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

Anonymous Author(s)
Affiliation
Address
email

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 brand-new language-image prompt learning approach. Owning an excellent scalability (0.05% 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.

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 as *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.

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.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

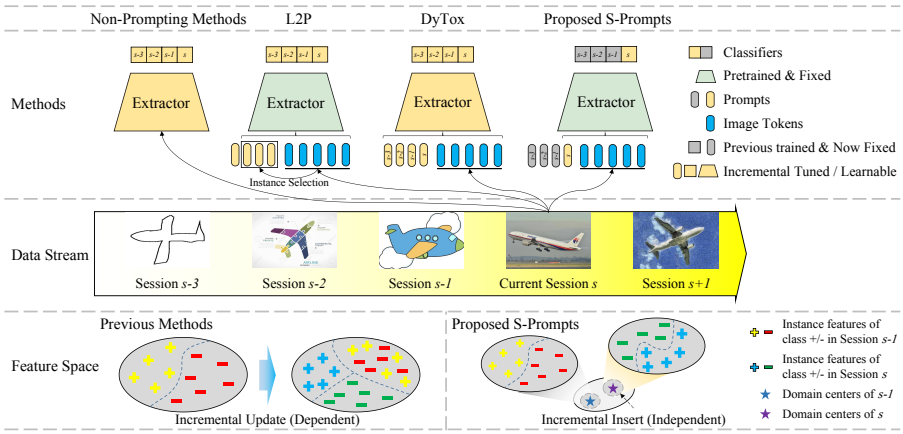

Figure 1: Comparison of the proposed S-Prompts paradigm against non-prompting methods, and prompting methods (L2P [54], DyTox [14]). Exiting methods generally learn sequential tasks/sessions/domains dependently, producing a single feature space where the classes from different domains are less separable (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).

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 moved on 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*[1] 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, hence a rehearsal buffer is no longer mandatory to mitigate forgetting. For instance, two recent prompting methods [14, 54][2] 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 prompt shares 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 [54], 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, 54] 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 [46, 29]. 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 prompting 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-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

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

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

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 sample, 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) [44]. Particularly, we suggest the CLIP-based approach that exploits a brand-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 *joint 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 brand-new 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., *0.05%* 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 [47, 45, 38, 7, 20, 24, 9] or replaying synthesized data from previous tasks using generative models [49, 52]. Distillation-based methods (e.g., [34, 45, 19, 5, 55, 57, 51, 36, 40, 11]) apply the technique of knowledge distillation [18] to mitigate catastrophic forgetting. In addition, there exist other kinds of methods like gradient-based methods (e.g., [46, 59, 15, 48, 50, 53]), regularization-based ones (e.g. [28, 60, 2, 1]), and expansion-based ones (e.g. [56, 30, 22]).

Recently, some of work like [39, 27, 25, 32, 54, 16, 63] 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 [39].

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 [54]. There have been emerging a bunch of prompting methods, such as [31, 33, 23, 61, 62, 3, 21, 54, 14], to name a few[3]. 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 [54] 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.

[3]For a comprehensive survey on the prompting methods, please refer to [35].

