arXiv:2303.07223v2 [cs.CV] 10 Jul 2024

PromptFusion: Decoupling Stability and Plasticity for Continual Learning

Haoran Chen[1], Zuxuan Wu[1,2†], Xintong Han[3],
Menglin Jia[4], and Yu-Gang Jiang[1,2]

[1]Shanghai Key Lab of Intell. Info. Processing, School of CS, Fudan University
[2]Shanghai Collaborative Innovation Center of Intelligent Visual Computing
[3]Huya Inc [4]Cornell Univeristy

Abstract. Current research on continual learning mainly focuses on relieving catastrophic forgetting, and most of their success is at the cost of limiting the performance of newly incoming tasks. Such a trade-off is referred to as the stability-plasticity dilemma and is a more general and challenging problem for continual learning. However, the inherent conflict between these two concepts makes it seemingly impossible to devise a satisfactory solution to both of them simultaneously. Therefore, we ask, "is it possible to divide them into two separate problems to conquer them independently?". To this end, we propose a prompt-tuning-based method termed PromptFusion to enable the decoupling of stability and plasticity. Specifically, PromptFusion consists of a carefully designed Stabilizer module that deals with catastrophic forgetting and a Booster module to learn new knowledge concurrently. Furthermore, to address the computational overhead brought by the additional architecture, we propose PromptFusion-Lite which improves PromptFusion by dynamically determining whether to activate both modules for each input image. Extensive experiments show that both PromptFusion and PromptFusion-Lite achieve promising results on popular continual learning datasets for class-incremental and domain-incremental settings. Especially on Split-Imagenet-R, one of the most challenging datasets for class-incremental learning, our method can exceed state-of-the-art prompt-based methods by more than 5% in accuracy, with PromptFusion-Lite using 14.8% less computational resources than PromptFusion. Code is available at https://github.com/HaoranChen/PromptFusion.

Keywords: Continual Learning · Prompt Tuning

1 Introduction

Despite great advances in deep learning, neural networks are often trained in a static supervised manner where all training data are available at once [13, 20, 37]. Continual learning [7, 12, 29], on the contrary, studies the behavior of neural networks under a more realistic scenario in which data arrive in a continual and

† Corresponding author.

dynamic procedure. Ideally, when confronted with new data, state-of-the-art models should be both stable to prevent performance degradation for previous tasks and plastic to learn sufficient information for the new task [48]. However, in practice, it is hard or even impossible to maintain such a balance, which is known as the stability-plasticity dilemma [1,32]. As a result, most of the current literature mainly focuses on one side of the problem, *i.e.*, relieving the problem of catastrophic forgetting, but overlooking the unseen sacrifice on the other side. A typical example is regularization-based approaches such as EWC [19]. In EWC, important parameters for previous tasks are kept intact which inevitably limits the capacity of learning new knowledge.

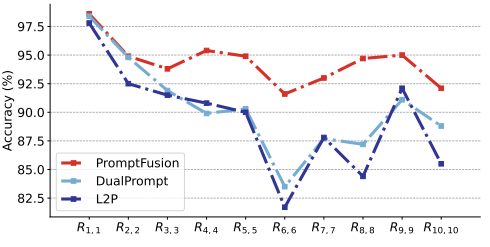

Fig. 1: Performance on the most recently learned task on the Split-Cifar100 dataset. Here, $R_{T,i}$ is the test accuracy on task i after learning task T. As is shown, the plasticity of L2P [46] and DualPrompt [45] is limited.

Since directly balancing this trade-off is extremely challenging, in this work, we instead seek to address the issue from a different perspective. Specifically, inspired by the Complementary Learning System [21,31], a biology theory that suggests intelligent agents must possess two learning systems, we hypothesize that the dilemma can be resolved similarly by leveraging two different architectures to tackle stability and plasticity independently. However, the incorporation of an additional architecture raises computational concerns as it might severely complicate the optimization process. So now the question becomes, is there a computationally efficient way to implement this idea?

Recently, prompt tuning [22,25,47] has become an emerging trend of fine-tuning models on downstream tasks in a computationally efficient manner. Since prompt tuning only trains an additional small set of parameters and freezes the pretrained backbone, it gives a potential solution to solve the above question. In fact, due to its strong transferring ability, prompt tuning has already been explored in continual learning [44–46]. The first work to do so trains a prompt pool followed by optimizing a prompt selection process for each task and achieves remarkable performances. Consequently, various follow-up approaches are introduced. However, as depicted in Fig 1, their plasticity is severely limited.

To this end, we propose PromptFusion, a simple yet effective framework that decouples stability and plasticity for continual learning using a Stabilizer module and a Booster module. Specifically, we instantiate the Stabilizer with CoOp [54] and the Booster with VPT [17], two mainstream prompt tuning methods. For the Stabilizer, PromptFusion initializes a new set of prompts for each incoming task and concatenates it to previously learned ones. During training, prompts from past tasks are frozen and only the current task-specific set is trained. As for the Booster, PromptFusion trains the same set of prompts for all tasks. Therefore, since only the newly added prompts of the Stabilizer are involved in training,

historical information can be well preserved (*stability*). In contrast, all prompts of the Booster are continuously updated, thereby allowing full learning of new information (*plasticity*). As a result, our design makes the Stabilizer suitable for stability and the Booster suitable for plasticity, achieving the best of both worlds.

To justify such an approach, a pilot study is carried out to empirically analyze the continual learning ability of both modules. Surprisingly, we find that in addition to aligning with our intention, their performance diverges on different datasets. Specifically, the Stabilizer is discovered to be much more robust to intra-class variations and is thus more suitable for the domain-incremental setting and complex datasets such as ImageNet-R, while the Booster is proficient on simple ones such as Cifar100. In light of such variation, to make full use of both modules, we train a weight parameter λ conditioned on the training dataset for the ensemble of their output logits. Following most continual learning approaches [4,36], we also apply a learnable weight mask to the final output logits for accommodating the imbalanced class distribution under both rehearsal and rehearsal-free settings (details in the following section). Extensive experiments on both class-incremental and domain-incremental datasets show that our proposed method achieves state-of-the-art results.

