Supplementary Materials: One-stage Prompt-based Continual Learning

Youngeun Kim, Yuhang Li, and Priyadarshini Panda

Yale University, CT, USA
{youngeun.kim,yuhang.li,priya.panda}@yale.edu

1 Experimental Details

We maintain the prompting configuration (*i.e.* lengths and locations) for L2P and DualPrompt as suggested by the CodaPrompt paper. The detailed configurations are as follows: L2P incorporates a prompt pool of size 20, a total prompt length of 20, and selects 5 prompts from the pool for inference. DualPrompt utilizes a length 5 prompt in layers $1 \sim 2$ and length 20 prompts in layers $3 \sim 5$. CodaPrompt uses a prompt length of 8 and 100 prompt components. For all PCL methods, including our OS-prompt, we employ the Adam optimizer [1] with β_1 = 0.9 and β_2 = 0.999, using a batch size of 128 images. We use cosine-decaying learning rate scheduling with learning rate $1e-3$. All images are resized to 224x224 and normalized. CIFAR-100 is trained for 20 epochs, and ImageNet-R for 50 epochs.

To train a prompt pool in our method, we freeze prompts from the previous stages ($1 \sim N-1$) and only train N stage's prompt. For example, for 10-task class incremental continual learning, we add 10 prompts to the prompt pool for each stages and only train newly added 10 prompts. To initialize the new prompts, we use Gram-Schmidt process [3] at the start of each new task, as described in CodaPrompt official Github implementation. Furthermore, in alignment with prior PCL implementations, we substitute the predictions from prior-task logits with negative infinity during the training phase of the current task. Note that we do not use the query attention proposed in CodaPrompt.

2 Does stronger ViT improve performance?

Setup. Here we provide the architecture details of advanced ViTs used in our work. We use pre-trained models from PyTorch Image Models (timm) [1].

• **Swin-Transformer-v2** [2] introduces three primary technical contributions to advance large-scale models in computer vision: firstly, a combined residual-post-norm method with cosine attention to enhance training stability; secondly, a log-spaced continuous position bias technique that allows seamless transfer from low-resolution pre-training to high-resolution downstream tasks; and thirdly,

[1] https://huggingface.co/timm

Table 1: Analysis on the impact of Reference ViT architecture on ImageNet-R.

Setting	Task-5		Task-10		Task-20	
Method (ViT Architecture)	$A_N(\uparrow)$	$F_N(\downarrow)$	$A_N(\uparrow)$	$F_N(\downarrow)$	$A_N(\uparrow)$	$F_N(\downarrow)$
OS-Prompt++ (ViT-B/16)	77.07 ± 0.15	2.23 ± 0.18	75.67 ± 0.40	1.27 ± 0.10	73.77 ± 0.19	0.79 ± 0.07
OS-Prompt++ (Swin-v2)	77.27 ± 0.21	2.24 ± 0.24	75.94 ± 0.28	1.29 ± 0.15	73.89 ± 0.38	0.72 ± 0.11
OS-Prompt++ (PoolFormer)	77.20 ± 0.50	2.19 ± 0.27	75.86 ± 0.40	1.42 ± 0.24	73.81 ± 0.64	0.73 ± 0.11
OS-Prompt++ (CaFormer)	77.16 ± 0.42	2.16 ± 0.24	75.91 ± 0.54	1.30 ± 0.12	73.83 ± 0.54	0.72 ± 0.05

Table 2: Analysis on the impact of ViT architecture on ImageNet-R.

Setting	Task-10	
Method (ViT Architecture)	$A_N(\uparrow)$	$F_N(\downarrow)$
L2P (ViT-B/16)	69.29 ± 0.73	2.03 ± 0.19
L2P (Swin-v2)	73.53 ± 0.41	1.36 ± 0.15
L2P (PoolFormer)	73.60 ± 0.34	1.42 ± 0.16
L2P (CaFormer)	73.63 ± 0.43	1.40 ± 0.10
DualPrompt (ViT-B/16)	71.32 ± 0.62	1.71 ± 0.24
DualPrompt (Swin-v2)	72.62 ± 0.58	1.12 ± 0.18
DualPrompt (PoolFormer)	72.61 ± 0.47	1.07 ± 0.07
DualPrompt (CaFormer)	72.83 ± 0.49	1.24 ± 0.14
CodaPrompt (ViT-B/16)	75.45 ± 0.56	1.64 ± 0.10
CodaPrompt (Swin-v2)	76.16 ± 0.36	1.25 ± 0.17
CodaPrompt (PoolFormer)	76.12 ± 0.10	1.08 ± 0.13
CodaPrompt (CaFormer)	76.55 ± 0.34	1.05 ± 0.05

SimMIM, a self-supervised pre-training method designed to curtail the reliance on vast quantities of labeled images. We use SwinV2-S model in our work.

