of tasks and instances) and split information (*i.e.*, the five splits adopted in the main paper).



(a) CDDB ViT-B/16 IN21K

(b) CORe50 ViT-B/16 IN21K

(c) DomainNet ViT-B/16 IN21K

(d) Office-Home ViT-B/16 IN21K

Figure 3. Incremental performance of different methods with ViT-B/16 IN21K. We report the performance gap after the last incremental stage between DUCT and the runner-up method at the end of the line.

III.1. Dataset introduction

We report the details about benchmark datasets in Table 6, and they are listed as follows.

- **DomainNet** [8]* is a dataset of common objects in six different domains. All domains include 345 categories of objects, such as bracelets, planes, birds, and cellos. The domains include clipart — a collection of clipart images; real — photos and real-world images; sketch — sketches of specific objects; infograph — infographic images with specific objects; painting — artistic depictions of objects in the form of paintings, and quickdraw — drawings of the worldwide players of the game 'Quick Draw!'. We use the officially recommended version 'Cleaned' in this paper.

- **Office-Home** [12]† is a benchmark dataset for domain adaptation which contains four domains where each domain consists of 65 categories. The four domains are Art — artistic images in the form of sketches, paintings, ornamentation, etc.; Clipart — a collection of clipart images; Product — images of objects without a background; and Real_World — images of objects captured with a regular camera. It contains 15,500 images, with an average of around 70 images per class

*https://ai.bu.edu/M3SDA/

†https://hemanthdv.github.io/officehome-dataset/

Table 1. Average and last performance of different methods with the 1st task order in Section III.2. The best performance is shown in bold. All methods are implemented with ViT-B/16 IN1K. Methods with † indicate implementations with exemplars (10 per class).

Method	Office-Home		DomainNet		Core50		CDDB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	73.48	76.23	47.99	35.80	74.44	72.18	50.23	50.23
Replay†	80.62	83.80	64.68	62.88	85.71	91.66	50.26	50.91
iCaRL† [10]	77.63	82.03	60.25	59.05	76.79	81.60	49.63	49.77
MEMO† [17]	70.17	59.80	60.65	60.33	64.89	67.55	53.00	53.67
SimpleCIL [19]	71.60	75.72	39.61	44.08	70.81	74.80	54.52	63.40
L2P [15]	74.49	78.44	49.00	51.07	83.47	88.33	58.69	67.89
DualPrompt [14]	75.88	80.72	51.92	52.73	85.42	88.09	63.10	71.72
CODA-Prompt [11]	81.20	85.17	58.49	58.21	87.70	90.85	63.33	72.28
EASE [18]	78.56	77.15	53.04	45.63	86.28	88.98	66.47	72.18
RanPAC [7]	78.72	82.28	53.48	54.97	79.44	81.32	72.47	81.04
S-iPrompt [13]	77.32	80.42	58.21	58.88	80.91	82.09	61.75	73.23
DUCT	**83.76**	**86.79**	**67.28**	**68.52**	**92.06**	**94.83**	**78.32**	**84.98**

Table 2. Average and last performance of different methods with the 2rd task order in Section III.2. The best performance is shown in bold. All methods are implemented with ViT-B/16 IN1K. Methods with † indicate implementations with exemplars (10 per class).

Method	Office-Home		DomainNet		Core50		CDDB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	76.50	74.37	36.94	32.04	72.70	76.93	50.99	47.88
Replay†	**83.04**	83.48	59.22	62.27	85.90	93.36	56.52	62.45
iCaRL† [10]	81.48	81.90	53.75	57.91	75.54	81.17	61.08	76.31
MEMO† [17]	71.44	62.92	53.41	57.04	61.30	67.52	50.54	48.59
SimpleCIL [19]	68.45	75.72	38.18	44.08	71.63	74.80	63.19	63.40
L2P [15]	75.30	79.03	45.15	50.90	83.63	87.57	67.01	70.56
DualPrompt [14]	75.41	80.70	46.31	52.29	84.09	87.55	59.85	72.07
CODA-Prompt [11]	81.21	84.53	52.86	57.38	87.91	91.21	63.05	73.53
EASE [18]	75.96	74.76	47.31	44.62	86.32	86.69	64.08	69.19
RanPAC [7]	78.03	82.28	50.11	54.98	79.44	81.32	75.00	81.04
S-iPrompt [13]	78.27	80.81	54.51	60.06	82.09	83.72	61.46	72.23
DUCT	81.95	**86.90**	**62.04**	**68.17**	**91.70**	**93.99**	**80.72**	**84.91**

and a maximum of 99 images in a class.