Nevertheless, while PromptFusion results in impressive performances, naively using both modules simultaneously would inevitably increase computational costs. Since existing works have already demonstrated the capability of using either the Stabilizer or the Booster in continual learning tasks [44–46], we wonder if we can use one of them as a base module that runs by default, and have the model adaptively decides on-the-fly whether to use the other one on a per-input basis. To this end, we further propose a computationally efficient framework PromptFusion-Lite that empirically chooses the Stabilizer as the default module and applies the Gumbel-Softmax trick [16] to determine whether or not to activate the Booster conditioned on inputs. Surprisingly, PromptFusion-Lite can still achieve competitive performance on continual learning benchmarks, while effectively dropping the overall computational overhead. In summary, our contributions are three-fold:

- We introduce PromptFusion and PromptFusion-Lite to address the stability-plasticity dilemma of continual learning in a computationally efficient manner. They take advantage of two modules, the Stabilizer and the Booster, which decouple stability and plasticity into two independent problems. PromptFusion naively uses both modules simultaneously, while PromptFusion-Lite adaptively selects the appropriate modules for each input.
- We conduct a detailed analysis of the Stabilizer and the Booster modules in the continual learning setting and discover that their performance diverges on different datasets. In particular, the Stabilizer is robust to intra-class variations and thus performs better with complex datasets in domain-incremental learning settings, while the Booster performs well only in class-incremental learning settings.
- Both our proposed PromptFusion and PromptFusion-Lite achieve state-of-the-art results on several popular benchmarks for class-incremental and domain-

incremental learning. Specifically, on Imagenet-R, the most challenging class-incremental learning benchmark, PromptFusion achieves an average accuracy of 78.7% in the memory-free condition, surpassing the state-of-the-art method CODA-Prompt by 5.3%. PromptFusion-Lite can achieve a competing accuracy of 75.6% with 14.8% less computational overhead.

2 Related Work

2.1 Continual Learning

In continual learning, the principle is to train a model on several tasks sequentially without forgetting knowledge from previously learned ones. There are three fundamental settings in literature for continual learning, namely task incremental [7, 19, 36], class incremental [9, 51], and domain incremental [10, 42]. For task incremental learning, the task identity for the input sample is provided at test time and therefore is regarded as the most relaxed condition. On the contrary, class incremental and domain incremental learning treat all samples the same during inference without prior knowledge about the task identity. The difference between these two is that in class incremental learning, data for each task are generally from the same distribution but belong to different categories. However, in domain incremental learning, data for each task belong to different distributions but have the same class labels.

Throughout the years, numerous efforts have been devoted to tackling continual learning, which can be mainly categorized into three groups: rehearsal-based methods [4, 14, 26, 36, 39] where memory is utilized to store samples from the past, architecture-based methods [9, 15, 30, 33, 38, 43, 51] where the network expands for an incoming task, and regularization based methods [2, 8, 19, 24, 52] where important parameters for previous tasks remain unchanged. However, most of these approaches mainly focus on the problem of catastrophic forgetting without considering the performance of learning the new task.

2.2 Prompt Learning

Recently, researchers in NLP have shown that learned large-scale language models can handle a wide range of downstream tasks with only a few or even no samples by prepending instructions to the input text [22, 23, 25]. Such instruction texts are called prompts. Consequently, prompts can be tuned instead of the weights of the entire network for a more efficient adaptation to downstream tasks. The success of prompt learning in NLP has also garnered attention in the vision community that motivates the establishment of many related methods [17, 18, 27, 28, 50, 54]. For example, DAPL [11] applies prompt learning in unsupervised domain adaptation by training a prompt for each source-target domain pair. MPA [6] extends it by adapting to the multi-source scenario through a two-stage alignment process. In the context of continual learning, a series of prompt-based works has achieved tremendous success. In L2P [46] and DualPrompt [45], a prompt pool is trained

such that for each task, the model samples task-specific prompts from it using a key-value selection process. However, they cannot be trained end-to-end, as the keys are optimized locally. Furthermore, they assume that every data in the mini-batch uses the same set of prompts, which is problematic during inference as data from different tasks might be present for the same mini-batch. Similarly, S-Prompt [44] is proposed in a similar fashion built on top of CoOp. It is, however, specifically designed for domain-incremental learning.

3 Preliminary

Since we instantiate PromptFusion with CoOp and VPT, two types of prompt learning approaches, we first briefly review the two methods, followed by presenting in detail how they are leveraged in the Stabilizer and the Booster modules.

CoOp CoOp [54] is a large-scale vision-language representation learning model built on top of CLIP [35]. It consists of an image encoder f and a text encoder g that aligns input images with text prompts. Unlike CLIP where the prompts are usually in the form of "a photo of a [CLS]", prompts in CoOp are trainable token parameters V_i of length M in the form of "$[V_1]...[V_M]$[CLS]". Given an image $\boldsymbol{x}$ with its label y and text prompt $\boldsymbol{P}_k$ for class k, CoOp first maps them to the same embedding space. Then they are aligned in a contrastive manner such that:

$$p(y=k|\boldsymbol{x}) = \frac{\exp(< g(\boldsymbol{P}_k), f(\boldsymbol{x}) > /T)}{\sum_{i=1}^{K} \exp(< g(\boldsymbol{P}_i), f(\boldsymbol{x}) > /T)} \tag{1}$$

is maximized when the input image $\boldsymbol{x}$ belongs to class k. Here, K is the total number of classes, $< \cdot, \cdot >$ denotes the cosine similarity, and T is a temperature parameter. During inference, the predicted label of a test image $\boldsymbol{x}$ is:

$$\arg\max_{k} < g(\boldsymbol{P}_k), f(\boldsymbol{x}) >, k \in \{1, ..., K\}. \tag{2}$$

