S-Prompts Learning with Pre-trained Transformers: An Occam's Razor for Domain Incremental Learning

Yabin Wang[1,2,*], **Zhiwu Huang**[2,†], **Xiaopeng Hong**[3,4,1‡]
[1]Xi'an Jiaotong University, P. R. China
[2]Singapore Management University, Singapore
[3]Harbin Institute of Technology, P. R. China
[4]Pengcheng Laboratory, P. R. China
iamwangyabin@stu.xjtu.edu.cn, zzhiwu.huang@gmail.com, hongxiaopeng@ieee.org

Abstract

State-of-the-art deep neural networks are still struggling to address the catastrophic forgetting problem in continual learning. In this paper, we propose one simple paradigm (named as S-Prompting) and two concrete approaches to highly reduce the forgetting degree in one of the most typical continual learning scenarios, i.e., domain increment learning (DIL). The key idea of the paradigm is to learn prompts independently across domains with pre-trained transformers, avoiding the use of exemplars that commonly appear in conventional methods. This results in a win-win game where the prompting can achieve the best for each domain. The independent prompting across domains only requests one single cross-entropy loss for training and one simple K-NN operation as a domain identifier for inference. The learning paradigm derives an image prompt learning approach and a novel language-image prompt learning approach. Owning an excellent scalability (0.03% parameter increase per domain), the best of our approaches achieves a remarkable relative improvement (an average of about 30%) over the best of the state-of-the-art exemplar-free methods for three standard DIL tasks, and even surpasses the best of them relatively by about 6% in average when they use exemplars. *Source code is available at* `https://github.com/iamwangyabin/S-Prompts`.

1 Introduction

State-of-the-art deep neural networks are capable of performing impressively on a wide variety of individual machine learning tasks. However, learning multiple tasks in sequence, generally called *continual learning*, remains a substantial challenge for deep learning. When trained on a new task, standard neural networks typically forget most of the knowledge related to previously learned tasks. This is a well-known phenomenon named as *catastrophic forgetting*.

There are three common continual learning scenarios. Task-incremental learning (TIL) is provided with task indexes for inference and thus using task-specific neural networks could overcome forgetting. Class-incremental learning (CIL) increases classes in sequence with task indexes being unknown for inference. Nevertheless, the classes are generally from the same domain, which more or less decreases the challenge. In this paper, we focus on exploring domain-incremental learning (DIL), where classes keep the same but the involved domains commonly vary a lot in sequence, with task indexes being not provided for inference. The larger domain gaps result in more severe forgetting.

[*]This work is done when Yabin Wang visits Singapore Management University
[†]Corresponding Author
[‡]Corresponding Author

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

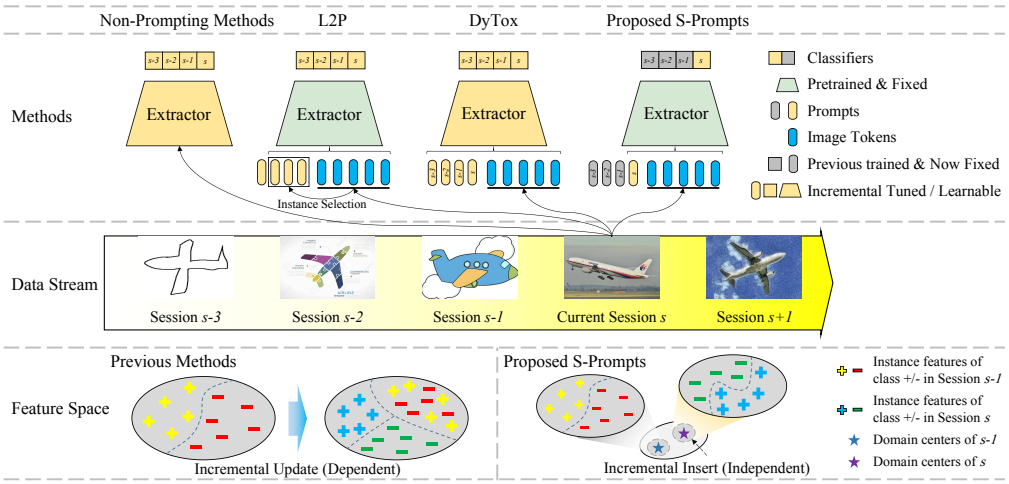

Figure 1: Comparison of the proposed S-Prompts paradigm against non-prompting methods, and prompting methods (L2P [66], DyTox [14]). Exiting methods generally learn sequential tasks/sessions/domains dependently, producing a single feature space where the classes from different domains are less separable (more forgetting or worse transferring). In contrast, the proposed S-Prompts learn the tasks independently. This paradigm leads to one subspace for every domain, making the classes more separable (less/no forgetting and better transferring).

To alleviate catastrophic forgetting, numerous methods have been proposed for continual learning. One of the most effective methods is to leverage a buffer of exemplars from old tasks to perform either a rehearsal or a distillation when tuning the whole network to learn new tasks (see Non-Prompting Methods in Fig.1). Both the rehearsal and the distillation typically decrease the forgetting degree. However, it is often more desired that the exemplars of old tasks are not stored for better data security and privacy. Moreover, storing a large amount of exemplars might run out available memories. Therefore, our focus is shifted to the challenging *exemplar-free DIL* in this paper.

One of the most promising solutions to the exemplar-free DIL problem is to learn a set of *prompts*[4] over transformers that achieve the state-of-the-arts in a wide range of areas. In this solution, the domain-specific knowledge is stored by a prompt pool, and hence a rehearsal buffer is no longer mandatory to mitigate forgetting. For instance, two recent prompting methods [14, 66][5] aim at *learning task/domain-specific prompts dependently across domains* in an incremental update fashion. In [14], the prompts are added task by task, and the newly added prompts share the same task-attention learning with the old prompts. This requests a joint tuning on all the prompts and the entire transformer using a distillation loss to keep a balance between the new prompting and the old ones. In [66], a pool of prompts is initialized at the right beginning, and a set of the prompts from the pool are selected for each instance. The instance-level prompts generally share task-level knowledge among any similar instances, by jointly tuning the selected prompts and the whole classifier, with the pre-trained transformer being fixed. While using different prompting approaches, [14, 66] both follow a commonly-respected principle that the prompting should keep sharing across tasks/domains. However, the sharing-driven dependent prompt tuning paradigm as well as many of the non-prompting paradigms generally result in *a tug-of-war (or a zero-sum game)*, in which one side's gain is equivalent to the other's loss, as studied in [56, 33]. In other words, these paradigms generally accumulate new knowledge in the same feature space, which is very likely to mix up the subspaces of old/new knowledge resulting in less separable classes (more forgetting or worse transferring) (Fig.1).

In this paper, we explore a rule-breaking idea to instead play *a win-win game, i.e., learning the prompts independently across domains so that the prompting can achieve the best for each domain.* The excellent generalization capability of the transformers and the strong transfer learning ability of the recently appeared prompting mechanisms allow for the realization of this idea. In particular, we introduce a new learning paradigm that learns the prompts independently domain by domain, and incrementally inserts the learned prompts in a pool. As shown in Fig.1, compared to the other prompting and non-prompting methods, the new paradigm suggests merely tuning the current domain-

[4]For simplicity and consistency, we use 'prompt' to represent 'prompt token' for context/domain knowledge encoding, and we utilize 'token' to denote those normal tokens on images and classes, throughout the paper.

[5]A concurrent work [12] learns class-level prompts dependently on transformers for the CIL problem.

related prompts and classifiers with all the rest unrelated network components being fixed. The frozen pre-trained transformer is capable of extracting general features, and the independent prompting reaches the best fit on each domain. The independent prompting is able to generate a feature space where each subspace is spanned by one single domain, and all the subspaces are less overlapped (see Feature Space in Fig.1). This could lead to no forgetting across domains. The learning process only requests the most naive cross-entropy loss for supervision. The inference is also simple and efficient. It simply uses K-NN to search for the nearest domain center generated by K-Means on the training data to the features of the given test samples, followed by prepending the learned domain-associated prompts with the image tokens to the transformer for the final classification. As S domain-related prompts will be finally learned independently, where S is the total number of domains, we name the proposed paradigm as *S-Prompts* or *S-Prompting* for simplicity throughout the paper. We hope the proposed S-Prompting becomes an *Occam's Razor* (i.e., a simple and elegant principle) for DIL.

The S-Prompting paradigm can be applied to any pre-trained transformers. In this paper, we exploit two approaches of *S-Prompts* learning. One is based on vision transformer (ViT) [13], and the other is based on Contrastive Language-Image Pre-Training transformer (CLIP) [54]. Particularly, we suggest the CLIP-based approach that exploits a new language-image prompting scheme. The key idea is to prompt the pair of language-image transformers synchronously with a pair of learnable language-image prompts at the two ends. This highly enhances CLIP's transfer learning ability on a variety of new domains. As a result, this approach owns a great ability to reach a favorable feature space where the subspaces of different domains are pushed far away from each other and the classes are separated clearly, leading to highly impressive DIL performances. To the best of our knowledge, this is the first time to introduce the *paired language-image prompting scheme* in the literature.

The contributions of this paper can be summarized as follows:

- We introduce a simple and effective S-Prompts learning paradigm with pre-trained transformers for the DIL problem. This paradigm breaks the commonly-respected principle in continual learning.
- We exploit an Image S-Prompts (S-iPrompts) learning approach and a novel Language-Image S-Prompts (S-liPrompts) learning approach, based on two different pre-trained transformers.
- The best S-Prompts surpasses the best of the state-of-the-art exemplar-free methods significantly (an average of *30%* relative improvement) for three standard DIL benchmark datasets, and even outperforms them relatively by an average of *6%* when they use exemplars. S-Prompts can scale to a large number of domains with a tiny parameter increase, e.g., *0.03%* per domain in S-liPrompts.

2 Related Work

Continual Learning. Numerous methods have been exploited to address catastrophic forgetting. For instance, memory-based methods alleviate forgetting by either replaying a small set of examples from past tasks saved in a memory [57, 55, 46, 7, 23, 28, 9] or replaying synthesized data from previous tasks using generative models [60, 63]. Distillation-based methods (e.g., [40, 55, 22, 5, 67, 70, 62, 44, 49, 11]) apply the technique of knowledge distillation [21] to mitigate catastrophic forgetting. In addition, there exist other kinds of methods like gradient-based methods (e.g., [56, 73, 16, 58, 61, 64]), regularization-based ones (e.g. [32, 74, 2, 1]), and expansion-based ones (e.g. [69, 34, 25]).

Recently, some of work like [48, 31, 29, 36, 66, 18, 78] adapts the CIL methods to the DIL problems such as continual deepfake detection, where each domain distinguishes between deepfakes and reals. Following them, one of our approaches uses the common classifiers from the CIL methods. While each domain share the same classes, the incremental classifiers simply treat them as difference classes. The finer-grained task could make networks more powerful [48].

Prompt Learning. The general idea of prompting is to learn a function to modify the input texts or images, such that the language or image model obtains additional information about the task [66]. There have been emerging a bunch of prompting methods, such as [35, 38, 26, 76, 77, 3, 24, 66, 14], to name a few[6]. Nevertheless, these methods merely learn either image-end prompts or language-end prompts (at most using conditions from image-end), while ours explicitly learns joint language-image prompting. Recently, L2P [66] and DyTox [14] exploit dependent prompting methods for continual learning, which leads to a tug-of-war issue, as discussed in Sec.1. Our suggested S-Prompting paradigm instead learns task/domain associated prompts independently for a win-win gain.