• **PoolFormer** [5] adopts a fundamental approach by utilizing pooling as the token mixer in their model. This pooling operator, distinguished by its lack of trainable parameters, straightforwardly averages the features of neighboring tokens. We use PoolFormer-M36 model in our work.

• **CaFormer** [6] incorporates depthwise separable convolutions in the initial layers and employing standard self-attention in the later layers. The model establishes 85.5% accuracy on ImageNet-1K without any reliance on external data sources or the use of distillation techniques. We use CaFormer-M36 model in our work.

Experimental Results. The QR loss is an architecture-agnostic design with respect to the reference ViT because it only requires forwarding an image to obtain the $[CLS]$ token embedding. To amplify the effects of the QR loss, we experiment with cutting-edge ViT models such as Swin-Transformer-v2 [2], PoolFormer [5], and CaFormer [5]. For a fair comparison, we ensure that all these advanced architectures are trained on ImageNet-1k. As presented in Table 1, leveraging advanced reference ViT architectures improves accuracy by providing stronger representations. For example, compared to ViT-B/16 baseline, using Swin-v2 improves the A_N metric by 0.2%/0.27%/0.12% on Task-5/Task-10/Task-20, respectively. This result underscores the advantage of our method, emphasizing its adaptability with future advanced ViT architectures.

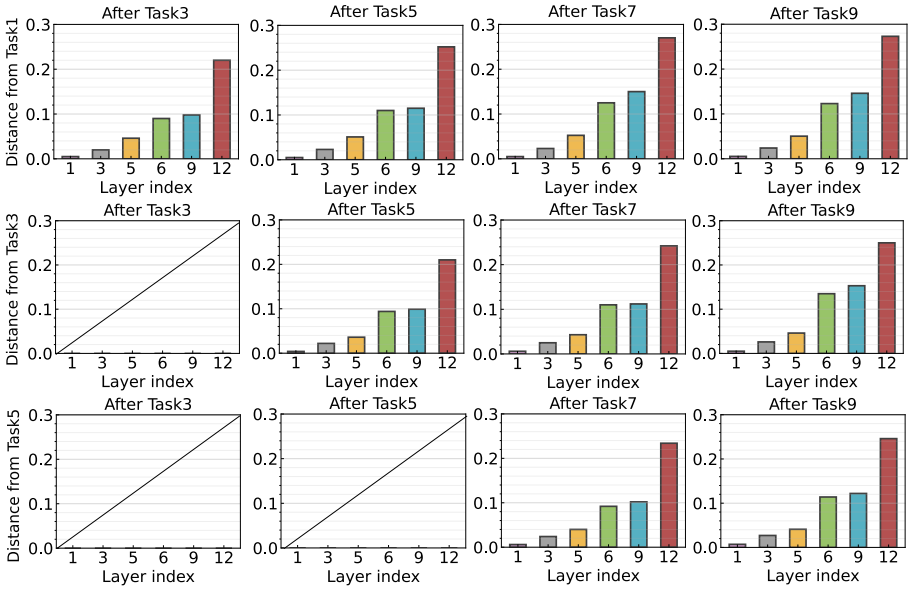

Fig. 1: We measure layer-wise feature distances between token embeddings of previous tasks when a new task is learned. Each column represents the embeddings after training a model on Tasks 3, 5, 7, and 9. Each row means the distance from the embeddings of Tasks 1, 3, and 5. For instance, the top-left figure represents the distance between token embeddings of Task 1 and those learned when Task 3 is completed. We train prompts using CodaPrompt [4] and use $1 - CosSim(x, y)$ to measure the layer-wise distance on the training dataset. We use a CIFAR100 10-task setting.

Moreover, we investigate the benefits of integrating advanced ViT architectures into previous PCL approaches. In Table 2, we provide the performance of various ViT architectures on 10-task ImageNet-R setting. L2P with ViT-B/16 yields an A_N of 69.29, whereas employing the Swin-v2, PoolFormer, and CaFormer architectures results in a notable performance improvement with A_N scores by $\sim$ 4.3%. For the DualPrompt method, the ViT-B/16 architecture provides an A_N of 71.32%. Transitioning to more advanced architectures improves the performance by $\sim$ 1.5%. Similarly, CodaPrompt shows performance improvement with Swin-v2, PoolFormer, and CaFormer architectures compared to ViT-B/16 architecture. In sum, the results suggest that regardless of the chosen method, more advanced architectures such as Swin-v2, PoolFormer, and CaFormer tend to outperform the ViT-B/16 configuration. Nevertheless, incorporating a more advanced query ViT does not alleviate computational cost; in fact, it may substantially increase computational cost. Conversely, while our OS-Prompt++ technique utilizes the strengths of advanced ViT architectures, we do not need to deploy the advanced ViT during inference, resulting in energy-efficiency.