VPT In contrast to CoOp, where the model seeks alignment from two modalities, VPT only relies on a Vision Transformer (ViT) that leverages prompt learning in a pure vision fashion [17]. In VPT, an input image is first divided into m fixed-sized patches I_i. With a class token, the input is first embedded into a latent space with positional embedding. Then, learnable prompt tokens U_i of length p are attached to the input in the form "[CLS]$[U_1]...[U_p][I_1]...[I_m]$". In shallow VPT, prompts are only inserted to inputs for the first Transformer layer, while for deep VPT, prompts are inserted for every layer. In this work, shallow VPT is adopted.

Prompt Tuning for Continual Learning In the present study, we focus on both the class incremental and the domain incremental setting. Formally, given N tasks $\mathcal{T} = (T_1, T_2, ..., T_N)$ where data for task T_t is denoted as $\mathcal{D}(\mathcal{X}^t, \mathcal{Y}^t)$, in class incremental scenarios, $\mathcal{Y}^i \cap \mathcal{Y}^j = \emptyset$ with $P(\mathcal{X}^i) = P(\mathcal{X}^j)$, while for domain incremental scenarios, $\mathcal{Y}^i = \mathcal{Y}^j$ with $P(\mathcal{X}^i) \neq P(\mathcal{X}^j)$, $i, j \in \{1, ..., N\}$, and $i \neq j$. Here $P(\cdot)$ denotes the probability density function. Since our method incorporates the usage of rehearsal memories, during each training phase, the model has access

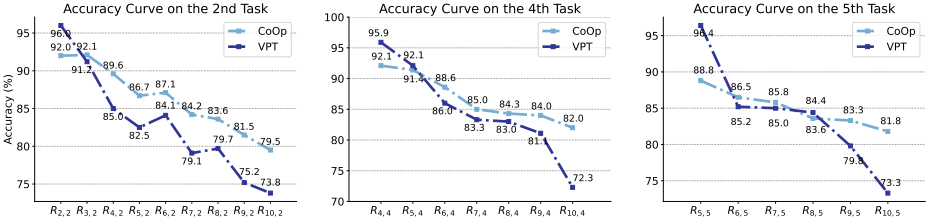

Fig. 2: Stability comparison between CoOp and VPT. Accuracy curves for tasks T_2, T_4, and T_5 using the two modules are presented. All three graphs show a similar trend where performance degradation in CoOp is much smaller than that in VPT. This shows that CoOp is much more robust against forgetting than VPT.

to both data from the current task and a few stored past samples, and the goal is to continuously update the model without forgetting past knowledge (*stability*) while achieving good results for the current one (*plasticity*).

One difference between VPT and CoOp is that prompts in CoOp are class-dependent while VPT has no such restrictions. Therefore, for incoming new classes, VPT is allowed to reuse the same set of prompts while CoOp, on the other hand, must learn new ones. As a result, for each task T_t, a new set of text prompts $\boldsymbol{P}_t^{stab} \in \mathbb{R}^{\frac{K}{N} \times M \times e^{stab}}$ is initialized for CoOp, where $\frac{K}{N}$ is the number of classes in each task and e^{stab} is the embedding dimension. If $t > 1$, $\boldsymbol{P}_t^{stab}$ is concatenated with previously learned prompts $\boldsymbol{P}^{stab} = \text{Concat}[\boldsymbol{P}_1^{stab}, ..., \boldsymbol{P}_t^{stab}]$ for alignment with image features in the embedding space. Note that $\boldsymbol{P}_1^{stab}, ..., \boldsymbol{P}_{t-1}^{stab}$ is kept frozen. As for VPT, another set of prompts $\boldsymbol{P}^{boost} \in \mathbb{R}^{p \times e^{boost}}$ is initialized before the training of the first task T_1 and constantly updated for each task T_t. While we could have trained a prompt for each task using VPT and concatenated it with previously learned ones as well, experiments show that such an approach results in poor performance. Thus, we stick to the above-stated design.

4 Pilot Study

In this section, we conduct a pilot study exploring whether stability and plasticity can be decoupled by leveraging the two proposed modules, the Stabilizer and the Booster. To this end, we systematically analyze the performance of CoOp and VPT in the setting of continual learning.

Decoupling Stability and Plasticity We begin the analysis by justifying our statement that our design of CoOp is suitable for stability while VPT is suitable for plasticity. Figure 2 presents three accuracy curves for task T_2, T_4 and T_5 on the Split-Cifar100 dataset. As shown, CoOp suffers much less from forgetting with an average performance drop of 9.9%; whereas the

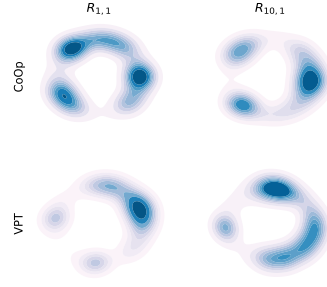

Fig. 3: KDE analysis on task T_1.

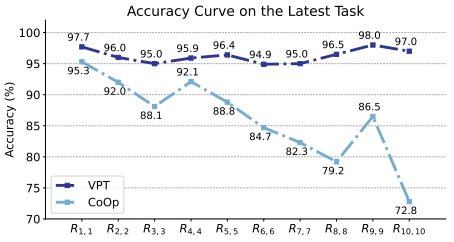

Fig. 4: Plasticity comparison between CoOp and VPT, where they exhibit opposite patterns.