[6]For a comprehensive survey on the prompting methods, please refer to [43].

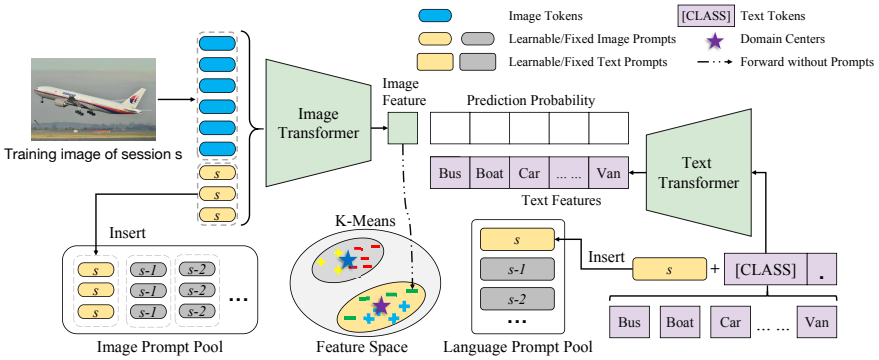

Figure 2: Illustration of the proposed language-image S-Prompts (S-liPrompts) learning approach. $s, s-1, s-2$ denote the session/domain indexes. For the inference, we simply adopt the following steps: 1) feeding the tokens of given test image into the image transformer to get the image feature, 2) using K-NN to search for the nearest centroids obtained by K-Means on training data to the test image feature, 3) feeding the image token and the nearest centroids (domain centers) related image prompt into the image transformer, with the centroid related language prompt and class tokens being fed into the text transformer, 4) computing the language-image prediction probabilities for the final classification. Note that both K-Means and K-NN are performed on the image feature space of the fixed pre-trained image transformer without using any prompts.

3 Proposed Approach

3.1 Problem Formulation and S-Prompts Learning

We focus on the problem of exemplar-free domain incremental learning (DIL). In a common and practical DIL scenario, the model is expected to learn knowledge of various domains in sequence to finally become a universal expert for all the domains. Formally, at domain s, the model accesses to the incoming data $\mathbb{D}_s = \{x_i^s, y_i^s\}_{i=1}^{N_s}$ where x_i^s is the i-th input image from domain s and y_i^s is the corresponding label, and the total number of samples at domain s is N_s. In the entire incremental learning stream, S domains are presented sequentially, and each domain s has data $\mathbb{D}_s$ from different or even highly heterogeneous domains (Fig.1). The total class categories the model observed is $\mathbb{C}$, and in DIL all categories are identified from the beginning and will not increase during the incremental learning stream. The exemplar-free DIL requests zero exemplar from the learned sessions during the training and the inference for better data security, privacy and cheaper memory consumption.

To address the exemplar-free DIL problem, we propose an S-Prompts learning paradigm that is simple and effective. The key idea is to learn prompts domain by domain independently with pre-trained transformers. The pre-trained transformers are fixed to extract highly expressive low-level features, and the prompts are tuned to encode the high-level domain knowledge. In this case, each domain knowledge will be encoded by one prompt or one set of prompts. This not only avoids the use of exemplars but also highly reduces the catastrophic forgetting. Accordingly, our focus is on learning the prompts of S domains or so-called *S-Prompts*. As shown in Fig.1, the suggested S-Prompting merely tunes the current domain related prompts and classifier with all the rest neural network components, including the transformer, the current session unrelated prompts and classifiers, being fixed. In this case, each set of domain-specific prompts are learned independently from the other domains related prompts. This requests a domain identifier for inference. In particular, we apply K-Means to store the centriods for each domain during training, and K-NN to search for the nearest centroid of the given test image feature to identify its corresponding domain during inference. Note that both K-Means and K-NN are performed on the feature space of the fixed pre-trained transformer without using any prompts. Finally, we prepend the domain associated prompts and the image tokens to the transformer for image classification. The inference steps are detailed in the caption of Fig.2. Since DIL's domain shifts are generally explicit and the state-of-the-art transformers are commonly good at image classification, the suggested domain identifier performs well in standard DIL tasks. Due to the strong generalization capability of transformers, the suggested S-Prompting still performs well for those cases with incorrect domain identification or those on unseen domains. Thanks to this mechanism, the suggested S-Prompting can achieve the best transferring for each domain, leading to the least forgetting. Moreover, the proposed S-Prompting only requests the most naive cross-entropy loss for the guidance on classification, which further enhances the paradigm's simplicity.

In this paper, we mainly study two concrete S-Prompting approaches: 1) *Image S-Prompts (S-iPrompts) learning* based on the pre-trained vision transformer (ViT) [13], and 2) *Language-Image S-Prompts (S-liPrompts)* based on the pre-trained Contrastive Language-Image Pre-Training transformer (CLIP) [54]. In essential, the main differences between these two approaches are two-folds: 1) prompt design, and 2) classifier design. We detail them below for these two approaches.

3.2 Image S-Prompts (S-iPrompts) Learning Approach

Prompt Design. In the approach of S-iPrompts, for one domain s, we suggest using an independent set of continuous learnable parameters (i.e., one image prompt) $P_s^i \in \mathbb{R}^{L_i \times D_i}$ as a part of inputs to the pre-trained ViT, where $L_i \in \mathbb{R}$ and $D_i \in \mathbb{R}$ indicate the prompt's length and embedding dimension respectively. As shown in Fig. 2, the whole embedding of any single image from domain s follows the format of $x = [x_{img}, P_s^i, x_{cls}]$, where x_{img} denotes the image tokens and x_{cls} represents the pre-trained class tokens of ViT. The extended image embedding is used as an input of the transformer blocks as normal ViT does. When trained on a new domain $s+1$, a new independent set of images prompts P_{s+1}^i is added. Hence, learning all the domains sequentially results in a pool of domain-wise prompt pool. The image prompt pool can be defined as $\mathcal{P}^i = \left\{P_1^i, P_2^i, ..., P_S^i\right\}$, where $P_s^i \in \mathbb{R}^{L_i \times D_i}$ is a single set of prompts belonging to domain s, and S is the total number of domains.

Classifier Design. The classifier of S-iPrompts is a normal fully connected (FC) layer, which is made of a linear projection parameterized by $[W_s, b_s]$, where $W_s \in \mathbb{R}^{C \times D_i}$, $b_s \in \mathbb{R}^C$, D_i, C indicate the embedding dimension and the total class number respectively, and s is the domain index. Formally, the classifier output is calculated by

$$\hat{y} = \sigma(W_s f(x) + b_s), \tag{1}$$

where $f(x)$ indicates the output image feature of the ViT encoder f, and σ is the softmax activation operation, $\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}}$, for $j = 1, 2, \ldots, C$ (C indicates the number of classes). Every learning session owns such one classifier, and thus we use a classifier pool $\mathcal{P}^{fc} = \{[W_1, b_1], [W_2, b_2], ..., [W_S, b_S]\}$ to store all these classifiers that are independent to each other. During the inference, once the domain center is selected, the associated prompts and the corresponding classifier will be chosen for image classification.

3.3 Language-Image S-Prompts (S-liPrompts) Learning Approach

The S-liPrompts approach aims at exploiting an effective prompting scheme on CLIP [54] so as to transfer the pre-trained CLIP to a variety of domains. As illustrated in Fig.2, the CLIP model consists of two transformers for image and text encoding respectively. The CLIP model was trained on an enormous amount of image-text pairs in a contrastive learning manner, which aligns the pair of language-image transformers very tightly. Hence, it is essential to prompt the pair of CLIP transformers synchronously for an effective transfer learning. To this end, we follow the paradigm of S-Prompts to exploit a joint language-image prompting scheme. The scheme learns a pair of an image-end prompt and a language-end prompt domain by domain, with the pre-trained CLIP always being fixed. On one hand, fixing CLIP enables the prompting model to inherit the extraordinary capability of CLIP in terms of zero-shot learning and few-shot learning [76]. On the other hand, the joint prompting can make the image-end transformer has an excellent fit into any upcoming domains that might be highly different from the pre-training data. This is mainly because synchronously prompting the language-end transformer could get more semantic information and more domain-aware (Fig.4).

Prompt Design. The design of image-end prompts for S-liPrompts is the same as that of S-iPrompts. For the design of language-end prompts, we suggest using a learnable context prompt to replace the artificially designed prompts used by the original CLIP. This design can better adapt pre-trained vision-language models to downstream tasks, especially for few-shot learning. In particular, for the learning domain s, we use M learnable vectors v as the context prompt $P_s^l = \{v_1, v_2, ..., v_M\} \in \mathbb{R}^{L_l \times D_l}$, where L_l, D_l are the prompt's length and embedding dimension respectively. We regard it as text prompts corresponding to image prompts used in ViT. For the j-th class, the whole input t_j of the text encoder is of the format $t_j = \left\{P_s^l, c_j\right\}$, where c_j is the word embeddings of the j-th class names $[\mathrm{CLASS}]$ and has the same dimension of D_l. The text prompts are also associated with one single domain, and they are independent to those from the other domains. After learning all the sequential domains, a pool of text prompts is formed as $\mathcal{P}^l = \left\{P_1^l, P_2^l, ..., P_S^l\right\}$, where S is the domain number.

The learnable context prompt can better adopt the pre-trained CLIP to various domains, which is an essential property for incremental learning.

Classifier Design. The text encoder g of CLIP receives the input t_j defined above, and produces a vectorized representation $g(t_j)$, where $j = 1, 2, \ldots, C$, and C denotes the class number. The contrastive language-image pre-training of CLIP uses a contrastive loss to learn a joint embedding space for vision and language models. One of its objectives is to maximize the cosine similarity of each image with its matched text and minimize the cosine similarities with all other unmatched texts. The other objective is computed in a similar fashion for each text. Let $f(x)$ be the image feature generated by the image encoder f, and $\{g(t_j)\}_j^C$ is a set of weight vectors produced by the text encoder $g(.)$, each $g(t_j)$ representing the text feature of j-class. The prediction probability is computed as

$$p(y_j|x) = \frac{\exp(\langle f(x), g(t_j)\rangle)}{\sum_{k=1}^{C} \exp(\langle f(x), g(t_k)\rangle)}, \tag{2}$$

where $\langle ., . \rangle$ is the cosine similarity, and $\langle x, y\rangle = \frac{x \cdot y}{|x||y|}$. We conduct the softmax activation operation σ over the result of Eq.2 to obtain the final logits for image classification.

Like S-iPrompts, S-liPrompts also grows the pool of domain-based classifiers (Fig.2). Compared with the FC-based classifiers in S-iPrompts, there are two main advantages using the CLIP-based classifier in S-liPrompts. Firstly, the language model provides more domain information using the domain-specific text prompts for DIL, because it generally leads to more domain-separable features (see Fig.4). Secondly, in the incremental process, only one domain context needs to be stored for each session, while the class names are shared, which highly reduces the memory consumption.