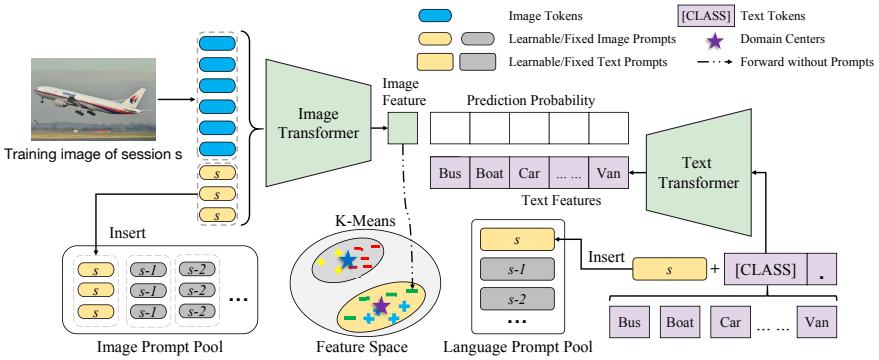

Figure 2: Illustration of the proposed language-image S-Prompts (S-liPrompts) learning approach. $s, s-1, s-2$ denote the session/domain indexes. For the inference, we simply adopt the following steps: 1) feeding the tokens of given test image into the image transformer to get the image feature, 2) using 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 centroids 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 wrong 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) [44]. 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, when the domain center is selected the associated prompts and the corresponding classifier is also selected 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 [44] 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 [61]. 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 $t_j = \{P_s^l, c_j\}$, where c_j is the word embeddings of the j-th class names [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 $\mathcal{P}^l = \{P_1^l, P_2^l, ..., P_S^l\}$, 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.

Compared with the traditional FC-based classifiers, there are two main advantages using the language model for the computation of classification weights. Firstly, the language model can provide 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 incremental session, while the class names are shared, which highly reduces the consumption of memory.

4 Experiments

Datasets. We perform experiments on three standard DIL benchmark datasets. They are CDDB [32], CORe50 [37], and DomainNet [42]. CDDB [32] 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 [37] 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 [42] 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 [16]. We follow [32] to report the average forward detection accuracy and the average forgetting degree on CDDB-Hard. For CORe50 and DomainNet, we follow [54] and [16] 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 [28], LwF [34] ER [8], GDumb [43], BiC [55], DER++ [4] and Co2L [6], as well as the recently published prompting methods DyTox [14] and L2P [54] and self-supervised learning method CaSSLe [16]. Among them, since EWC [28], LwF [34], L2P [54], CaSSLe [16] 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 the best performances. To compare fairly, we use almost the same ViT models (i.e., ViT-B/16 [13]) for all the real competitors as well as our S-iPrompts/S-liPrompts. The proposed S-liPrompts's text-end encoder is the same as that of CLIP [44]. For more implementations details, please refer to the supplementary material.

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 [54]. * denotes results copied from [32].

Method	Buffer size	Average Acc (↑)	Forgetting (↑)
LRCIL [41]	100/class	76.39*	-4.39*
iCaRL [39]		79.76*	-8.73*
LUCIR [19]		82.53*	-5.34*
LRCIL [41]	50/class	74.01*	-8.62*
iCaRL [39]		73.98*	-14.50*
LUCIR [19]		80.77*	-7.85*
DyTox [14]		86.21	-1.55
EWC [28]	0/class	50.59	-42.62
LwF [34]		60.94	-13.53
DyTox [14]		51.27	-45.85
L2P [54]		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	-

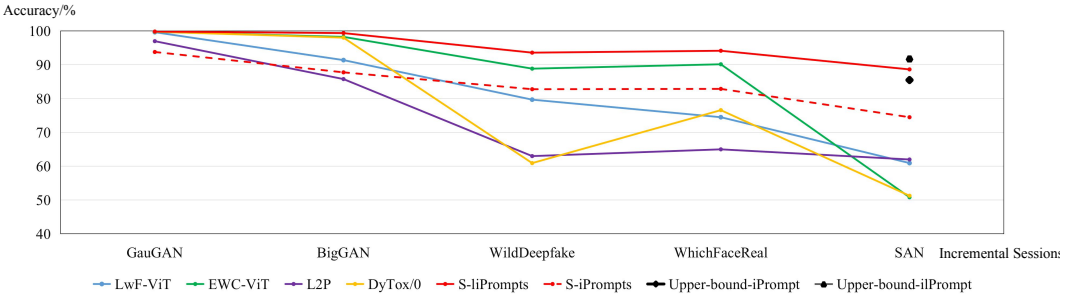

Figure 3: Accuracy curves of the competing methods and the proposed S-Prompts on CDDB-Hard.

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 $+0.05\%$ per domain. Compared to the other prompting methods DyTox and L2P, S-liPrompts enjoys much less memory overhead of the dynamic classifiers due to its language model based classifier design (i.e., extremely cheep prompts and shared class names) when the class number 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 supplementary material 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

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 [54]. * indicates results copied from [54].