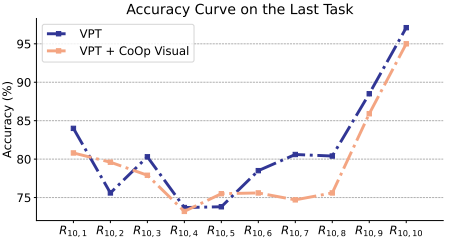

Fig. 5: Comparison between VPT and VPT with weights from CoOp's visual encoder, *i.e.*, CLIP.

average drop for VPT reaches 23.0%, indicating that the stability for VPT is limited. We also plot the change in feature distribution for task T_1 in Figure 3 with Gaussian Kernel Density Estimation (KDE). KDE is a popular non-parametric method for estimating the probability density function based on kernels as weights. As is shown in the graph, the distribution change for CoOp compared to VPT is much smaller, which coincides with results in Figure 2. Figure 4, on the other hand, depicts the performance for each newly learned task. While accuracy for VPT consistently achieves around 95%, CoOp exhibits a clear pattern of deterioration. Their margin in performance reaches up to 24.8% at T_{10} and could get worse when more tasks follow. While the evidence from both figures supports our hypothesis, an additional crucial piece is required to complete the entire picture: CoOp and VPT differ in their backbone networks.

Prompt or Backbone? To test whether patterns in Figure 2 and Figure 4 are indeed from different designs of prompts, an additional set of experiments is conducted where we apply pre-trained weights of the image encoder of CoOp to VPT. As is shown in Figure 5, experimental results demonstrate that their performances remain approximately the same. They both demonstrate a preference for plasticity over stability, as the performance on the last task T_{10} is high. Therefore, we conclude that the effect of the backbone network weight on the patterns we discovered is neglectable.

Furthermore, another interesting finding from Figure 5 is that using the backbone weights from the visual encoder of CoOp, i.e., CLIP [35], results in a worse performance. This is surprising as CLIP is generally considered more powerful than ImageNet pre-trained ViT. As a matter of fact, [45] reports a similar trend in performance drop when switching state-of-the-art continual learning methods with a stronger backbone, suggesting that a stronger network trained in a fully supervised manner does not necessarily lead to better continual learning ability.

Divergent Performance Meanwhile, while testing both modules on different types of datasets, we find out that CoOp is more advanced on complex datasets especially when large intra-class variations exist. VPT, on the other hand, handles simpler ones better. To justify this finding, a t-SNE visualization on Split-Cifar100, Split-Imagenet-R, and Core50 is presented in Figure 6. Here, Split-Cifar100 is

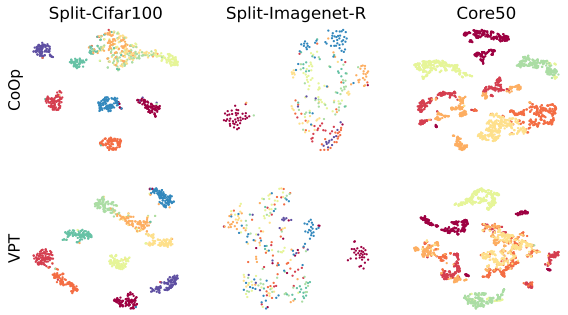

Fig. 6: t-SNE visualization on Split-Cifar100, Split-Imagenet-R, and Core50.

a relatively simple dataset, while Split-Imagenet-R and Core50 incorporate covariate shifts in the data distribution. As depicted in Figure 6, CoOp exhibits a more clustered feature than VPT on Imagenet-R and Core-50, suggesting that it is more robust to intra-class variations. Alternatively, on Cifar100, the feature obtained from VPT is more clustered. Therefore, by utilizing the two modules, another potential benefit is that the resulting model can accommodate the varying characteristics of different datasets.

5 Method

PromptFusion with memory With results from the pilot study, we confirm that stability and plasticity can be decoupled using the proposed Stabilizer module and the Booster module. Based on this observation, we first present PromptFusion. The overview of PromptFusion is given in Figure 7. Formally, denote the Stabilizer model as S and the Booster model as B. Given an input image $\boldsymbol{x_i}$ with label y_i, $\boldsymbol{x_i}$ is first passed to the two modules S and B. Their respective output $S(\boldsymbol{x_i})$ and $B(\boldsymbol{x_i})$ are then fused by the trainable parameter λ through a weighted average. Due to using a memory buffer, the result is further element-wise multiplied by a weight mask $\boldsymbol{W}$ to balance old and new classes. Specifically, we would like old classes to be rectified and new classes to be weakened. Therefore, we divide the learning of $\boldsymbol{W}$ into two matrices, α and β such that:

$$\boldsymbol{W} = \text{Concat}[\frac{\theta}{\sigma(\beta)}, \sigma(\alpha)], \theta = \frac{T_i}{2}, \tag{3}$$

where $\sigma(\cdot) = \frac{1}{1+e^{-x}}$ is the sigmoid function and θ is a weight parameter scaled by the current task id T_i. Putting them together, the final output $\boldsymbol{z}_i$ is derived as:

$$\boldsymbol{z}_i = \boldsymbol{W} \odot [(1 - \sigma(\lambda))S(\boldsymbol{x}_i) + \sigma(\lambda)B(\boldsymbol{x_i})], \tag{4}$$

where $\odot$ indicates the element-wise multiplication operation. Further, following the idea in [3] that suggests CLIP can be enhanced with image prompts, we apply a similar approach to augment CoOp by inserting prompts $\tilde{\boldsymbol{P}}^{stab}$ in the