4 Experiments

Datasets. We perform experiments on three standard DIL benchmark datasets. They are CDDB [36], CORe50 [45], and DomainNet [52]. CDDB [36] is a dataset for continual deepfake detection, which is a challenging DIL task. It designs Easy, Long, and Hard tracks. We select the most challenging track (i.e., the Hard track). The track requests learning on 5 sequential deepfake detection domains (about 27,000 images), which are GauGAN, BigGAN, WildDeepfake, WhichFaceReal, and SAN respectively. CORe50 [45] is a widely used dataset for continual object recognition. It has 50 categories from 11 distinct domains. The continual learning setting uses 8 domains (120,000 images) for incremental training and the data from the rest (i.e., unseen) domains as test set. DomainNet [52] is a dataset for domain adaptation and DIL, which has 345 categories and roughly 600,000 images. Images from DomainNet is split into 6 domains. The DIL setup on DomainNet is the same as that of CaSSLe [18]. We follow [36] to report the average forward detection accuracy and the average forgetting degree on CDDB-Hard. For CORe50 and DomainNet, we follow [66] and [18] to present the average forward classification accuracy.

Methods. We compare the proposed S-Prompts methods (S-iPrompts, S-liPrompts) against the state-of-the-art CIL/DIL methods. They include non-prompting methods EWC [32], LwF [40] ER [8], GDumb [53], BiC [67], DER++ [4] and Co2L [6], as well as the prompting methods DyTox [14] and L2P [66] and self-supervised learning method CaSSLe [18]. Among them, since EWC [32], LwF [40], L2P [66], CaSSLe [18] can work for examplar-free DIL, we regard them as our real competitors. For DyTox and L2P, we use their official implementations by tuning their parameters for better performances. To compare fairly, we use the same ViT model (i.e., ViT-B/16 [13]) for most of the real competitors [7] as well as our S-iPrompts/S-liPrompts. The proposed S-liPrompts's text-end encoder is the same as that of CLIP [54]. For more implementations details and studies, please refer to Appendix.

Overheads. For each domain, our proposed S-Prompting methods additionally use a set of prompts as well as a set of centroids from K-Means. For S-liPrompts, the overhead in memory is merely of $+0.03\%$ per domain. Compared to the other prompting methods DyTox and L2P, S-liPrompts enjoys much less memory overhead of the growing pool of domain-based classifiers due to its CLIP based classifier design (i.e., extremely cheap prompts and shared class names) when the class number

[7] Among the competitors, only DyTox [14] uses a more advanced ViT model(ConViT [15]), since it fails to get promising results (e.g., the result is even inferior than that of the random guess on CDDB) using ViT-B/16 [13].

Table 1: Results on CDDB-Hard for deepfake DIL. **Bold**: best exemplar-free DIL results, Underline: second best exemplar-free DIL results. Upper-bound: supervised finetuning on the i.i.d. data of all tasks, which is usually regarded as the upper bound performance a method can achieve [66]. * denotes results copied from [36].

Method	Buffer size	Average Acc (↑)	Forgetting (↑)
LRCIL [51]	100/class	76.39*	-4.39*
iCaRL [48]		79.76*	-8.73*
LUCIR [22]		82.53*	-5.34*
LRCIL [51]	50/class	74.01*	-8.62*
iCaRL [48]		73.98*	-14.50*
LUCIR [22]		80.77*	-7.85*
DyTox [14]		86.21	-1.55
EWC [32]	0/class	50.59	-42.62
LwF [40]		60.94	-13.53
DyTox [14]		51.27	-45.85
L2P [66]		61.28	-9.23
S-iPrompts (ours)		74.51	-1.30
S-liPrompts (ours)		**88.65**	**-0.69**
Upper-bound (S-iPrompts)	-	85.50	-
Upper-bound (S-liPrompts)	-	91.91	-

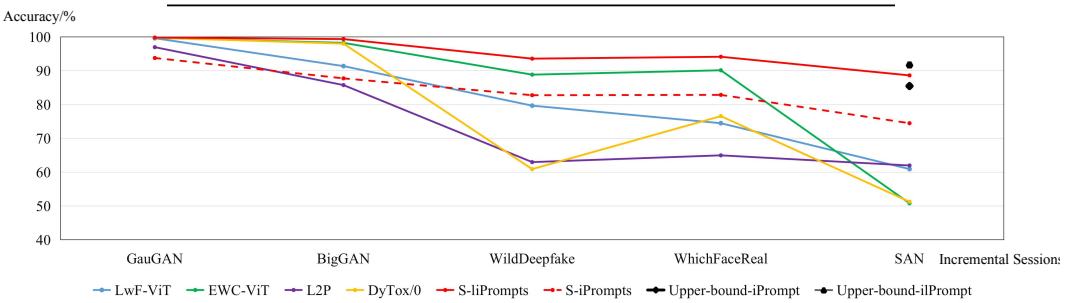

Figure 3: Accuracy curves of the competing methods and the proposed S-Prompts on CDDB-Hard.

is large (e.g., 345 classes in DomainNet). Accordingly, the proposed methods own an excellent scalability. On the other hand, S-liPrompts takes 11.84 millisecond (ms) in average to classify an image on a single RTX-3090, while the pre-trained CLIP model takes an average of 9.37ms. This shows that the additional domain identification merely takes around 2.5ms, which is affordable for the real-world scenarios. The appendix presents more details about the overheads.

Results. The results in Table 1, Table 2 and Table 3 demonstrate that the proposed S-iPrompts and S-liPrompts significantly outperform the other exemplar-free methods including the two recent prompting methods L2P and DyTox. We can compute that the proposed S-liPrompts obtains a considerable relative improvement (an average of roughly 30%) over the best of these methods in terms of forward classification accuracy. From Table 1 and Fig.3, we could also find the proposed S-Prompting methods' forgetting degrees are much less than those of the others. On the other hand, compared to those methods using exemplars of old tasks, our S-liPrompting still enjoys the clear superiority (an average of about 6% relative improvement) over the best of them. Besides, it is fairly surprising that our S-liPrompts even surpasses its upper-bound (joint training version) on DomainNet that contains highly heterogeneous domains. This is very likely that the joint learning cannot handle the large domain shifts well, and our independent prompting avoids this issue. Another interesting phenomena is that our S-Prompts still enjoy the extraordinary generalization capability (both S-iPrompts and S-liPrompts clearly surpass others) on the unseen domains on CORe50 where all the tested domains do not appear in the incremental training. The inferior performance of L2P on DomianNet might be due to the less transfer learning capability of its dependent instance-level prompting (prompt sharing) on heterogeneous domains. While DyTox reaches closer to the proposed S-liPrompts on DomainNet, it requests a large memory overhead (i.e., 50×345=17,250, where 50, 345 are the exemplar number per/class and the class number respectively) for the storage of exemplars[8].

Ablations. The studies on the cases of fixing either FC length/weights or Prompt length/weights in Table 4 justifies the necessity of using the proposed growing classifier pool and independent

[8]DyTox [14] often fails without using exemplars, as it requests a distillation on exemplars for better prompting.

Table 2: Results on CORe50 for object DIL. **Bold**: best exemplar-free DIL results, Underline: second best exemplar-free DIL results. Upper-bound: supervised finetuning on the i.i.d. data of all tasks, which is usually regarded as the upper bound performance a method can achieve [66]. * indicates results copied from [66].

Method	Buffer size	Average Acc (↑)
ER [8]		80.10±0.56*
GDumb [53]		74.92±0.25*
BiC [67]		79.28±0.30*
DER++ [4]	50/class	79.70±0.44*
Co^{2}L [6]		79.75±0.84*
DyTox [14]		79.21±0.10
L2P [66]		81.07±0.13*
EWC [32]		74.82±0.60*
LwF [40]		75.45±0.40*
L2P [66]	0/class	78.33±0.06*
S-iPrompts (ours)		83.13±0.51
S-liPrompts (ours)		**89.06**±0.86
Upper-bound (S-iPrompts)	-	84.01±0.53
Upper-bound (S-liPrompts)	-	93.19±0.21

Table 3: Results (task-agnostic test results) on DomainNet for object DIL. **Bold**: best exemplar-free DIL results, Underline: second best exemplar-free DIL results. †: the total buffer size is 50×345=17,250, where 345 is the total number of DomainNet classes. It is a large memory overhead. Upper-bound: supervised finetuning on the i.i.d. data of all tasks, which is usually regarded as the upper bound performance a method can achieve [66]. * indicates results copied from [18].

Method	Buffer size	Average Acc (↑)
DyTox [14]	50/class†	62.94
EWC [32]		47.62
LwF [40]		49.19
SimCLR [10]-CaSSLe [18]		44.2*
BYOL [20]-CaSSLe [18]		49.7*
Barlow Twins[72]-CaSSLe [18]		48.9*
Supervised Contrastive [30]-CaSSLe [18]	0/class	50.9*
L2P [66]		40.15
S-iPrompts (ours)		50.62
S-liPrompts (ours)		**67.78**
Upper-bound (S-iPrompts)	-	63.22
Upper-bound (S-liPrompts)	-	64.08

image/language prompting for S-iPrompts/S-liPrompts. We also study the case that the domains are randomly selected for the inference. It is surprising that the accuracy (80.12%) does not drop a lot and is still better than those of the state-of-the-art methods. In fact, the domain identification accuracies are not high using either ViT-based or CLIP-based features, as shown in Table 5. Both of these two results show that the proposed S-Promptings are not fundamentally influenced by the domain identification accuracy. As shown in Fig.4, the intrinsic reason might be that the proposed S-iPrompts and S-liPrompts actually result in a clearer domain separation compared to the pre-trained ViT and CLIP, and also they inherit the excellent generalization capabilities of the used transformers. This is justified by the result of S-liPrompts (K=5 & Zero-shot) that is merely trained on the first domain and tested on all the domains. Through the study on different K values in K-Means, we find that the proposed S-liPrompts works very stably for different K values. Hence, we consistently use K=5 for all the experiments on CDDB-Hard, CORe50 and DomainNet.

Apart from the number of centers K for K-Means (studied in the Table 4), there are three other key hyper-parameters for the proposed S-Prompts, including the length of an image prompt L_i, the length of a language prompt L_l (only for S-liPrompts), and the value of K in K-NN selection algorithm. We perform further ablations on the proposed S-liPrompts by keeping one of these hyper-parameters varied and fixing the rest hyper-parameters on CDDB-Hard. Fig.5 (Top Left) illustrates the accuracy curve of using different image prompt lengths. We can see that too small L_i has negative impact, which means that the prompts have not much ability to transfer the pre-trained model to downstream tasks. However, too large L_i can not help the model to get better performance. Fig.5 (Top Right) illustrates the accuracy curve of using different language prompt length. Compared with the study

Table 4: Results of ablating components of the proposed two S-Prompts (i.e., S-iPrompts, S-liPrompts) methods on CDDB-Hard for exemplar-free deepfake DIL. Different K values for K-Means.