Method	Buffer size	Average Acc (↑)
ER [8]		80.10±0.56*
GDumb [43]		74.92±0.25
BiC [55]		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 [54]		81.07±0.13*
EWC [28]		74.82±0.60*
LwF [34]		75.45±0.40*
L2P [54]	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 [54]. * indicates results copied from [16].

Method	Buffer size	Average Acc (↑)
DyTox [14]	50/class†	62.94
EWC [28]		47.62
LwF [34]		49.19
SimCLR [10]-CaSSLe [16]		44.2*
BYOL [17]-CaSSLe [16]		49.7*
Barlow Twins[58]-CaSSLe [16]		48.9*
Supervised Contrastive [26]-CaSSLe [16]	0/class	50.9*
L2P [54]		40.15
S-iPrompts (ours)		50.62
S-liPrompts (ours)		**67.78**
Upper-bound (S-iPrompts)	-	63.22
Upper-bound (S-liPrompts)	-	64.08

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[4].

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 dynamic FC and prompts in the final S-iPrompts and S-liPrompts approaches. We also study the case that the domains are randomly selected (i.e., without using K-NN to identify the related domains) 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.

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

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 Prompt length)	52.80	-29.13
S-liPrompts (fixed Language Prompt weights)	72.29	-0.99
S-liPrompts (fixed Image Prompt Length)	64.90	-12.7
S-liPrompts (K=5 & Domain Random Selection)	80.12	-1.04
S-liPrompts (K=5 & Zero-shot)	66.57	-
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**

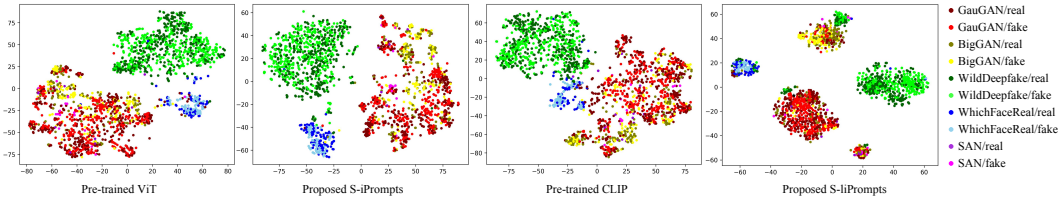

Figure 4: t-SNE visualization on the resulting feature spaces of pre-trained ViT, proposed S-iPrompts, pre-trained CLIP and proposed S-liPrompts on CDDB-Hard.

Table 5: Domain Identification Accuracy 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

5 Discussion and Conclusion

Continual learning studies the problem of learning from a data stream from changing domains/tasks. The general principal 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. It does not require the accumulation of the knowledge from previously learned domains. Instead, it merely learns the knowledge independently domain by domain, through simply prompting the state-of-the-art transformers. Though it additionally requires a domain identifier for the inference, the empirical studies show that the suggested prompting paradigm is fairly robust to the incorrect domain identification or even unseen domains. The increased prompting enjoys a great scalability to a large number of domains. Hence, following the suggested new learning rule, we recommend just learning a set of prompts independently on each new domain, and perform the inference with a rough domain identification.

There remain limitations on the suggested paradigm. For example, we do not obtain appealing results when applying it directly to the class-incremental learning 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. Besides, it might provide an inspiration to zero-shot and few-shot learning, since we discover that the joint language-image prompting S-liPrompts (66.57%) works better than the language prompting method CoOp [61] (65.75%) in the zero-shot scenario of CDDB.

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

[16] 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.

[17] 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. *NeurIPS*, 2020.

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

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

[20] 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.

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

[22] 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. *NeurIPS*, 2019.

[23] 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.

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

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

[26] 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.

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

[28] 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.

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

[30] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Advances in neural information processing systems*, 30, 2017.

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

[32] 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.

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

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

[35] 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.

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

[37] 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.

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

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

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

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

[42] 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.

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

[44] 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.

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

[46] 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.

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

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

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

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

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

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

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

[54] 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.

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

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

[57] 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.

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

[59] 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.

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

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

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

[63] 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]