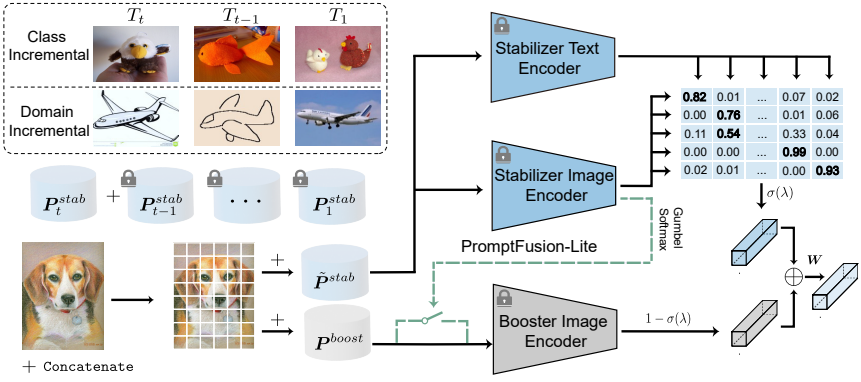

Fig. 7: An overview of PromptFusion and PromptFusion-Lite. PromptFusion consists of two architectures, the Stabilizer, and the Booster; PromptFusion-Lite has an additional decision module that adaptively determines whether to use the Booster on a per-input basis. Note that both our proposed methods work in class incremental and domain incremental settings.

image patches. Finally, with CE referring to the cross-entropy loss, the learning objective is:

$$\min_{\boldsymbol{\Theta}} \sum_{i=1} \texttt{CE}(\boldsymbol{z}_i, y_i), \boldsymbol{\Theta} := \{\alpha, \beta, \boldsymbol{P}^{stab}, \tilde{\boldsymbol{P}}^{stab}, \boldsymbol{P}^{boost}\}. \tag{5}$$

PromptFusion without memory To align with state-of-the-art prompt-based methods that achieve promising results under the memory-free setting, we refer to the rehearsal approach [53] that generates old task statistics when training on the current task. Specifically, when learning each new task, we record the mean μ_k and covariance Σ_k of the feature for each class k and model the class feature as a Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$. Then, when passing on to the next task, we first sample features from the distribution, and further finetune PromptFusion on the generated set. Since the set is composed of both features from the current and past tasks, it is relatively balanced. Therefore, the weight mask $\boldsymbol{W}$ is also accommodated accordingly:

$$\boldsymbol{W} = \text{Concat}[\frac{1}{\sigma(\beta)}, \sigma(\alpha)] \tag{6}$$

PromptFusion-Lite While we can use both modules simultaneously and naively weighted sum their output, doing so would inevitably increase the computational cost. Based on existing works that demonstrate the efficacy of the Stabilizer in continual learning, we seek an alternative approach, where we empirically run the Stabilizer by default, and the use of the Booster is determined on-the-fly conditioned on inputs. It is worth pointing out that deriving on/off binary decisions is non-trivial, as it is non-differentiable. To mitigate this issue, we use a straight-through Gumbel-Softmax estimator [16]. Specifically, for each input

image, we compute a decision entry as:

$$M_{i,k}(v_i) = \frac{\texttt{exp}(\texttt{log}(\boldsymbol{F}(v_i)_k + G_{i,k})/\tau)}{\sum_{j=1}^{K} \texttt{exp}(\texttt{log}(\boldsymbol{F}(v_i)_j + G_{i,j})/\tau)}, \tag{7}$$

where $K = 2$ is the number of categories, $G_i = -\log(-\log(U_i))$ is the Gumbel distribution in which $U_i \sim U(0, 1)$, v_i is the feature vector of the input image obtained from the image encoder of CLIP, $\boldsymbol{F}(\cdot)$ is an MLP consisting of two linear layers with the `tanh` activation function that transforms the feature vector to a 2-dimensional output, and τ is a temperature parameter. Without loss of generality, we set $M_{i,0} = 1$ to indicate using the Booster and $M_{i,1} = 1$ to indicate the opposite, such that the input x_i is masked accordingly when passing to B. In contrast to Eqn 5, the learning objective for PromptFusion-Lite is now:

$$\begin{gathered} \min_{\boldsymbol{\Theta}} \sum_{i=1} \texttt{CE}(\boldsymbol{z}_i, y_i) + \zeta(\sum_{i=1} M_{i,0} - \gamma)^2 + \delta \texttt{KD}(\boldsymbol{F}, \boldsymbol{F}'), \\ \boldsymbol{\Theta} := \{\alpha, \beta, \boldsymbol{P}^{stab}, \tilde{\boldsymbol{P}}^{stab}, \boldsymbol{P}^{boost}, \boldsymbol{F}\}, \end{gathered} \tag{8}$$

where the second term refers to a usage penalty that limits the activation of the Booster and the last term refers to the knowledge distillation of $\boldsymbol{F}$ before and after training on the new task, intending to mitigate its forgetting. Here $\boldsymbol{F}'$ refers to a frozen copy of $\boldsymbol{F}$ after finishing training on the current task. γ is a predefined target for the fraction of times the Booster is expected to use and ζ and δ are weight parameters controlling the contribution to the total loss. The full picture of the algorithm is given in the appendix.

6 Experiments

6.1 Experimental Setup

Datasets Experiments are conducted on four popular benchmark datasets of continual learning, namely Cifar100, CUB200, Imagenet-R, and Core50, where Cifar100, CUB200, and Imagenet-R are evaluated under the class-incremental setting, and Core50 is evaluated under the domain-incremental setting. Cifar100 is a relatively simple dataset containing 100 classes. CUB200 is composed of 200 subcategories belonging to birds. Imagenet-R is also composed of 200 classes that consist of data from different styles such as cartoons, graffiti, and origami. It is considered one of the most difficult datasets for class-incremental learning as semantic and covariate shifts occur. Throughout the experiments, all the above datasets are split into 10 tasks with an even number of classes in each task. Core50, on the other hand, is a popular dataset for domain-incremental learning that consists of 50 objects from 11 domains. In particular, 8 of them are used for training and the rest 3 for testing. None of the images in the 3 domains are seen during training. For all three datasets, we use class orders the same as [45, 46] for a fair comparison.