Method	Average Acc (↑)	Forgetting (↑)
S-iPrompts (fixed FC length)	52.07	-28.89
S-iPrompts (fixed FC weights)	72.13	-0.86
S-iPrompts (K=5 & Zero-shot)	62.08	-
S-iPrompts (final, K=5)	74.51	-1.30
S-liPrompts (fixed Language & Image Prompt length)	64.90	-12.7
S-liPrompts (fixed Language Prompt length)	52.80	-29.13
S-liPrompts (fixed Language Prompt weights)	72.29	-0.99
S-liPrompts (K=5 & Zero-shot)	66.57	-
S-liPrompts (K=5 & Domain Random Selection)	80.12	-1.04
S-liPrompts (K=1)	88.72	-1.09
S-liPrompts (K=3)	88.55	-0.77
S-liPrompts (K=7)	88.40	-0.82
S-liPrompts (K=9)	88.32	-0.88
S-liPrompts (final, K=5)	**88.65**	**-0.69**

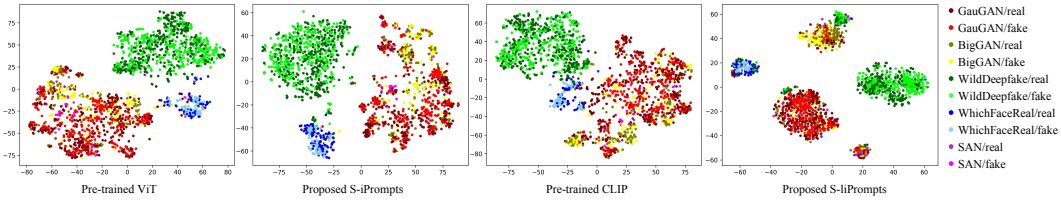

Figure 4: t-SNE visualization on the resulting feature spaces of pre-trained ViT, proposed S-iPrompts, pre-trained CLIP and proposed S-liPrompts on CDDB-Hard.

Table 5: Domain identification accuracy (task-wise) when K=5 for K-Means and K=1 for K-NN on CDDB-Hard.

	GauGAN	BigGAN	WildDeepfake	WhichFaceReal	SAN	Average Acc (↑)
ViT	0.82	0.78	0.98	0.98	0.56	0.82
CLIP	0.74	0.68	0.97	0.97	0.66	0.80

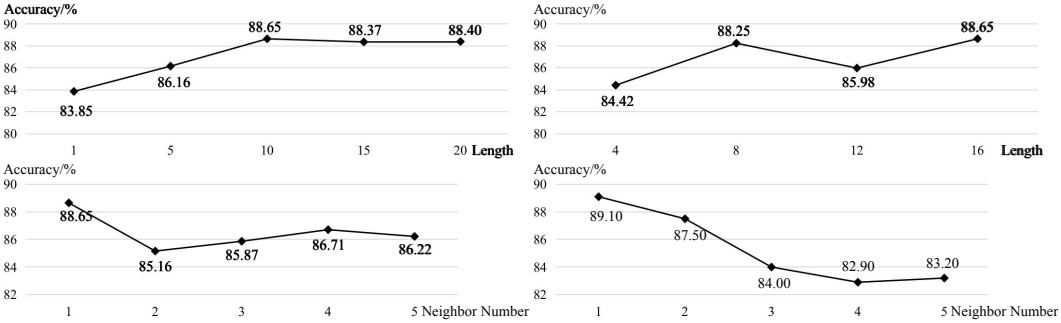

Figure 5: Domain incremental learning accuracies of the proposed S-liPrompts (Top Left) using different image prompt lengths with the language prompt length being fixed as 16, (Top Right) using different language prompt lengths with the image prompt length being fixed as 10, (Bottom Left) using different K values for K-NN based domain identification, and (Bottom Right) domain identification accuracies (task-agnostic) of the cases with different K values for K-NN, on the CDDB-Hard dataset.

on image prompts, language prompts also favor a sufficient length (the longest one is 16 in our case). Fig.5 (Bottom Left) studies the influence of using different K values in K-NN on the domain incremental learning accuracy. From the results, we can find that 1-NN works clearly better than the other cases. The main reason is that the other K-NN cases result in a clear performance drop for domain identification (Fig.5 (Bottom Right)). Therefore, we suggest using 1-NN that is the simplest and the most effective for the proposed S-Prompts.

TIL vs. DIL. Table 6 shows the results of the proposed S-liPrompts in the scenarios of task-incremental learning (TIL) and domain-incremental learning (DIL). TIL offers domain indexes for the inference, while DIL does not provide the information. We report task-wise average accuracy (AA) that computes the average on all the task-based accuracies, and also we present task-agnostic AA that computes the average on all the test images without considering the task indexes. Therefore,

Table 6: Accuracies (%) of the proposed S-liPrompts in the setups of task-incremental learning (TIL) and domain-incremental learning (DIL) on CDDB-Hard. TIL provides domain indexes for the inference phase, while DIL does not offer domain indexes for the test. AA: average accuracy. Task-wise AA: average on all the task-based accuracies. Task-agnostic AA: average on all the test images without considering the task indexes.

	GauGAN	BigGAN	WildDeepfake	WhichFaceReal	SAN	Task-wise AA	Task-agnostic AA
TIL	99.85	99.50	82.21	96.50	85.56	**92.72**	**92.51**
DIL	99.30	96.75	82.06	96.25	68.89	88.65	91.54

Table 7: Accuracies (%) of the application of the trained S-liPrompts on S1-S5 to out-of-domains OOD1-OOD3 that are 3 unseen domains used in CDDB. S1-S5: GauGAN, BigGAN, WildDeepfake, WhichFaceReal, SAN, which are used to train S-liPrompts on CDDB-Hard. OOD1-OOD3: FaceForensic++, Glow, StarGAN, which are not used to trained on S-liPrompts. AA: average accuracy. Task-wise AA: average on all the task-based accuracies. Task-agnostic AA: average on all the test images without considering the task indexes. †: CDDB-Hard runner-up method that uses 50 exemplars per class, while ours utilizes ZERO exemplar.

	S1	S2	S3	S4	S5	OOD1	OOD2	OOD3	Task-wise AA	Task-agnostic AA
S1 (ours)	99.85	87.75	49.15	50.50	50.00	50.92	63.52	71.10	65.35	68.43
S2 (ours)	99.90	98.88	51.24	52.50	50.00	74.68	73.42	80.06	72.59	76.56
S3 (ours)	99.90	98.75	82.16	52.25	51.11	73.29	66.83	78.15	75.31	77.89
S4 (ours)	99.85	98.50	82.06	96.25	52.22	75.60	64.58	92.60	82.71	82.78
S5 (ours)	99.30	96.75	82.06	96.25	68.89	75.69	64.44	92.46	**84.48**	**82.63**
S5 (DyTox†[14])	97.90	96.12	83.23	94.50	62.22	81.42	54.83	55.77	78.25	69.02

the task-agnostic AA is actually more in line with the real-world settings. The task-agnostic AAs show that S-liPrompts almost *closes the gap between DIL and TIL.*

Generalization. We test the generalization capability of the proposed S-liPrompts on unseen domains i.e., out-of-domain data (OOD). In detail, we apply the pre-trained S-liPrompts on CDDB-Hard's 5 domains to 3 OOD data, which are FaceForensic++, Glow, StarGAN used in CDDB [36]. Table 7 reports the results of the OOD experiment. The accuracies on the three OODs increase clearly as S-liPrompts is trained on more seen data (S1-S5). We report the accuracies in both task-wise and task-agnostic cases. The results show that our S-liPrompts has a better generalization ability to these OODs than DyTox [14], which is the runner-up on CDDB-Hard.

5 Discussion and Conclusion

Continual learning studies the problem of learning from a data stream from changing domains/tasks. The general principle of conventional methods is to adapt to new domains/tasks by accumulating previously acquired knowledge, while protecting previous learning from being erased. Beyond this commonly-respected principle, this paper proposes a rule-breaking paradigm, which is simple but effective, for the domain incremental learning problem. In particular, the proposed paradigm performs incremental learning without using any expert knowledge from previously learned domains except for general knowledge from pre-trained transformers. Instead, it merely learns the expert knowledge independently domain by domain, through simply prompting the state-of-the-art transformers. The independent knowledge learning mechanism avoids the traditional tug-of-war among different learning sessions and instead plays a win-win game to achieve the best for each learning session. The learned expert knowledge for each domain is finally gathered in a pool for a general inference on varying domains. Our comprehensive empirical study shows that the proposed learning paradigm enjoys a groundbreaking gain (about 30% relative improvement) over the state-of-the-art competitors as well as a great scalability to a large number of incrementally appearing domains.

There remain limitations on the suggested paradigm. For example, we do not obtain appealing results when applying it directly to the class-incremental learning (CIL) tasks. This is because these tasks are more sensitive to the incorrect domain/task identification. Nevertheless, the proposed paradigm still has a high potential to inspire a wide range of research areas. For instance, the suggested prompting can be applied directly to any transfer learning problems, as it essentially learns to prompt the transformers to any upcoming new domains. In addition, it provides an inspiration to zero-shot and few-shot learning, since the proposed S-liPrompts shows superior generalization capability on unseen domains i.e., out-of-domain data (OOD).

Acknowledgments. This work is funded by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (MSS21C002). This work is also supported by the National Natural Science Foundation of China (62076195) and the Fundamental Research Funds for the Central Universities (AUGA5710011522).

References

[1] Afra Feyza Akyürek, Ekin Akyürek, Derry Wijaya, and Jacob Andreas. Subspace regularizers for few-shot class incremental learning. In *ICLR*, 2022.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[3] Hyojin Bahng, Ali Jahanian, Swami Sankaranarayanan, and Phillip Isola. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint arXiv:2203.17274*, 2022.

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020.

[5] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018.

[6] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *ICCV*, 2021.

[7] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[8] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[11] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *CVPR*, 2021.

[12] Jieren Deng, Jianhua Hu, Haojian Zhang, and Yunkuan Wang. Incremental prototype prompt-tuning with pre-trained representation for class incremental learning. *arXiv preprint arXiv:2204.03410*, 2022.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[14] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, 2022.

[15] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[16] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *AISTATS*, 2020.

[17] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[18] Enrico Fini, Victor G Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *CVPR*, 2022.

[19] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *NeurIPS*, 2020.

[21] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[22] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, 2019.

[23] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*, 2018.

[24] Tony Huang, Jack Chu, and Fangyun Wei. Unsupervised prompt learning for vision-language models. *arXiv preprint arXiv:2204.03649*, 2022.

[25] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *NeurIPS*, 2019.

[26] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022.

[27] Patrick Kahardipraja, Brielen Madureira, and David Schlangen. Towards incremental transformers: An empirical analysis of transformer models for incremental nlu. *arXiv preprint arXiv:2109.07364*, 2021.

[28] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *ICLR*, 2018.

[29] Sohail Ahmed Khan and Hang Dai. Video transformer for deepfake detection with incremental learning. In *ACMMM*, 2021.

[30] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[31] Minha Kim, Shahroz Tariq, and Simon S Woo. Cored: Generalizing fake media detection with continual representation using distillation. In *ACMMM*, 2021.

[32] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[33] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is np-hard. In *ICML*, 2020.

[34] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NeurIPS*, 2017.

[35] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[36] 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. *arXiv preprint arXiv:2205.05467*, 2022.

[37] Duo Li, Guimei Cao, Yunlu Xu, Zhanzhan Cheng, and Yi Niu. Technical report for iccv 2021 challenge sslad-track3b: Transformers are better continual learners. *arXiv preprint arXiv:2201.04924*, 2022.

[38] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[39] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, 2019.

[40] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[41] Zhuoyun Li, Changhong Zhong, Sijia Liu, Ruixuan Wang, and Wei-Shi Zheng. Preserving earlier knowledge in continual learning with the help of all previous feature extractors. *arXiv preprint arXiv:2104.13614*, 2021.

[42] Minqian Liu, Shiyu Chang, and Lifu Huang. Incremental prompting: Episodic memory prompt for lifelong event detection. *arXiv preprint arXiv:2204.07275*, 2022.

[43] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.

[44] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020.