- **CDDB-Hard** [5][‡]. As a dataset mixed with real-world and model-generated images, the continual deepfake detection benchmark (CDDB) aims to simulate real-world deepfakes' evolution. The authors put out three tracks for evaluation, 'EASY,' 'Hard,' and 'Long,' respectively. Following [13], we choose the 'Hard' track for evaluation since it poses a more challenging problem due to its complexity.

- **CORe50** [6][§]. Composed of 11 sessions characterized by different backgrounds and lighting, CORe50 is built for continual object recognition. Numerous RGB-D images are divided into 8 indoor sessions for training and 3 outdoor sessions for testing, and each session includes a sequence of about 300 frames for all 50 objects.

[‡]https://coral79.github.io/CDDB_web/

[§]https://vlomonaco.github.io/core50/index.html#dataset

III.2. Domain Sequences

In domain-incremental learning, different algorithms' performances may be influenced by the order of domains. Consequently, we randomly shuffle the domains and organize five domain orders in the main paper, which are further utilized for a holistic evaluation. The task orders are reported in Table 7, 8, 9, 10.

IV. Compared Methods

In this section, we introduce the methods that were compared in the main paper. **Note that we re-implement all methods using the same pre-trained model as initializa-**

Table 3. Average and last performance of different methods with the 3rd task order in Section III.2. The best performance is shown in bold. All methods are implemented with ViT-B/16 IN1K. Methods with † indicate implementations with exemplars (10 per class).

Method	Office-Home		DomainNet		Core50		CDDB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	81.82	74.93	44.84	29.45	77.35	79.12	51.68	52.81
Replay†	86.98	84.10	67.23	60.52	84.91	91.60	87.52	75.30
iCaRL† [10]	84.40	81.64	61.74	54.41	75.54	81.17	88.66	**86.05**
MEMO† [17]	75.50	63.23	62.97	56.99	68.42	71.78	53.94	52.55
SimpleCIL [19]	76.32	75.72	45.74	44.08	69.67	74.80	56.64	63.40
L2P [15]	81.94	79.57	52.16	45.05	84.21	88.24	67.87	67.87
DualPrompt [14]	82.45	80.79	54.01	49.28	84.91	88.39	66.26	71.19
CODA-Prompt [11]	63.05	73.53	60.91	56.08	88.58	91.20	68.77	76.26
EASE [18]	81.45	74.76	52.19	40.81	86.26	85.26	67.48	72.18
RanPAC [7]	83.84	82.28	57.57	54.08	78.07	81.62	78.46	81.04
S-iPrompt [13]	83.40	80.68	64.02	61.22	82.65	84.20	69.30	72.99
DUCT	**87.31**	**86.94**	**70.08**	**67.06**	**91.97**	**94.78**	**88.84**	85.84

Table 4. Average and last performance of different methods with the 4th task order in Section III.2. The best performance is shown in bold. All methods are implemented with ViT-B/16 IN1K. Methods with † indicate implementations with exemplars (10 per class).