Evaluation Metrics Following conventional settings, we report the Average Accuracy after training on all tasks to evaluate the overall continual learning

Table 1: Results on the Split-Cifar100, Split-CUB200, Split-Imagenet-R and Core50 datasets. Split-Cifar100, Split-Imagenet-R and Split-CUB200 are class-incremental learning datasets, and Core50 is a domain-incremental learning dataset. Compared methods are grouped based on memory size. * denotes that the results are obtained through our PyTorch re-implementation [41].

Method	Split-Cifar100		Split-CUB200		Split-Imagenet-R		Core50	
	Buffer Size	Avg Acc	Buffer Size	Avg Acc	Buffer Size	Avg Acc	Buffer Size	Avg Acc
BiC [49]		81.4		81.9		64.6		79.3
GDumb [34]		81.7		61.8		65.9		74.9
DER++ [51]		83.9		77.4		66.7		79.7
Co^{2}L [5]		82.5		*N.A.*		65.9		79.8
L2P* [46]	5000	85.9	1000	82.3	5000	70.8	5000	85.1
S-Prompt* [44]		*N.A.*		*N.A.*		*N.A.*		92.1
DualPrompt* [45]		87.5		83.9		65.2		87.2
PromptFusion		**88.5**		**85.1**		**82.2**		**93.7**
PromptFusion-Lite		87.3		82.7		81.5		93.2

ability of the tested method. Formally, let $R_{T,i}$ be the classification accuracy of task T_i after training on task T_T, then the Average Accuracy A_T is defined as

$$A_T = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}. \tag{9}$$

Implementation Details We adopt a ViT-B-16 backbone for both CoOp and VPT and leave it frozen during the training phase. For the prompt size, M in CoOp and p in VPT are set to 30. The prompt length for augmentating CoOp is set to 40. All prompts are trained using the mini-batch AdamW optimizer with a learning rate of 0.002 on Split-Cifar100 and Split-CUB200, and a learning rate of 0.003 on Split-Imagenet-R and Core50, equipped with a cosine annealing scheduler. Considering their diverse difficulties, Split-Cifar100 and Split-CUB200 are trained for 3 epochs for every task while Split-ImageNet-R and Core50 are trained for 5 epochs.

6.2 Performance Comparison to State-of-the-art

We compare against both classical and most recent state-of-the-art continual learning methods: EWC [19], LwF [24], BiC [49], GDumb [34], DER++ [51], Co^{2}L [5], L2P [46], S-Prompt [44], DualPrompt [45] and CODA-Prompt [40]. In particular, DER++ is the best-performing non-prompt-based method, while L2P, S-Prompt, DualPrompt, and CODA-Prompt are all prompt-based methods. Since S-Prompt is specifically designed for domain-incremental learning, we only report its performance on the Core50 dataset.

The results on Split-Cifar100, Split-CUB200, Split-Imagenet-R and and Core50 are shown in Table 1, where PromptFusion outperforms all other alternatives.

For Split-Cifar100, PromptFusion reaches an Avg Acc of 88.5%. The performance is 2.6% and 1% higher than L2P and DualPrompt respectively. For Split-CUB200, PromptFusion outperforms L2P and DualPrompt by 2.8% and 1.2%. Considering that Cifar100 and CUB200 are relatively simple, the improvement is significant. For Split-Imagenet-R, PromptFusion can achieve an Avg Acc of 82.2%, significantly surpassing other methods. Similarly, PromptFusion-Lite also achieves competitive performances with less computational overhead than PromptFusion, as will be discussed in the next subsection.

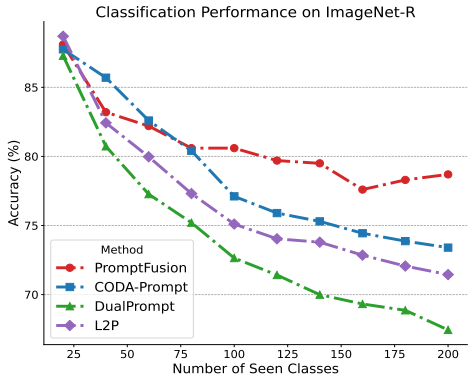

Fig. 8: Comparison on Split-Imagenet-R with no memory.

Here, the above results are obtained using a memory buffer. Since the current paradigm for prompt-based methods also focuses on the memory-free situation, we test PromptFusion on Split-Imagenet-R with 0 memory. As is shown in Fig 8, PromptFusion outperforms all other methods. In particular, when compared against CODA-Prompt, the best prompt-based approach, PromptFusion outperforms it by 5.3%. A major reason for this performance gain is that incorporating the Stabilizer makes our method robust against intra-class variations, as discussed in Section 4.

While Split-Cifar100, Split-CUB200, and Split-Imagenet-R are all commonly used datasets in the class-incremental learning scenario, we also test PromptFusion and PromptFusion-Lite under the domain-incremental learning scenario on the Core50 dataset. Notably, under such setting, both L2P and DualPrompt perform inferior to S-Prompt, as their main focus is on class-incremental learning. Nevertheless, our methods still outperform S-Prompt by 1.1% and 1.3% respectively, indicating that the capability of them is not restricted to any specific type of continual learning setting. Considering that S-Prompt is also based on CoOp, this is strong evidence that the success of our approach is not simply from this specific module.

Table 2: GFLOPs of prompt-based methods on Split-Imagnet-R.