[45] 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.

[46] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

[47] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[48] Francesco Marra, Cristiano Saltori, Giulia Boato, and Luisa Verdoliva. Incremental learning for the detection and classification of gan-generated images. In *WIFS*, 2019.

[49] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *CVPR*, 2021.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *IROS*, 2020.

[52] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

[53] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

[54] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[55] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[56] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2018.

[57] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[58] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *ICLR*, 2020.

[59] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

[60] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[61] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *CVPR*, 2021.

[62] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *ECCV*, 2020.

[63] Gido M van de Ven, Zhe Li, and Andreas S Tolias. Class-incremental learning with generative classifiers. In *CVPR*, 2021.

[64] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *CVPR*, 2021.

[65] 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. *arXiv preprint arXiv:2204.04799*, 2022.

[66] 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*, 2022.

[67] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

[68] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, 2021.

[69] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[70] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, 2020.

[71] Pei Yu, Yinpeng Chen, Ying Jin, and Zicheng Liu. Improving vision transformers for incremental learning. *arXiv preprint arXiv:2112.06103*, 2021.

[72] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

[73] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[74] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[75] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: A python toolbox for class-incremental learning, 2021.

[76] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *arXiv preprint arXiv:2109.01134*, 2021.

[77] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022.

[78] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. General incremental learning with domain-aware categorical representations. In *CVPR*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

6 Appendix

In the appendix, we additionally present detailed algorithms of the proposed S-liPrompts, thorough experimental settings of the competing methods and the proposed ones, more ablation experiments, as well as more discussions on related works.

6.1 Algorithm Details

Inference Pipeline. Fig.6 illustrates the inference pipeline of the proposed language-image S-Prompts (S-liPrompts). For the inference phase, we suggest the following steps: 1) feeding the tokens of a given test image into the image transformer to obtain the image feature, 2) utilizing K-NN to search for the nearest centroids obtained by K-Means on training data to the given test image, 3) feeding the image token and the nearest centroids associated image prompts into the image transformer, with the centroid related language prompts and class tokens being fed into the text transformer, 4) calculating the language-image prediction probabilities for the final classification. Note that K-NN is conducted on the image feature space of the fixed pre-trained image transformer without using any prompts. **Algorithm Details.** Alg.1 and Alg.2 show the training/testing process details of the proposed S-liPrompts.

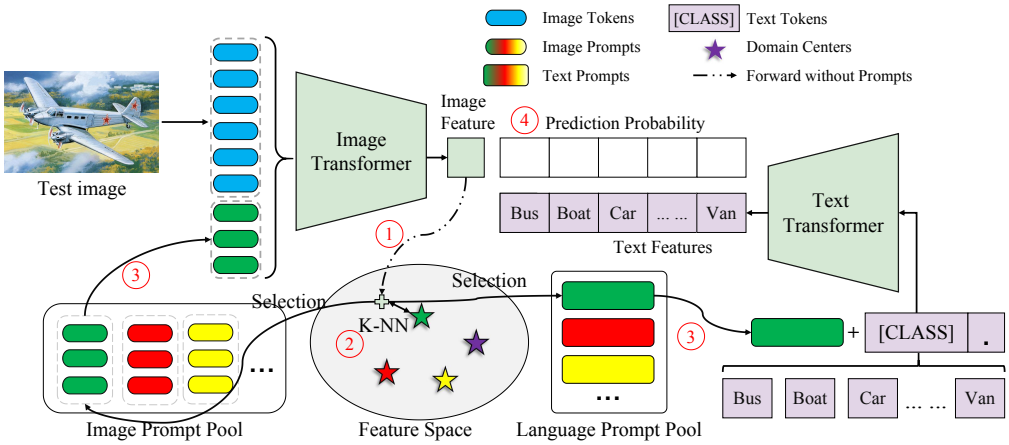

Figure 6: Illustration of the inference pipeline of the proposed S-liPrompts. The displayed indexes correspond to the following four inference steps respectively: 1) obtaining transformer feature of given test image, 2) searching for the nearest centroids obtained by K-Means on training data to the given test image, 3) feeding the image/text tokens and the nearest centroids associated image/text prompts into the image/text transformers, 4) computing the language-image prediction probabilities for the final classification.

6.2 Experimental Details

Implementation details. We implement the proposed S-Prompts in PyTorch with NVIDIA RTX 3090 GPUs. We use the same image encoder architecture of ViT-B/16 [13] for S-iPrompts and S-liPrompts across all

Algorithm 1 Training process of proposed S-liPrompts

Initialization: Training data:$\mathbb{D}_s = \{x_i^s, y_i^s\}_{i=1}^{N_s}, s = 1, 2, ..., S$; Pre-trained image model: f; Pre-trained language model: g; Class embeddings: $c_j, j = 1, 2, ..., C$; Image Prompt Pool: $\mathcal{P}^i = \emptyset$; Language Prompt Pool: $\mathcal{P}^l = \emptyset$; Domain centroids: $\mathcal{M} = \emptyset$; Maximum Iteration: T_{max}; Batch size: N_{bs}; Number of centroids per session: N_K;
for $s = 1, 2, ..., S$ **do**
  Initialize image prompt P_s^i for domain s
  Initialize language prompt P_s^l for domain s
  for $iter = 1, 2, ..., T_{max}$ **do**
    Extract a mini-batch samples $\{x_i^s, y_i^s\}_{i=1}^{N_b}$ from Traning Data $\mathbb{D}_s$;
    Prepend x_i^s with corresponding prompts P_s^i by $[x_i^s, P_s^i, x_{cls}]$;
    Prepare input for text encoder by $t_j = \left\{P_s^l, c_j\right\}, j = 1, 2, ..., C$;
    Generate class representation by $g(t_j)$;
    Compute the prediction probability by $p = \frac{\exp(\langle f([x_i^s, P_s^i, x_{cls}]), g(t_j)\rangle)}{\sum_{k=1}^{C} \exp(\langle f([x_i^s, P_s^i, x_{cls}]), g(t_k)\rangle)}$;
    Calculate Cross-Entropy loss by $\mathcal{L}_B = \mathcal{L}(\sigma(p), y_i^s)$;
    Update $\theta_{P_s^i} = \theta_{P_s^i} - \eta \bigtriangledown_{P_s^i} \theta_{P_s^i}$;
    Update $\theta_{P_s^l} = \theta_{P_s^l} - \eta \bigtriangledown_{P_s^l} \theta_{P_s^l}$;
  end for
  Initialize a temporal memory buffer B to store features
  for $i = 1, 2, ..., N_s$ **do**
    Extract a sample x_i^s from training data $\mathbb{D}_s$
    $B \leftarrow f([x_i^s, x_{cls}])$
  end for
  Run K-Means on B to get N_K domain centroids $\{m_i^s\}_{i=1}^{N_K}$
  $\mathcal{M} \leftarrow \{m_i^s\}_{i=1}^{N_K}$
  $\mathcal{P}^l \leftarrow P_s^l$
  $\mathcal{P}^i \leftarrow P_s^i$
end for

Algorithm 2 Testing process of proposed S-liPrompts

Initialization: Test image:x; Pre-trained image model:f; Pre-trained language model:g; Class embeddings:$c_j, j = 1, 2, ..., C$; Image Prompt Pool:$\mathcal{P}^i$; Language Prompt Pool:$\mathcal{P}^l$; Domain centroids:$\mathcal{M}$; Number of domain centroids:N_M; Number of neighbors for K-NN:N_K;

Calculate image feature $f_x = f([x, x_{cls}])$
for $j = 1, 2, ..., N_M$ **do**
  Extract the j-th domain centroid m from $\mathcal{M}$
  Calculate L1 distance $d_j = \sum |f_x - m|$
end for
Arrange the calculated Euclidean distances $[d_j]_{j=1}^{N_M}$ in a non-descend order
Take the first N_K distances from the sorted list
Run K-NN to find the corresponding domain identification s
Prepend x with corresponding prompts P_s^i by $[x, P_s^i, x_{cls}]$
Prepare input for text encoder by $t_j = \left\{P_s^l, c_j\right\}, j = 1, 2, ..., C$
Generate class representation by $g(t_j)$
Compute the prediction probability by $p = \frac{\exp(\langle f([x, P_s^i, x_{cls}]), g(t_j)\rangle)}{\sum_{k=1}^{C} \exp(\langle f([x, P_s^i, x_{cle}]), g(t_k)\rangle)}$ for classification

Table 8: Average accuracies (%) of using random domain selection and our proposed domain selection on CDDB and DomainNet for domain incremental learning. The two domain selection cases are both based on the proposed S-liPrompts.

	Random Domain Selection	Proposed Domain Selection	Relative Improvement
CDDB	80.12	88.65	8.53
DomainNet	49.94	67.78	17.84

Table 9: Average accuracies (%) of the originally proposed S-liPrompts (each inference uses one single domain-selected CLIP prediction) and its ensemble version (each inference utilizes the voting strategy on all domain-based CLIP predictions) on the CDDB and DomainNet datasets for domain incremental learning.

	Voting for Ensemble	Proposed Domain Selection	Relative Improvement
CDDB	65.47	88.65	23.18
DomainNet	58.85	67.78	8.93

tasks/domains. The text encoder of S-liPrompts is the same as that of CLIP [54]. In our experiments, the total class number C is 2 for CDDB [36], 50 for CORe50 [45], and 345 for DomainNet [52] respectively. For all the experiments in the main paper, the length of a single image prompt L_i is 10, and the length of one language prompt L_l is 16. The embedding dimension D_i is 768 for both of the used ViT and CLIP. The dimension of word embedding c_i and the embedding dimension of text transformer D_l are both 512. The implementation of K-Means and K-nearest neighbors (KNN) are both based on scikit-learn [50]. The distance measure is the L1 distance.

Training details. Our S-Prompts is insensitive to the setting of hyper-parameters and we do not tune much. For all benchmarks, we adopt SGD optimizer with a momentum of 0.9, an initial learning rate of 0.1, a batch size of 128, and a cosine annealing scheduler [47]. The number of epochs is different for different datasets, which should be large enough to fit the training sets. Specifically, we use 50 epochs for CDDB, 10 epochs for DomainNet and CORe50. All the training images are resized to 224×224. We only adopt simple data augmentation as other methods [48, 22], e.g., horizontal flip and random crop.

Memory overhead. The number of context vectors for CLIP is 16 and the dimension is 512, and thus the number of parameters of language prompts for each session is $34,816$. The length of image prompts is 10 and the dimension is 768, which has $33,792$ parameters. The dimension of K-Means cluster centers is 512 for S-liPrompts and 768 for S-iPrompts. In our experiments, we use 5 centers for all the three used benchmark datasets, and hence it adds $12,288$ to $18,432$ parameters per session. Accordingly, S-iPrompts only adds $52,224$ parameters to the original pre-trained ViT-B/16 model, leading to a paltry 0.05% total parameter increase. Similarly, S-liPrompts adds $80,896$ parameters per session. Compared with pre-trained CLIP (with ViT-B/16 image encoder), it has a total parameter increase of 0.03%.