Method	Office-Home		DomainNet		Core50		CDDB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	82.10	77.09	25.82	16.65	75.97	75.21	53.80	50.42
Replay†	86.38	82.67	**65.35**	60.26	85.89	92.23	89.91	77.27
iCaRL† [10]	83.38	78.71	58.30	51.55	68.25	72.91	**92.67**	**88.60**
MEMO† [17]	71.84	65.06	62.28	54.18	68.20	69.76	87.53	80.93
SimpleCIL [19]	81.31	75.72	40.06	44.08	71.73	74.80	66.56	63.40
L2P [15]	85.49	81.71	48.59	45.47	83.35	87.01	77.38	61.50
DualPrompt [14]	84.74	81.02	50.90	46.77	85.23	86.89	81.46	72.89
CODA-Prompt [11]	71.06	74.84	60.17	55.89	88.10	91.79	79.73	73.98
EASE [18]	84.69	74.76	48.28	42.92	86.30	86.69	70.23	50.48
RanPAC [7]	86.59	82.28	54.97	53.48	79.31	81.32	85.28	79.66
S-iPrompt [13]	85.52	80.19	61.32	60.89	82.11	83.50	81.28	73.06
DUCT	**89.79**	**86.98**	64.12	**64.70**	**92.12**	**94.56**	89.38	84.31

tion. They are listed as follows.

- **Finetune** is a simple baseline in DIL, which directly optimizes the model with cross-entropy loss. It will suffer catastrophic forgetting since there is no restriction on preserving previous knowledge.
- **Replay** [9] is an exemplar-based method, which saves a set of exemplars from previous domains (*i.e.*, in this paper, we save 10 exemplars per class) and replay them when learning new domains. Hence, forgetting can be alleviated since the model can revisit informative instances from previous domains when learning new ones. Of note, for classes with fewer than 10 instances, repeatable sampling is allowed to conform to the requirement.
- **iCaRL** [10] is a knowledge distillation-based continual learning algorithm, which saves the previous model in memory. During updating, apart from the cross-entropy loss for learning new tasks, it also introduces the knowledge distillation loss between old and new models to avoid forgetting. It also requires saving an increasing number of exemplars.
- **MEMO** [17] is an expansion-based continual learning algorithm that partially expands the network to catch new features. As for the implementation, we follow the original paper to decouple the network and expand the last transformer block for each new task.
- **SimpleCIL** [19] proposes this simple baseline in pre-trained model-based continual learning. It freezes the backbone representation, extracts the class center of each class, and utilizes a cosine classifier updated by assigning class centers to the classifier weights.

Table 5. Average and last performance of different methods with the 5th task order in Section III.2. The best performance is shown in bold. All methods are implemented with ViT-B/16 IN1K. Methods with † indicate implemented with exemplars (10 per class).

Method	Office-Home		DomainNet		Core50		CDDB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	77.69	78.18	38.49	26.9	76.73	77.52	53.40	49.21
Replay†	84.12	84.72	67.45	59.88	85.37	92.22	50.34	50.12
iCaRL† [10]	80.99	81.26	62.11	54.14	74.86	77.55	50.10	49.77
MEMO† [17]	66.95	64.24	70.29	63.49	61.18	64.61	53.36	50.35
SimpleCIL [19]	80.75	75.72	51.15	44.08	70.78	74.80	63.06	63.40
L2P [15]	81.37	81.38	57.33	51.13	83.21	88.22	65.68	54.42
DualPrompt [14]	82.55	81.00	57.33	48.12	82.99	83.48	71.00	69.18
CODA-Prompt [11]	63.33	72.28	66.85	57.41	87.33	92.81	71.06	74.84
EASE [18]	85.13	80.23	51.68	44.62	86.37	87.47	70.65	60.75
RanPAC [7]	84.31	82.28	61.48	54.98	79.52	81.32	83.39	79.64
S-iPrompt [13]	82.70	80.47	67.73	61.23	81.98	83.38	68.76	72.09
DUCT	**88.55**	**86.92**	**72.29**	**66.59**	**91.88**	**94.21**	**83.46**	**85.48**

Table 6. Details on domain size, train/test split, and instance number of the benchmark datasets. The dataset split and selection follows [11, 13–15].