Method	**Split-Imagnet-R**	
	Avg Acc	GFLOPs
L2P [34]	70.8	38.9
DualPrompt [51]	65.2	35.2
PromptFusion	**82.2**	42.6
PromptFusion-Lite	81.5	34.4

6.3 GFLOPs Comparisons to State-of-the-art

While computational concerns might be raised about our approaches, we show that they are similar to other prompt-based methods. This is because for L2P and DualPrompt, an additional raw pre-trained ViT is used to extract the [CLS] feature of the input to select the optimal prompts from the prompt pool. Consequently, as is shown in Table 2, PromptFusion only requires about

15% more GFLOPs than L2P and DualPrompt. Moreover, by incorporating the adaptive selection module, PromptFusion-Lite's GFLOPs requirement can even fall below that of L2P and DualPrompt. In particular, on Split-Imagnet-R, PromptFusion-Lite achieves an Avg Acc of 0.7% less than PromtFusion with 19.2% fewer GFLOPs.

6.4 Effect of memory size

While PromptFusion shows competitive performance in the memory-free setting, we would also like to test how it behaves for different buffer sizes. We analyze sizes of 1K, 2K, 3K, 4K, and 5K and report the results in Figure 9. In general, the effect of buffer size on the overall performance is rather small. Specifically on Core50, the buffer size does not significantly affect the performance of PromptFusion. This is unsurprising as the test data for Core50 is a constant set that has not been seen during training. Therefore the only factor influencing the performance is the model's capability of accumulating domain-invariant knowledge from sequential training on the training set, which PromptFusion demonstrates to excel at.

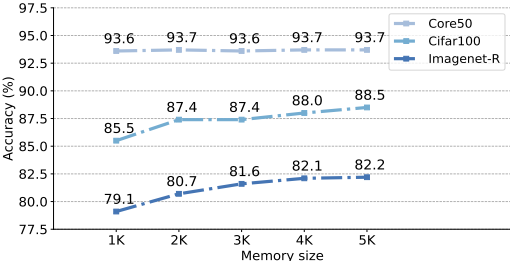

Fig. 9: Average Acc for different memory sizes on Split-Cifar100, Split-Imagenet-R, and Core50.

6.5 Ablation Study

Weight λ, Mask $\boldsymbol{W}$ and CoOp Augmentation We report in Table 3 the ablation study on weight λ, mask $\boldsymbol{W}$ and CoOp augmentation, and the results show that all pieces are significant to the success of our methods. Specifically, for Mask $\boldsymbol{W}$, its effect on Split-Cifar100 is much higher than that on Split-Imagenet-R. We posit that this is because new and old classes in the Imagenet-R dataset are more diverse as semantic and covariate shifts occur, resulting in weaker inter-class interference. As for λ, excluding it in PromptFusion means a simple summation of the two outcomes, and results in Table 3 demonstrate it to be sub-optimal. As discussed in Section 4, we expect different values of λ depending on the dataset being assessed. Indeed, experiments show that $\lambda = 1.06$ for Split-Cifar100 and $\lambda = 0.01$ for Split-Imagenet-R. Finally, as introduced in Section 5, CoOp is augmented by incorporating another set of image prompts in addition to the

Table 3: Ablation on different components of PromptFusion.

Mask $\boldsymbol{W}$	λ	Augmentation	**Split-Cifar100**
×	✓	✓	84.2
✓	×	✓	87.4
✓	✓	×	87.1
✓	✓	✓	**88.5**

language prompt, and the performance would increase by 1.4%, with only 0.02M extra trainable parameters.

Usage Penalty and Knowledge Distillation We further conduct an ablation study on PromptFusion-Lite targeting its objective function. Specifically, we would like to analyze the effect of the second and third terms in Eqn 8. As is shown in Tab 4, without using the usage penalty term, PromptFusion-Lite would almost always choose to activate the Booster, as evidenced by it having the same GFLOPs as PromptFusion. Without using knowledge distillation on $\mathbf{F}$, on the other hand, will result in a 0.3% performance degradation and slightly higher GFLOPs. Therefore, we conclude that both pieces are significant to the overall success of PromptFusion-Lite.

Table 4: Ablation on PromptFusion-Lite objective.

Usage Penalty	Knowl. Distill.	**Split-Cifar100** Avg Acc	GFLOPs
×	✓	**88.2**	42.6
✓	×	87.0	38.8
✓	✓	87.3	**38.3**

Prompt Length We also examine how prompt length affects the overall performance and the results are reported in Table 5. As is shown, our choice of $M = 30$ and $p = 30$ produces the best Average Acc. This would require a total of 1.66M trainable parameters on Split-Cifar100, which is infinitesimal compared to other approaches.

Table 5: Ablation on prompt length M and p for PromptFusion

Text Prompt	**Split-Cifar100**	Image Prompt	**Split-Cifar100**
$M = 20$	88.1	$p = 20$	87.9
$M = 40$	88.2	$p = 40$	88.1
$M = 30$	**88.5**	$p = 30$	**88.5**

7 Conclusion

In this paper, we introduced a dual architecture design PromptFusion for tackling the stability-plasticity dilemma in continual learning. PromptFusion is built on top of a Stabilizer module instantiated with CoOp and a Booster module instantiated with VPT, that decouples stability and plasticity into two independent problems. Furthermore, to reduce the computational overhead caused by the additional architecture, we proposed PromptFusion-Lite that leverages the Stabilizer as a base model, and adaptively decides on-the-fly whether to activate the Booster conditioned on the input sample. Extensive experiments showed that our approach achieved state-of-the-art results in class-incremental and domain-incremental learning under both memory and memory-free settings with PromptFusion-Lite successfully reducing more than 10% computational cost.

Acknowledgement

This project was supported by NSFC under Grant No. 62102092.