Baseline methods. We implement EWC [32] and LwF [40] with pre-trained ViT-B/16 using the pycil[9] toolbox [75]. The hyper-parameters remain the same as those in their original paper. DyTox [14] and L2P [76] are two main baselines in this paper, and we list the parameters below for reference. We use the official implementation[10] of DyTox for its evaluation on CDDB, CORe50 and DomainNet. The original backbone of DyTox is a shallow ConViT [15] (5 self-attention blocks and a task-attention block) without pre-training. For a fair comparison with other methods, we use a pre-trained base-ConViT[11] with first 5 layers as the backbone. The model size of 5 layers base-ConViT (78M) is almost the same as the base-ViT (86M) used in other methods including L2P and ours. We use the learning rate of 0.0001 and 50 epochs for CDDB to get the proper performance. For DomainNet and CORe50, the learning rate is 0.001 and the number of epoch is 20. All the other parts of DyTox remain the same as those of the official implements, and the changes of learning rate and the number of epochs are for getting better performance. We evaluate the official L2P codebase[12] directly on CDDB and DomainNet by tuning its key parameters, i.e., learning rate, trade-off parameter λ, prompt pool size, N value for key selection. The learning rate for CDDB is tuned to 0.1, and that for DomainNet is 0.8. We use 30 epoch for CDDB and 20 for DomainNet. For both datasets, the prompt pool size is tuned to 20, and the number of key selection is 5. The trade-off parameter λ is adjusted to 0.001.

[9] `https://github.com/G-U-N/PyCIL`

[10] `https://github.com/arthurdouillard/dytox`

[11] DyTox [14] originally suggests training base-ConViT from scratch. But this case's performance (about 56%) is much lower than that (86.21%) of training from the pre-trained base-ConViT (78M), which is also higher than that (84.20%) of training from the pre-trained base-ViT (86M), on the CDDB-Hard dataset.

[12] L2P [76] merely releases the exemplar-free code at `https://github.com/google-research/l2p`. We tried to adapt it to the cases using exemplars, but they generally work badly, e.g., about 59% on CDDB-Hard.

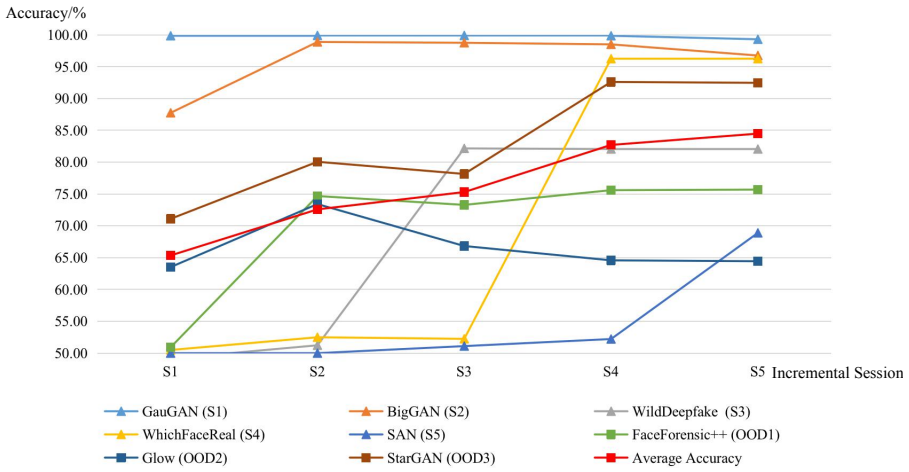

Figure 7: Accuracy curves of the proposed S-liPrompts generalized from five seen CDDB-Hard domains (S1-S5: GauGAN, BigGAN, WildDeepfake, WhichFaceReal, SAN) to three unseen domains (out-of-domain) OOD1-OOD3: FaceForensic++, Glow, StarGAN, which were used by CDDB [36]. The value (S1-S5) along the horizontal direction indicates the S-liPrompts model when trained until the given seen domain data. For example, S3 means that the model was trained along the stream of S1, S2 and S3 sequentially. The curves with different colours indicate the detection performance on the eight seen/unseen domains (S1-S5 and OOD1-OOD3) respectively, and the red one shows the average accuracy.

6.3 More Experiments

Random Domain Selection vs. Proposed Domain Selection. In addition to the comparison on CDDB (Table 4 in the main paper), we further compare the case of performing random domain selection against that of using our proposed domain selection on DomainNet. Table 8 reports the results on these two datasets. It shows the consistently remarkable relative improvements (by more than 8% on CDDB and about 18% on DomainNet) of the proposed domain selection over the random domain selection. This considerable improvement justifies the necessity of the proposed domain selection, which attributes to the significance of the proposed independent prompting paradigm. The out-of-distribution (OOD) experiment in Table 7 verifies that the random domain selection (in Table 8) has promising results mainly due to the strong generalization of the proposed language-image prompt learning scheme over the CLIP model. Nevertheless, the results can be further improved significantly using the proposed domain selection 8, showing the clear superiority of the proposed independent prompting paradigm.

Domain Ensemble vs. Proposed Domain Selection. We further study one of the most popular ensemble methods (i.e., voting) on the proposed S-liPrompts. Table 9 summarizes the comparison of the proposed S-liPrompts against the voting-based version in terms of average accuracies on CDDB for deepfake domain incremental learning. The significant relative increases (about 23% on CDDB, 9% on DomainNet) show that the proposed S-liPrompts method favors the proposed domain selection method that chooses the most expertised (i.e., the most related domain-based) prompting model for the inference on each test sample. The excellent performance of the proposed domain-based prompting models verifies the significance of the proposed independent prompting scheme. For the voting-based inference on each test sample, we first feed all the learned domain-based prompts to the CLIP models one by one resulting in multiple predictions, and we then use the majority voting strategy on the individual results to get the final prediction. The clearly superior performance of the proposed S-liPrompts with domain selection mainly stems from the most expertised model for the given test sample. Except for this model, the rest ones are less expertised for the given test sample, and they might be dominant to corrupt the final prediction when doing the majority voting. In this case, the voting scheme could lead to clear performance degradation. Moreover, it requests running all the learned CLIP-based prompting models for the voting on each test sample, and thus it is much more time-consuming than the proposed inference scheme.

Test on Unseen Domains. Based on Table 7 in the paper, Fig.7 additionally shows the detection accuracy curves of the proposed S-liPrompts when applied to three unseen domains (FaceForensic++, Glow and StarGAN). The S-liPrompts models were trained on the five seen domains (GauGAN, BigGAN, WildDeepfake, WhichFaceReal and SAN) from CDDB-Hard sequentially. As studied in the main paper, the results in Fig.7 demonstrate that the proposed S-liPrompts owns an excellent generalization ability on the unseen domains.

Successes and Failures. Fig.8 lists some success cases which are predicted to incorrect domains but are classified correctly in the end using S-liPrompts on CDDB-Hard. It is interesting to see that most of them are predicted to a semantically close domain. For example, as shown in the first two rows in Fig.8, some faces from

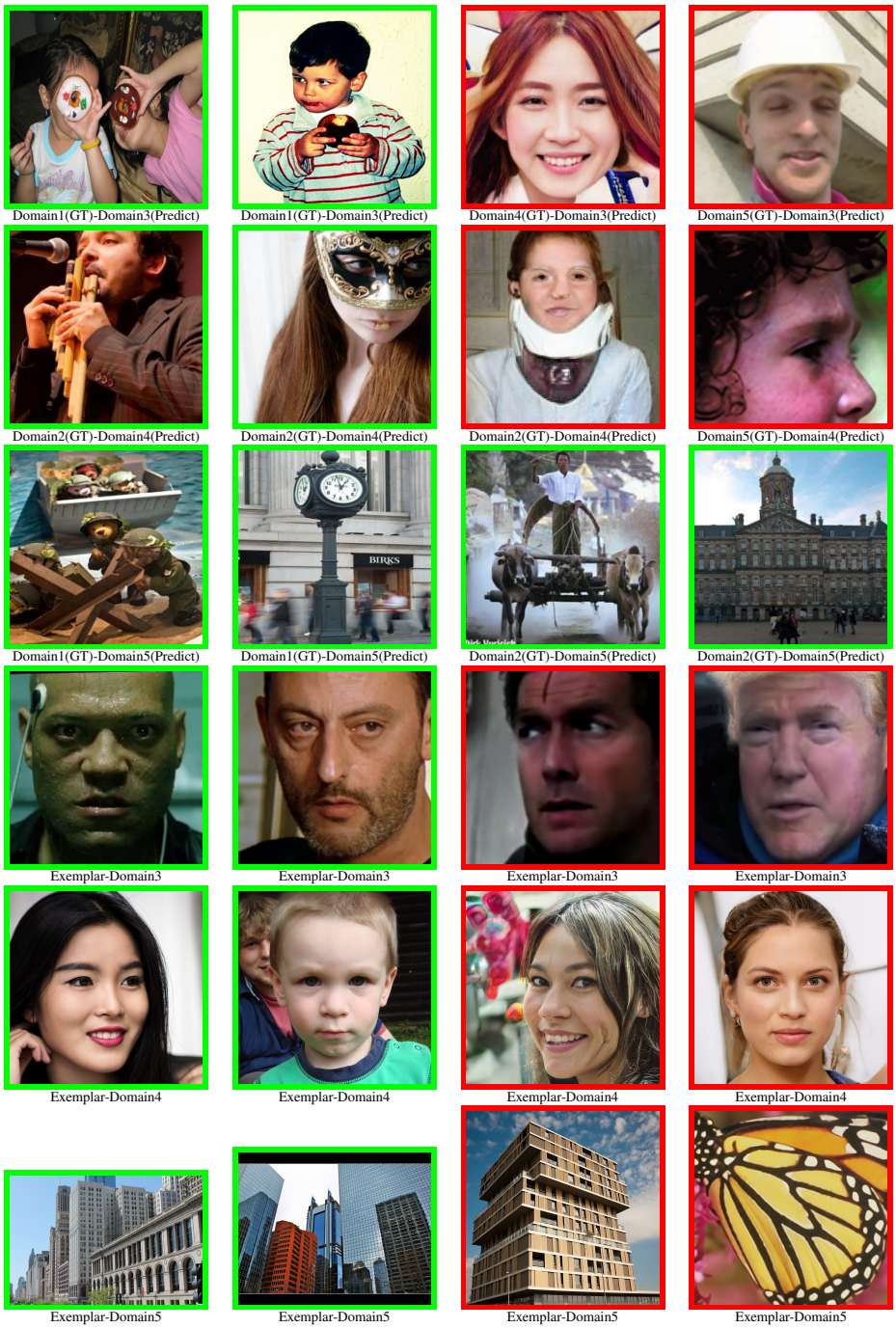

Figure 8: Success examples that are predicted to incorrect domains but are classified correctly in the end using the proposed S-liPrompts on CDDB-Hard. DomainA(GT)-DomainB(Predict): ground truth domain is A, and predicted domain is B. Exemplar-DomainA: exemplars from domain A. Domain1-Domain5: GauGAN, BigGAN, WildDeepfake, WhichFaceReal, SAN. All images with a green boundary are real, and all images with a red boundary are fake.