	Domains	Size	Test set
CDDB-Hard	biggan	4.0k	standard splits 75%:25% ('san' - 80%:20%)
	gaugan	10.0k	
	san	440	
	whichfaceisreal	2.0k	
	wild	10.5k	
CORe50	s1	14.9k	s3, s7, s10 Indoor:Outdoor 8:3
	s2	14.9k	
	s4	14.9k	
	s5	14.9k	
	s6	14.9k	
	s8	14.9k	
	s9	14.9k	
	s11	14.9k	
DomainNet	clipart	48.1k	standard splits 70%:30%
	infograph	51.6k	
	painting	72.2k	
	quickdraw	172.5k	
	real	172.9k	
	sketch	69.1k	
Office-Home	Art	2.4k	random splits 70%:30%
	Clipart	4.3k	
	Product	4.4k	
	Real World	4.3k	

- **L2P** [15] is the first work introducing prompt tuning in continual learning. With the pre-trained weights frozen, it learns a prompt pool containing many prompts. During training and inference, instance-specific prompts are selected to produce the instance-specific embeddings. However, as alluded to before, learning new domains will lead to the overwriting of existing prompts, thus triggering forgetting.
- **DualPrompt** [14] extends L2P in two aspects. Apart from the prompt pool and prompt selection mecha-

Table 7. Task orders of CDDB-Hard.

CDDB-Hard	Task 1	Task 2	Task 3	Task 4	Task 5
Order 1	san	whichfaceisreal	biggan	wild	gaugan
Order 2	wild	whichfaceisreal	san	gaugan	biggan
Order 3	biggan	gaugan	wild	whichfaceisreal	san
Order 4	gaugan	biggan	wild	whichfaceisreal	san
Order 5	whichfaceisreal	san	gaugan	biggan	wild

Table 8. Task orders of CORe50.

CORe50	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Order 1	s11	s4	s2	s9	s1	s6	s5	s8
Order 2	s2	s9	s1	s6	s5	s8	s11	s4
Order 3	s4	s1	s9	s2	s5	s6	s8	s11
Order 4	s1	s9	s2	s5	s6	s8	s11	s4
Order 5	s9	s2	s5	s6	s8	s11	s4	s1

Table 9. Task orders of DomainNet.

DomainNet	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Order 1	clipart	infograph	painting	quickdraw	real	sketch
Order 2	infograph	painting	quickdraw	real	sketch	clipart
Order 3	painting	quickdraw	real	sketch	clipart	infograph
Order 4	quickdraw	real	sketch	clipart	infograph	painting
Order 5	real	quickdraw	painting	sketch	infograph	clipart

Table 10. Task orders of Office-Home.

Office-Home	Task 1	Task 2	Task 3	Task 4
Order 1	Art	Clipart	Product	Real_World
Order 2	Clipart	Product	Real_World	Art
Order 3	Product	Clipart	Real_World	Art
Order 4	Real_World	Product	Clipart	Art
Order 5	Art	Real_World	Product	Clipart

nism, it further introduces prompts instilled at different depths and task-specific prompts. During training and inference, the instance-specific and task-specific prompts work together to adjust the embeddings.

- **CODA-Prompt** [11] aims to avoid the prompt selection cost in L2P. It treats prompts in the prompt pool as bases and utilizes the attention results to combine multiple prompts as the instance-specific prompt.
- **EASE** [18] designs lightweight feature expansion technique with adapters to learn new features as data devolves. To fetch a classifier with the same dimension as ever-expanding features, it utilizes class-wise similarity to complete missing class prototypes.
- **RanPAC** [7] extends SimpleCIL by randomly projecting the features into the high-dimensional space and learning the online LDA classifier for final classification.
- **S-iPrompt** [13] is specially designed for pre-trained model-based domain-incremental learning. It learns task-specific prompts for each domain and saves domain centers in the memory with K-Means. During inference, it first forwards the features to select the nearest domain center via KNN search. Afterward, the selected prompt will be appended to the input.

V. More Experiments

To further demonstrate the effectiveness of DUCT in different scenarios, we conduct a series of more comprehensive experiments in this section.