References

1. Abraham, W.C., Robins, A.: Memory retention–the synaptic stability versus plasticity dilemma. Trends in neurosciences (2005) 2
2. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV (2018) 4
3. Bahng, H., Jahanian, A., Sankaranarayanan, S., Isola, P.: Exploring visual prompts for adapting large-scale models. arXiv preprint arXiv:2203.17274 (2022) 8
4. Belouadah, E., Popescu, A.: Il2m: Class incremental learning with dual memory. In: ICCV (2019) 3, 4
5. Cha, H., Lee, J., Shin, J.: Co2l: Contrastive continual learning. In: ICCV (2021) 11
6. Chen, H., Wu, Z., Jiang, Y.G.: Multi-prompt alignment for multi-source unsupervised domain adaptation. NeurIPS (2023) 4
7. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. TPAMI (2021) 1, 4
8. Dhar, P., Singh, R.V., Peng, K.C., Wu, Z., Chellappa, R.: Learning without memorizing. In: CVPR (2019) 4
9. Douillard, A., Ramé, A., Couairon, G., Cord, M.: Dytox: Transformers for continual learning with dynamic token expansion. In: CVPR (2022) 4
10. Fini, E., Da Costa, V.G.T., Alameda-Pineda, X., Ricci, E., Alahari, K., Mairal, J.: Self-supervised models are continual learners. In: CVPR (2022) 4
11. Ge, C., Huang, R., Xie, M., Lai, Z., Song, S., Li, S., Huang, G.: Domain adaptation via prompt learning. arXiv preprint arXiv:2202.06687 (2022) 4
12. Hadsell, R., Rao, D., Rusu, A.A., Pascanu, R.: Embracing change: Continual learning in deep neural networks. Trends in cognitive sciences (2020) 1
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 1
14. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR (2019) 4
15. Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. Advances in Neural Information Processing Systems **32** (2019) 4
16. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016) 3, 9
17. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: ECCV (2022) 2, 4, 5
18. Ju, C., Han, T., Zheng, K., Zhang, Y., Xie, W.: Prompting visual-language models for efficient video understanding. arXiv preprint arXiv:2112.04478 (2021) 4
19. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. PNAS (2017) 2, 4, 11
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS (2012) 1
21. Kumaran, D., Hassabis, D., McClelland, J.L.: What learning systems do intelligent agents need? complementary learning systems theory updated. Trends in cognitive sciences (2016) 2
22. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021) 2, 4

23. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021) 4
24. Li, Z., Hoiem, D.: Learning without forgetting. TPAMI (2017) 4, 11
25. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586 (2021) 2, 4
26. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. NeurIPS (2017) 4
27. Lu, Y., Liu, J., Zhang, Y., Liu, Y., Tian, X.: Prompt distribution learning. In: CVPR (2022) 4
28. Mahdi Derakhshani, M., Sanchez, E., Bulat, A., Guilherme Turrisi da Costa, V., Snoek, C.G., Tzimiropoulos, G., Martinez, B.: Bayesian prompt learning for image-language model generalization. ICCV (2023) 4
29. Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., Sanner, S.: Online continual learning in image classification: An empirical survey. Neurocomputing (2022) 1
30. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: CVPR (2018) 4
31. McClelland, J.L., McNaughton, B.L., O'Reilly, R.C.: Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review (1995) 2
32. Mermillod, M., Bugaiska, A., Bonin, P.: The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects (2013) 2
33. Mirzadeh, S.I., Chaudhry, A., Yin, D., Nguyen, T., Pascanu, R., Gorur, D., Farajtabar, M.: Architecture matters in continual learning. arXiv preprint arXiv:2202.00275 (2022) 4
34. Prabhu, A., Torr, P.H., Dokania, P.K.: Gdumb: A simple approach that questions our progress in continual learning. In: ECCV (2020) 11, 12
35. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021) 5, 7
36. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR (2017) 3, 4
37. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS (2015) 1
38. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016) 4
39. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. NeurIPS (2017) 4
40. 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: CVPR (2023) 11
41. Sun, H.L., Zhou, D.W., Ye, H.J., Zhan, D.C.: Pilot: A pre-trained model-based continual learning toolbox. arXiv preprint arXiv:2309.07117 (2023) 11
42. Van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv preprint arXiv:1904.07734 (2019) 4
43. Von Oswald, J., Henning, C., Sacramento, J., Grewe, B.F.: Continual learning with hypernetworks. arXiv preprint arXiv:1906.00695 (2019) 4

44. Wang, Y., Huang, Z., Hong, X.: S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. arXiv preprint arXiv:2207.12819 (2022) 2, 3, 5, 11
45. Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.Y., Ren, X., Su, G., Perot, V., Dy, J., et al.: Dualprompt: Complementary prompting for rehearsal-free continual learning. arXiv preprint arXiv:2204.04799 (2022) 2, 3, 4, 7, 10, 11
46. Wang, Z., Zhang, Z., Lee, C.Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., Pfister, T.: Learning to prompt for continual learning. In: CVPR (2022) 2, 3, 4, 10, 11
47. Wei, C., Xie, S.M., Ma, T.: Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. In: NeurIPS (2021) 2
48. Wu, G., Gong, S., Li, P.: Striking a balance between stability and plasticity for class-incremental learning. In: ICCV. pp. 1124–1133 (2021) 2
49. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR (2019) 11
50. Wu, Z., Weng, Z., Peng, W., Yang, X., Li, A., Davis, L.S., Jiang, Y.G.: Building an open-vocabulary video clip model with better architectures, optimization and data. TPAMI (2024) 4
51. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: CVPR (2021) 4, 11, 12
52. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML (2017) 4
53. Zhang, G., Wang, L., Kang, G., Chen, L., Wei, Y.: Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In: ICCV (2023) 9
54. Zhou, K., Yang, J., Loy, C.C., Liu, Z.: Learning to prompt for vision-language models. IJCV (2022) 2, 4, 5