Domain 1, 2, 4, 5 are predicted to either Domain 3 (WildDeepfaces) or Domain 4 (WhichFaceReal), where all the images are of faces as shown by the exemplars of Domain 3 and Domain 4 (in row 4 and row 5 of Fig.8). In this case, they are very likely to be classified correctly, due to the excellent learning power of S-liPrompts on Domain 3 (WildDeepfaces) and Domain 4 (WhichFaceReal). The same observation can be derived on those images at row 3 in Fig.8. As their architecture details are very similar to those in the exemplars of Domain 5 (SAN), they are assigned to Domain 5 that still leads to the right classification. Fig.9 shows some classical failure

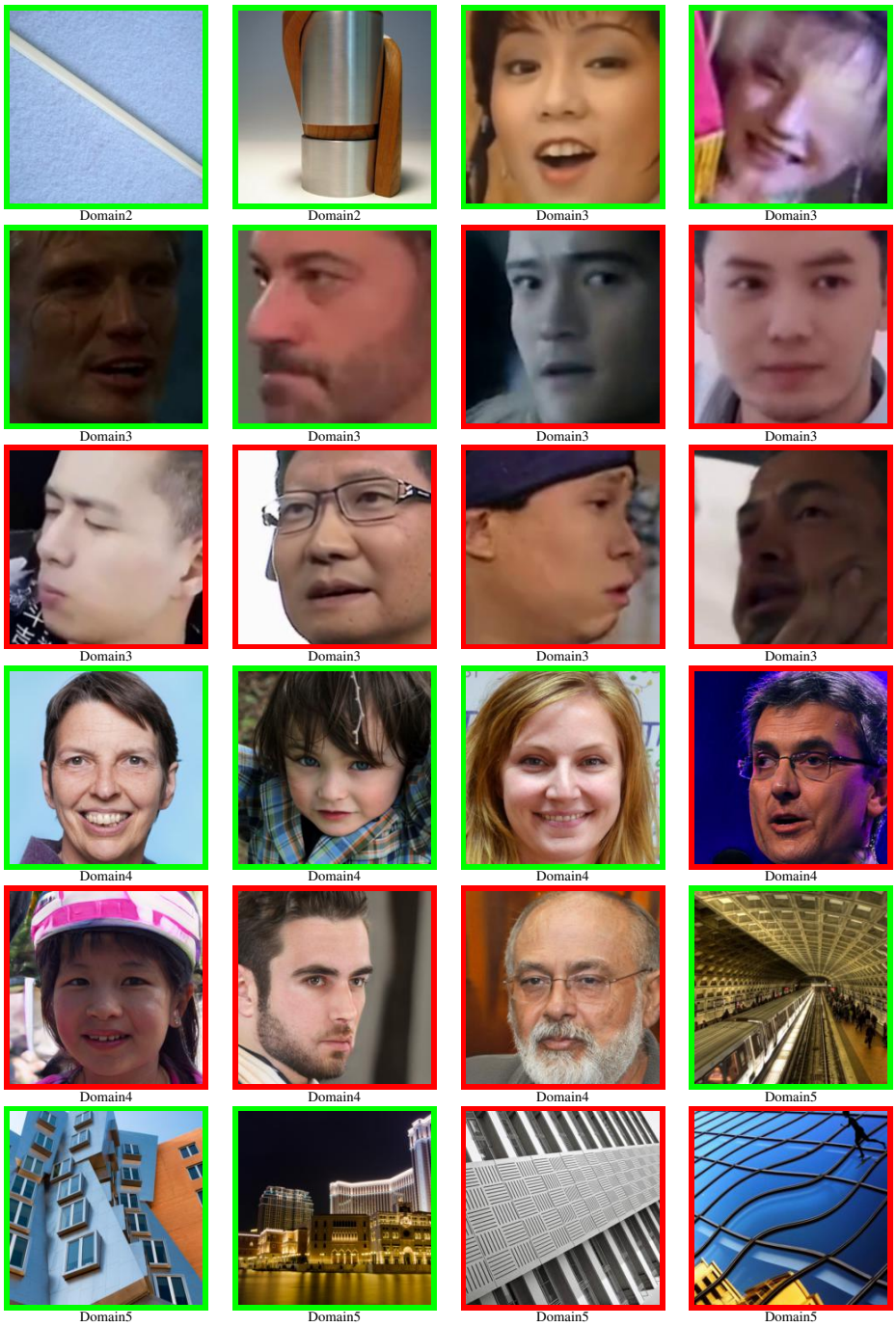

Figure 9: Failure examples that are predicted to correct domains but are classified incorrectly using the proposed S-liPrompts on CDDB-Hard. Domain1-Domain5: GauGAN, BigGAN, WildDeepfake, WhichFaceReal, SAN. All images with a green boundary are real, and all images with a red boundary are fake.

cases. As we can see in Table 6, the challenging cases are mostly on the domains of WildDeepfake and SAN. For the SAN cases, we can see that some irregular-looking real buildings (e.g., wired structures) are misclassified as fake, while those fake buildings with realistic-looking structures are classified as real, as displayed in Fig.9. For the WildDeepfake domain, there exist a large number of real faces that are often of blurry or occluded or low contrast or non-frontal. These real faces are generally hard for real/fake detection. On the other hand, the fake faces are often of the same quality. This highly increases the challenge for the deepfake detection.

Feature Spaces of Competing Methods. Fig.10 uses t-SNE to visualize the resulting feature spaces of pre-trained ViT, CLIP, the two most competing methods DyTox [14] and L2P [76] as well as our S-iPrompts and

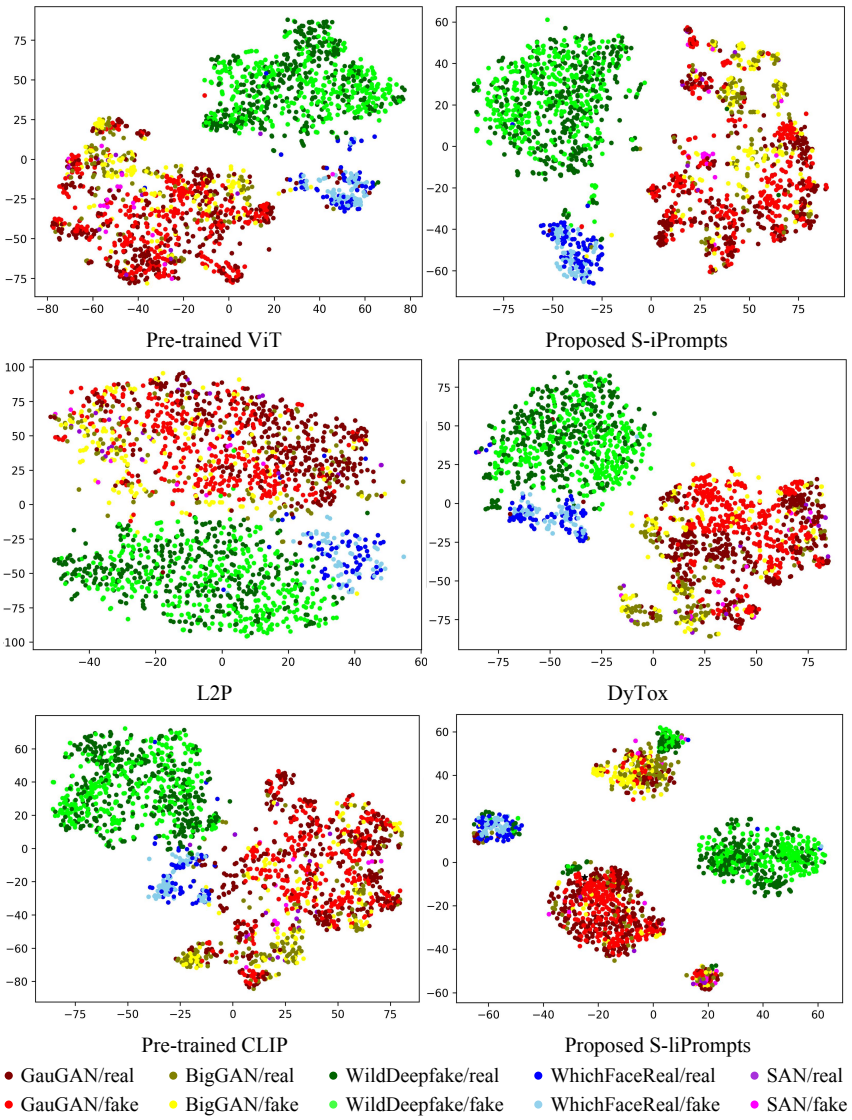

Figure 10: t-SNE visualization on the resulting feature spaces of pre-trained ViT, proposed S-iPrompts, DyTox [14], L2P [76], pre-trained CLIP, and proposed S-liPrompts on the CDDB-Hard dataset.

Table 10: Detection accuracies (%) of the proposed S-liPrompts/S-iPrompts and the main competing methods (L2P and DyTox) on the CDDB dataset for deepfake domain incremental learning. **Bold**: best results, Underline: second best results.

	GauGAN	BigGAN	WildDeepfake	WhichFaceReal	SAN	Average	Min	Max
S-liPrompts	99.30	96.75	82.06	96.25	68.89	**88.65**	**68.89**	**99.30**
S-iPrompts	90.30	81.88	72.76	84.25	43.30	74.50	43.30	90.30
L2P	80.73	62.60	58.98	57.48	46.59	61.28	46.59	80.73
DyTox	48.91	50.00	59.19	50.00	50.00	51.62	48.91	59.19

S-liPrompts on CDDB-Hard. As we discussed in the main paper, dependent learning across domains is very likely to mix up the subspaces of old/new domains. The t-SNE results of L2P and DyTox can verify this to some extent. In contrast, our proposed S-Prompts (especially S-liPrompts) is capable of producing much more separable subspaces. This demonstrates that our independent learning could reach a more favorable feature space where subspaces of different domains are pushed far away from each other. This property could lead to better performances that were reported in the main paper, showing the clear superiority of the proposed independent prompting paradigm over the commonly-respected dependent learning principle.

Table 11: Precisions (%) of the proposed S-liPrompts/S-iPrompts and the main competing methods (L2P and DyTox) on the used CDDB dataset for deepfake domain incremental learning. Here precision is the ratio tp/(tp + fp) where tp is the number of true positives and fp is the number of false positives. **Bold**: best results, Underline: second best results.

	GauGAN	BigGAN	WildDeepfake	WhichFaceReal	SAN	Average	Min	Max
S-liPrompts	99.80	98.45	81.97	96.54	75.76	**90.50**	**75.76**	**99.80**
S-iPrompts	90.95	83.64	73.21	84.08	43.75	75.13	43.75	90.95
L2P	83.10	77.29	65.39	57.79	66.67	70.05	57.79	83.10
DyTox	52.99	50.00	85.86	50.00	50.00	57.77	50.00	85.86

Table 12: Memory overheads of the proposed S-iPrompts (ViT-based), S-liPrompts (CLIP-based), the ViT-based prompting methods (DyTox and L2P), and ViT-based non-prompting ones (Others without expansion) on the CDDB dataset. In the setup, there are 5 sessions, each of which includes 2 classes (real and deepfake classes). The average increase corresponds to the parameter increase per session.

	DyTox	L2P	Proposed S-iPrompts	Proposed S-liPrompts	Others
Base Model	86M	86M	86M	201M	86M
Total Increase	7.09M	92.16K	0.26M	0.40M	0
Average Increase	1.42M	18.43K	52.22K	80.89K	0
Relative Increase (Increase/Base)	1.65%	0.02%	0.05%	0.03%	NA

Metrics to Measure Separation Degree. We used t-SNE as it is a popular choice for visualization. Though the significant superiority of our proposed S-liPrompts is shown clearly via t-SNE, it is mainly in terms of domain separation rather than class separation. This phenomenon is mainly from the fact that t-SNE visualization is limited to the 2-dim projection of the original high-dimensional data. Therefore, apart from using t-SNE visualization, we additionally use quantitative metrics to measure the degree of class separation domain by domain. Accordingly, we (re-)collected the classification/detection accuracies in Tables 1 & 2 (main paper), which reflects the average domain-wise accuracies and thus can be a favorable metric for this purpose. Following the related paper [36], we additionally introduce a precision based metric to measure the domain-wise class separation degree. The results in Table 10 and Table 11 demonstrate the consistent superiority of the proposed S-iPrompts/S-liPrompts methods in terms of the class separation using such various metrics.