3 How Stable are Token Embeddings across Continual Learning?

In Fig. 1, we provide more examples showing how stable are token embeddings across continual learning. We show how this change happens at each layer during continuous learning tasks. From our study, two main observations can be highlighted: (1) When we add prompts, there is a larger difference in the deeper layers. For example, layers $1 \sim 5$ have a small difference (≤ 0.1). However, the last layer shows a bigger change (≥ 0.1). (2) As learning continues, the earlier layers remain relatively stable, but the deeper layers change more. This observation concludes that although prompt tokens are included, the token embedding of early layers shows minor changes during training. Therefore, using token embeddings from these earlier layers would give a stable and consistent representation throughout the continual learning stages.

4 Algorithm

In this study, we introduce the algorithms of both the prior PCL and our OS-Prompt, detailed in Algorithm 1 and 2, respectively. As discussed in the primary text, earlier PCL methods deploy a query ViT to produce a prompt query, as seen in line 2 of Algorithm 1. With this generated prompt query, prompts are formed during the feedforward process, as illustrated in lines 3 to 5 of Algorithm 1. In contrast, our approach utilizes the intermediate [CLS] token as a prompt query, as indicated in line 4 of Algorithm 2.

Algorithm 1 Prior PCL works

Input image x, Query ViT $Q(\cdot)$, BackBone ViT $F(\cdot)$ consists of layer f_l, where $l \in \{1, ..., L\}$, prompt formation function $g()$, Prompt pool P_l

1: # feedforward step with image x
2: $q = Q(x)$ ▷ Compute a prompt query
3: **for** $l \leftarrow 1$ to 5 **do** ▷ Apply a prompt pool for the first five layers
4: $\phi_l \leftarrow g(q, P_l)$
5: $x_{l+1} \leftarrow f_l(x_l, \phi_l)$
6: **end for**
7: **for** $l \leftarrow 6$ to L **do** ▷ Standard ViT operation for leftover layers
8: $x_{l+1} \leftarrow f_l(x_l)$
9: **end for**
10: **return** $Classifier(x_{L+1})$

5 DomainNet Results

Table 3 provides results on 5-task DomainNet settings. The results show a similar trend with ImageNet-R and CIFAR-100 experiments.

Algorithm 2 OP-Prompt (Ours)

Input image x, BackBone ViT $F(\cdot)$ consists of layer f_l, where $l \in \{1, ..., L\}$, prompt formation function $g(\cdot)$, Prompt pool P_l

```
# feedforward step with image x
for l ← 1 to 5 do                          ▷ Apply a prompt pool for the first five layers
    q_l ← x_{l_[CLS]}                      ▷ Use [CLS] token as a prompt query
    φ_l ← g(q_l, P_l)
    x_{l+1} ← f_l(x_l, φ_l)
end for
for l ← 6 to L do                          ▷ Standard ViT operation for leftover layers
    x_{l+1} ← f_l(x_l)
end for
return Classifier(x_{L+1})
```

Table 3: Performance comparison on DomainNet 5-task setting.

Method	$A_N(\uparrow)$	$F_N(\downarrow)$
UB	79.65	-
FT	18.00 ± 0.26	43.55 ± 0.27
ER	58.32 ± 0.47	26.25 ± 0.24
Deep L2P	69.58 ± 0.39	2.25 ± 0.08
DualPrompt	70.73 ± 0.49	**2.03 ± 0.22**
CodaPrompt	73.24 ± 0.59	3.46 ± 0.09
OS-Prompt	72.24 ± 0.13	2.94 ± 0.02
OS-Prompt++	**73.32 ± 0.32**	2.07 ± 0.06

References

1. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 1
2. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12009–12019 (2022) 1, 2
3. Pursell, L., Trimble, S.: Gram-schmidt orthogonalization by gauss elimination. The American Mathematical Monthly **98**(6), 544–549 (1991) 1
4. Smith, J.S., Karlinsky, L., Gutta, V., Cascante-Bonilla, P., Kim, D., Arbelle, A., Panda, R., Feris, R., Kira, Z.: Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11909–11919 (2023) 3
5. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10819–10829 (2022) 2

6. Yu, W., Si, C., Zhou, P., Luo, M., Zhou, Y., Feng, J., Yan, S., Wang, X.: Metaformer baselines for vision. arXiv preprint arXiv:2210.13452 (2022) 2