Table 11. Incremental performances of methods on five representative pre-trained models, *i.e.*, ResNet101, ViT-S/16, ViT-B/16-IN1K, ViT-B/16-IN21K, and ViT-B/16-DINO, comparing the selected best-performing algorithms and DUCT. 'NA' indicates that these methods are incompatible with ResNet. DUCT **achieves the optimal performances across all backbones.**

	ResNet101	ViT-S/16	ViT-B/16-DINO	ViT-B/16-IN1K	ViT-B/16-IN21K
CODA-Prompt [11]	NA	52.90	55.47	58.49	58.99
EASE [18]	NA	40.84	38.72	53.04	53.11
S-iPrompt [13]	NA	55.12	60.12	58.21	60.13
DUCT	**50.22**	**64.55**	**62.89**	**67.28**	**67.37**

Table 12. The upper bound on the performance of ViT-B/16-IN21K for all benchmarks, *i.e.*, the performance achieved by fine-tuning on the union of the entire dataset.

	CDDB-Hard	Core50	DomainNet	Office-Home
Upperbound	93.44	96.29	76.28	87.92

V.1. More backbones

Apart from ViT-B/16-IN1K and ViT-B/16-IN21K, we also evaluate additional backbones at different scales, including ResNet101 [3], ViT-S/16, and ViT-B/16-DINO [1]. The results are reported in Table 11. Note that the compared methods are all based on prompt tuning technique [4] which is designed for Vision Transformers [2]. Hence, these compared methods are incompatible with ResNet structures, while DUCT can still boost ResNet in continual learning tasks, further highlighting that DUCT is a general and versatile framework. The outcomes demonstrate that DUCT outperforms all competitors across all PTMs.

V.2. Upper bound performance

To investigate the upper-bound performance of each dataset, we conduct joint training and present the results in Table 12. We follow [16] and adopt a learning rate of 0.0001 for the representation layer and 0.01 for the classification layer, with a batch size of 128. We report the optimal results in the table.

V.3. Class-Incremental Learning experiments

Since DUCT is a general framework, it can also be applied to the class-incremental learning (CIL) scenario. By equally splitting 345 classes of DomainNet into 5 incremental stages, we formulate a class-incremental learning setting, each containing 69 classes. We keep the other settings the same as in the main paper and report the results in Table 13. The results indicate that DUCT can also be applied to the class-incremental learning setting, which we shall explore in future work.

Table 13. Incremental performances of selected methods on CIL setting, evaluated on the most challenging benchmark dataset, DomainNet, using ViT-B/16-IN21K as the backbone. 'B0' indicates that classes are uniformly split across tasks. **The results indicate that** DUCT **performs effectively in CIL scenarios.**

Method	DomainNet B0 Inc69	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
CODA-Prompt [11]	72.50	71.05
EASE [18]	69.29	68.45
S-iPrompt [13]	73.79	71.81
DUCT	**75.29**	**74.68**

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9650–9660, 2021. 8

[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. In *ICLR*, 2020. 8

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 8

[4] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727. Springer, 2022. 8

[5] Chuqiao Li, Zhiwu Huang, Danda Pani Paudel, Yabin Wang, Mohamad Shahbazi, Xiaopeng Hong, and Luc Van Gool. A continual deepfake detection benchmark: Dataset, methods, and essentials. In *WACV*, pages 1339–1349, 2023. 4

[6] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pages 17–26. PMLR, 2017. 4

[7] Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, 2023. 4, 5, 6, 7

[8] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate

Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, pages 1406–1415, 2019. 3

[9] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990. 5

[10] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 4, 5, 6

[11] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, pages 11909–11919, 2023. 4, 5, 6, 7, 8

[12] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5018–5027, 2017. 3

[13] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *NeurIPS*, 35:5682–5695, 2022. 4, 5, 6, 7, 8

[14] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pages 631–648, 2022. 4, 5, 6

[15] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pages 139–149, 2022. 4, 5, 6

[16] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *ICCV*, pages 19148–19158, 2023. 8

[17] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023. 4, 5, 6

[18] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 2024. 4, 5, 6, 7, 8

[19] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 133:1012–1032, 2025. 4, 5, 6