Memory Consumption Comparison. We further report the memory overheads of the proposed S-iPrompts/S-liPrompts methods (ViT/CLIP-based), the ViT-based prompting methods (DyTox and L2P) as well as those ViT-based non-prompting methods (without expansion). As the non-prompting methods do not expand their architectures for the increasing tasks, their model parameters keep the same for the learning process. Therefore, the main comparison is on the prompting based methods including ours. As shown in Table 12, the proposed S-iPrompts increases the model parameters for the domain-specific prompts that are of 52.22K (0.05% relative increase) for each domain (session), and the proposed S-liPrompts's increase is of 80.89K (0.03% relative increase) per session. By comparison, DyTox needs to add one task-attention block over the ViT-based model and dynamically expands the domain-specific tokens (prompts) every session. L2P is like ours directly using ViT-based model without adding any neural network blocks, and it assigns a certain memory to save the pool of used prompts once for the initialization. From the results, we can see that DyTox's increase is the most, and that of L2P is the least. The relative increase ratio of the proposed S-liPrompts/S-iPrompts is very close to that of L2P, while obtaining clearly better performances than the other competing methods.

Backbones Used by Competing Methods. As presented in the main paper, to compare fairly, we used the same backbone (i.e., base-ViT [13]) for all the real competitors (except for DyTox that used a more advanced ViT model, i.e., ConViT [15]) as well as the proposed methods (S-iPrompts, S-liPrompts) for all the experiments (Table 1, 2, 3). In Footnote 3 of the supp. material, we additionally state that we reported the results of DyTox with base-ConViT [15] rather than base-ViT [13], since the performance (86.21%) of the former model is better than that (84.20%) of the latter model on CDDB. Hence, we follow the suggestion of the original DyTox paper to use a more advanced ViT model (ConViT) for DyTox in the main paper. Besides, DyTox with base-ViT performs very badly on CDDB with the average accuracy being 48.89%, which is even worse than that of the random guess on CDDB. We further evaluate DyTox with base-ViT on the other two datasets (CORe50 and DomainNet), and we find that the ViT-based exemplar-free DyTox still fails on these two datasets as discovered on the ConViT-based DyTox in the main paper. This is mainly because DyTox requests a distillation on exemplars, which are more demanded to balance the multiple classes from new/old domains on CORe50 and DomainNet, for a promising prompt learning (Footnote 7 in the main paper). Therefore, we report the results of the better DyTox (DyTox with base-ConViT) in the main paper, and we here merely report its results using examplars on the three benchmark datasets. The results of Table 13 show that DyTox with base-ConViT generally outperforms (or at least comparable with) DyTox using base-ViT on the three used datasets. Besides, CaSSLe is a self-supervised incremental learning method, and replacing supervised trained weights trained on ImageNet is not suitable. Hence, we use its officially released backbone.

Table 13: Accuracies (%) of the exemplar-based DyTox using two different backbones (i.e., base-ViT [13] and base-ConViT [15]) on the three benchmarks.

	DyTox using Pre-trained base-ViT	DyTox using Pre-trained base-ConViT
CDDB	84.20	86.21
CORe50	80.11	79.21
DomainNet	60.83	62.94

Table 14: Accuracies (%) of the proposed S-iPrompts and the main competing exemplar-free methods (L2P and DyTox) on the three used datasets. Note that CDDB is for continual binary-class classification, and CORe50 and DomainNet are both for continual multi-class classification. DyTox generally fails for the two multi-class classification tasks (CORe50 and DomainNet), as it requests a distillation on examplars for the more challenging balance problem among multiple classes from new/old domains.

	L2P	DyTox	Proposed S-iPrompts	Relative Improvement
CDDB	61.28	51.27	74.51	13.23
CORe50	78.33	Fails	83.13	4.80
DomainNet	40.15	Fails	50.62	10.47

Contribution of Independent Image-end Prompting. Table 14 reports the results that are copied from Table 1, 2, 3 of the main paper. The considerable relative improvement (13% on CDDB, 5% on CORe50, 10% on DomainNet) of the proposed S-iPrompts (independent prompting) over the two main competitors L2P and DyTox (dependent prompting) justifies the significant superiority of the contribution of independent image prompting. We conduct this comparison fairly in the scenario of exemplar-free domain incremental learning (DIL), which is the main aim of our paper, i.e., better data security, privacy and less memory consumption. In the context of exemplar-free DIL, DyTox collapses on CORe50 and DomainNet, because it requires a distillation on selected exemplars for a better balance among multiple classes from new/old domains on these two datasets, which seems essential for a promising dependent prompting [14]. In addition, we also compared the proposed S-iPrompts against those state-of-the-art exemplar-based methods for a reference. We are delighted to find that S-iPrompts (exemplar-free) outperforms the exemplar-based DyTox clearly (about 4% increase) and the exemplar-based L2P (about 2% improvement) on CORe50, although it is outperformed by the exemplar-based DyTox on CDDB and DomainNet using a large number of exemplars (e.g., 17,250 exemplars on DomainNet). The cost of using large exemplar data is generally not favorable in the real-world scenarios.

Contribution of Independent Language-Image Prompting. Table 15 presents the results that are mainly from Table 4 in the main paper, with additional notes in the second and third columns for further clarification. As shown in Table 15, performing CLIP or CLIP+S-liPrompts directly cannot achieve improved performance over S-liPrompts using ViT as a backbone. In contrast, applying the proposed language-image prompting scheme on the CLIP model makes remarkable improvement. From the comparison between S-liPrompts (final, K=5) and S-liPrompts (fixed Language Prompt weights), we can find the contribution of language-image prompting brings an significant improvement of about 16% over the proposed S-iPrompts that is based on the contribution of independent image-end prompting. By comparing S-liPrompts (fixed Image Language Prompt Length) against S-liPrompts (fixed Language Prompt weights), we can see the considerable superiority (about 8% relative improvement) of S-iPrompts (independent image-end prompting) over L2P-like learning paradigm (dependent image-end prompting), which verifies the significant contribution of independent image-end prompting again. Additionally, the results in Table 1, 2, 3 of the main paper show that our best model S-liPrompts supasses the best of the state-of-the-art exemplar-free methods significantly (30% relative improvement on average) for the three standard DIL benchmark datasets, and even outperforms them clearly (an average increase of 6%) when they use exemplars.

In addition, we evaluate the ablation on CORe50 and DomainNet. Table 16 shows the results. The observation on these two datasets is consistent with that on CDDB. In particular, for DomainNet, the considerable improvement of the final S-liPrompts model demonstrates its clear superiority over the ablated model. In comparison, the improvement on CORe50 is clear but relatively smaller for the CLIP case, while the increase is remarkable when comparing our ViT-based method (S-iPrompts) against ViT-based L2P on CORe50. This might be because CORe50's domain gaps are not as large as those of CDDB and DomainNet (the CORe50 domains are captured by the same Kinect sensor under different backgrounds and lighting [45]). On CORe50, we find most of the evaluated methods have much less forgetting compared to those on CDDB and DomainNet due to the smaller domain gaps of CORe50. In this case, the more powerful language-image prompting scheme might share beneficial knowledge from those similar domains, which does not harm the performance a lot. In contrast, the relative improvements are very remarkable on CDDB (Table 15) and DomainNet (Table 16), showing the significant superiority of the proposed prompting method in the cases with large domain gaps.

Table 15: Accuracies (%) of ablating components of the proposed S-liPrompts on the CDDB dataset. Note that we use the official implements[13] of CLIP to do Zero-shot classification on three benchmarks without any change. Here we use the text templates of ImageNet as their implementation for CLIP (Zero-shot). LIP: Language-Image Prompt, LP: Language Prompt.

Method	Prompting Scheme	Corresponding Method	CDDB
CLIP (Zero-shot)	handcrafted language prompts	Vanilla CLIP	49.52
S-liPrompts (fixed LIP Length)	dependent image prompts tuning + dependent language prompts tuning	CLIP + L2P	64.90
S-liPrompts (fixed LP weights)	independent image prompting + dependent language prompts tuning	CLIP + S-iPrompts	72.29
S-liPrompts (final, K=5)	independent image prompting + independent language prompting	CLIP+ S-liPrompts	88.65

Table 16: Accuracies (%) of ablating components of the proposed S-liPrompts on the CORe50 and DomainNet datasets. LIP: Language-Image Prompt.

Method	Prompting Scheme	Corresponding Method	CORe50	DomainNet
S-liPrompts (fixed LIP Length)	dependent image prompts tuning + dependent language prompts tuning	CLIP + L2P	85.51	56.79
S-liPrompts (final, K=5)	independent image prompting + independent language prompting	CLIP+ S-liPrompts	89.06	67.78

6.4 More Discussions on Related Works

As presented in the main paper, there exist some expansion-based continual learning methods like [69, 34, 17, 59, 25, 19, 39]. They either dynamically expand the network architectures or reshape their internal structures task by task. This results in the following three issues: 1) Though they generally separate the structure and parameter learning significantly, they still request effective sharing of parameters among tasks. The sharing-driven network tuning inevitably results in a non-trivial trade-off (or so-called tug-of-war) between old task learning and new task learning [56, 33]. 2) They generally require that the task indexes should be provided for the inference. 3) They often result in a significantly growing number of model parameters for a large number of tasks.

To overcome the second issue of the expansion-based methods, [68, 41] suggest learning a single classifier on a concatenation of all resulting feature maps from different subsets of parameters, which connects all the tasks/domains. Nevertheless, it suffers from the dramatically increased memory overhead issue when handling a long task sequence, which thus requires a complex post-processing for pruning. To overcome this large memory consumption (i.e, the third issue mentioned above), [14] suggests expanding prompts instead over a transformer network. The prompt expansion leads to a small amount of parameter increase. Yet, as discussed in the main paper, like many of traditional methods, the dependent learning across tasks always goes for a tug-of-war (i.e., the first issue mentioned above), where it can only achieve a trade-off between transferring and forgetting and thus it is almost impossible to achieve the best for every task/domain. By comparison, our paper breaks the commonly-respected learning principle and instead establishes a new independent prompting paradigm to play a win-win game, where it can achieve the best for each domain. In addition, our proposed S-Prompts can address another two issues well with a rough domain identifier and a negligible parameter increase through the proposed prompting paradigm.

There have been also emerging some other transformer-based incremental learning methods like [71, 27, 37], and other incremental prompting methods like [42, 65][14]. The transformer-based methods without prompting demand much more tuning on the whole network than the prompting methods like [76] and ours. Also, they keep applying traditional dependent learning paradigm to play the tug-of-war game. The prompting method [42] suggests learning episodic memory prompts to explicitly carry previously learned knowledge to subsequent tasks for class-incremental event detection. However, it still follows the conventional learning paradigm to perform experience replay and knowledge distillation for the trade-off between transfer and interference.

[14] [42, 65] are two concurrent works that are recently available in arXiv but not formally published up to our knowledge.